Proclaiming Dictators and Juntas
or
Testing Boolean Formulae

Michal Parnas
The Academic College
of Tel-Aviv-Yaffo
Tel-Aviv, ISRAEL
michalp@mta.ac.il

Dana Ron*
Department of EE – Systems
Tel-Aviv University
Ramat Aviv, ISRAEL
danar@eng.tau.ac.il

Alex Samorodnitsky†
Institute for Advanced Study
Princeton, NJ 08540
asamor@ias.edu

December 6, 2001

Abstract

We consider the problem of determining whether a given function $f : \{0,1\}^n \to \{0,1\}$ belongs to a certain class of Boolean functions $\mathcal{F}$ or whether it is *far* from the class. More precisely, given query access to the function f and given a distance parameter ϵ, we would like to decide whether $f \in \mathcal{F}$ or whether it differs from every $g \in \mathcal{F}$ on more than an ϵ-fraction of the domain elements. The classes of functions we consider are singleton ("dictatorship") functions, monomials, and monotone DNF functions with a bounded number of terms. In all cases we provide algorithms whose query complexity is independent of n (the number of function variables), and polynomial in the other relevant parameters.

1 Introduction

The newly founded country of Eff is interested in joining the international organization Pea. This organization has one rule: It does not admit dictatorships. Eff claims it is not a dictatorship but is unwilling to reveal the procedure by which it combines the votes of its government members into a final decision. However, it agrees to allow Pea's special envoy, Tee, to perform a small number of experiments with its voting method. Namely, Tee may set the votes of the government members (using Eff's advanced electronic system) in any possible way, and obtain the final decision given these votes. Tee's mission is not to actually identify the dictator among the government members (if such exists), but only to discover *whether* such a dictator exists. Most importantly, she must do so by performing as few experiments as possible. Given this constraint, Tee may decline Eff's request to join Pea even if Eff is not exactly a dictatorship but only behaves like one most of the time.

The above can be formalized as a *Property Testing Problem*: Let $f : \{0,1\}^n \to \{0,1\}$ be a fixed (but unknown) function, and let $\mathcal{P}$ be a fixed property of functions. We would like to determine, by querying f, whether f has the property $\mathcal{P}$, or whether it is ϵ-*far* from having the property for a given distance parameter ϵ. By ϵ-*far* we mean that more than an ϵ–fraction of its values should be modified so that it obtains the property

*Supported by the Israel Science Foundation (grant number 32/00-1).
†Supported by NSF (grant number CCR–9987845) and by a State of New Jersey grant.

$\mathcal{P}$. For example, in the above setting we would like to test whether a given function f is a "dictatorship function". That is, whether there exists an index $1 \le i \le n$, such that $f(x) = x_i$ for every $x \in \{0,1\}^n$.

Previous work on testing properties of functions mainly focused on algebraic properties (e.g., [BLR93, RS96, Rub99]), or on properties defined by relatively rich families of functions such as the family of all monotone functions [GGL+00, DGL+99]. Here we are interested in studying the most basic families of Boolean functions: singletons, monomials, and DNF functions.

One possible approach to testing whether a function f has a certain property $\mathcal{P}$, is to try and actually *find* a good approximation for f from within the family of functions $\mathcal{F}_\mathcal{P}$ having the tested property $\mathcal{P}$. For this task we would use a *learning algorithm* that performs queries and works under the uniform distribution. Such an algorithm ensures that if f has the property (that is, $f \in \mathcal{F}_\mathcal{P}$), then with high probability the learning algorithm outputs a *hypothesis* $h \in \mathcal{F}_\mathcal{P}$ such that $\Pr[f(x) \ne h(x)] \le \epsilon$, where ϵ is a given distance (or error) parameter. The testing algorithm would run the learning algorithm, obtain the hypothesis $h \in \mathcal{F}_\mathcal{P}$, and check that h and f in fact differ only on a small fraction of the domain. This last step is performed by taking a sample of size $\Theta(1/\epsilon)$ from $\{0,1\}^n$ and comparing f and h on the sample. Thus, if f has property $\mathcal{P}$ then it will be accepted with high probability, and if f is ϵ-far from having $\mathcal{P}$ (so that $\Pr[f(x) \ne h(x)] > \epsilon$ for every $h \in \mathcal{F}_\mathcal{P}$, then it will be rejected with high probability.

Hence, provided there exists a learning algorithm for the tested family $\mathcal{F}_\mathcal{P}$, we obtain a testing algorithm whose complexity is of the same order of the learning algorithm. To be more precise, the learning algorithm should be a *proper* learning algorithm. That is, the hypothesis h it outputs must belong to $\mathcal{F}_\mathcal{P}$.[1]

A natural question that arises is whether we can do better by using a different approach. Recall that we are not interested in actually finding a good approximation for f in $\mathcal{F}_\mathcal{P}$ but we only want to know whether such an approximation *exists*. Therefore, perhaps we can design a different and more efficient testing algorithm than the one based on learning. In particular, the complexity measure we would like to improve is the *query complexity* of the algorithm.

As we show below, for all the properties we study, we describe algorithms whose query complexity is polynomial in $1/\epsilon$, where ϵ is the given distance parameter, and *independent* of the input size n.[2] As we discuss shortly, the corresponding proper learning algorithms have query complexities that depend on n, though only polylogarithmically. Thus our improvement is not so much of a quantitative nature (and we also note that our dependence on $1/\epsilon$ in some cases is worse). However, we believe that our results are of interest both because they completely remove the dependence on n in the query complexity, and also because in certain aspects they are inherently different from the corresponding learning algorithms. Hence they may shed new light on the structure of the properties studied.

1.1 Our Results.

We present the following testing algorithms:

- An algorithm that tests whether f is a singleton function. That is, whether there exists an index $1 \le i \le n$, such that $f(x) = x_i$ for every $x \in \{0,1\}^n$, or $f(x) = \bar{x}_i$ for every $x \in \{0,1\}^n$. This algorithm has query complexity $O(1/\epsilon)$.
- An algorithm that tests whether f is a monomial with query complexity $\tilde{O}(1/\epsilon^3)$.

[1] This is as opposed to *non-proper* learning algorithms that given query access to $f \in \mathcal{F}_\mathcal{P}$ are allowed to output a hypothesis h that belongs to a more general hypothesis class $\mathcal{F}' \supset \mathcal{F}_\mathcal{P}$. Non-proper learning algorithms are not directly applicable for our purposes.

[2] The running times of the algorithms are all linear in the number of queries performed and in n. This dependence on n in the running time is clearly unavoidable, since even writing down a query takes time n.

- An algorithm that tests whether f is a monotone DNF having at most ℓ terms, with query complexity $\tilde{O}(\ell^4/\epsilon^3)$.

Techniques. Our algorithms for testing singletons and for testing monomials have a similar structure. In particular, they combine two tests. One test is a "natural" test that arises from an exact logical characterization of these families of functions. In the case of singletons this test uniformly selects pairs $x, y \in \{0,1\}^n$ and verifies that $f(x \wedge y) = f(x) \wedge f(y)$, where $x \wedge y$ denotes the bitwise 'and' of the two strings. The corresponding test for monomials performs a slight variant of this test. The other test in both cases is a seemingly less evident test with an algebraic flavor. In the case of singletons it is a linearity test [BLR93] and in the case of monomials it is an affinity test. This test ensures that if f passes it then it has (or is close to having) a certain structure. This structure aids us in analyzing the logical test.

The testing algorithm for monotone DNF functions uses the test for monomials as a sub-routine. Recall that a DNF function is a disjunction of monomials (the terms of the function). If f is a DNF function with a bounded number of terms, then the test will isolate the different terms of the function and test that each is in fact a monomial. If f is far from being such a DNF function, then at least one of these tests will fail with high probability.

It is worthwhile noting that, given the structure of the monotone DNF tester, any improvement in the complexity of the monomial testing algorithm will imply an improvement in the DNF tester.

1.2 Related Work

Property Testing. Property testing was first defined and applied in the context of algebraic properties of functions [RS96], and has since been extended to various domains, perhaps most notably those of graph properties (e.g. [GGR98, GR97, AFKS99]). (For a survey see [Ron00]). The relation between testing and learning is discussed at length in [GGR98]. In particular, that paper suggests that testing may be applied as a preliminary stage to learning. Namely, efficient testing algorithms can be used in order to help in determining what hypothesis class should be used by the learning algorithm.

As noted above, we use linearity testing [BLR93] in our test for singletons, and affinity testing, which can be viewed as an extension of linearity testing, for testing monomials. Other works in which improvements and variants of linearity testing are analyzed include [BCH$^+$95, AHRS99]. In particular, the paper by Bellare et. al. [BCH$^+$95] is the first to establish the connection between linearity testing and Fourier analysis.

Learning Boolean Formulae. Singletons, and more generally monomials, can be easily learned under the uniform distribution. The learning algorithm uniformly selects a sample of size $\Theta(\log n/\epsilon)$ and queries the function f on all sample strings. It then searches for a monomial that is consistent with f on the sample. Finding a consistent monomial (if such exists) can be done in time linear in the sample size and in n. A simple probabilistic argument (that is a slight variant of Occam's Razor [BEHW87] [3]) can be used to show that a sample of size $\Theta(\log n/\epsilon)$ is sufficient to ensure that with high probability any monomial that is consistent with the sample is an ϵ-good approximation of f.

There is a large variety of results on learning DNF functions (and in particular monotone DNF), in several different models. We restrict our attention to the model most relevant to our work, namely when membership queries are allowed and the underlying distribution is uniform. The best known algorithm results from combining the works of [BJT99] and [KS99], and builds on Jackson's celebrated Harmonic

[3] Applying the theorem known as Occam's Razor would give a stronger result in the sense that the underlying distribution may be arbitrary (that is, not necessarily uniform). This however comes at a price of a linear, as opposed to logarithmic, dependence of the sample/query complexity on n.

Sieve algorithm [Jac97]. This algorithm has query complexity $\tilde{O}\left(r \cdot \left(\frac{\log^2 n}{\epsilon} + \frac{\ell^2}{\epsilon^2}\right)\right)$, where r is the number of variables appearing in the DNF formula, and ℓ is the number of terms. However, this algorithm does not output a DNF formula as its hypothesis. On the other hand, Angluin [Ang88] describes a proper learning algorithm for monotone DNF formula that uses membership queries and works under arbitrary distributions. The query complexity of her algorithm is $\tilde{O}(\ell \cdot n + \ell/\epsilon)$. Using the same preprocessing technique as suggested in [BJT99], if the underlying distribution is uniform then the query complexity can be reduced to $\tilde{O}\left(\frac{r \cdot \log^2 n}{\epsilon} + \ell \cdot \left(r + \frac{1}{\epsilon}\right)\right)$. Recall that the query complexity of our testing algorithm is a faster growing function of ℓ and $1/\epsilon$, but does not depend on n. Hence we get better results when ℓ and $1/\epsilon$ are sub-logarithmic in n, and in particular when they are constant.

Finally, we note that similarly to the Harmonic-Sieve based results for learning DNF, we appeal to the Fourier coefficients of the tested function f. However, somewhat differently, these do not appear explicitly in our algorithms but are only used in part of our analysis.

1.3 Organization

We start with some necessarily preliminaries in Section 2. In Section 3 we present our algorithm for testing singleton functions. The algorithm for testing monomials is presented in Section 4, and the algorithm for testing monotone DNF in Section 5. In Section 6 we discuss a possible (simpler) alternative to the singleton test, and in Section **??** we present an alternative analysis of the affinity test.

2 Preliminaries

Definition 1 *Let $x, y \in \{0,1\}^n$, and let $[n] \stackrel{\rm def}{=} \{1, \ldots, n\}$.*

- *We denote by $|x|$ the number of ones in the vector x.*
- *We write $y \succeq x$ if in each coordinate $y_i \geq x_i$.*
- *We let $2^x \stackrel{\rm def}{=} \{z \in \{0,1\}^n : z \preceq x\}$. Hence, $|2^x| = 2^{|x|}$.*
- *We let $x \wedge y$ denote the string $z \in \{0,1\}^n$ such that for every $i \in [n]$, $z_i = x_i \wedge y_i$.*
- *We let $x \oplus y$ denote the string $z \in \{0,1\}^n$ such that for every $i \in [n]$, $z_i = x_i \oplus y_i$.*

Definition 2 (Singletons, Monomials, and DNF functions) *A function $f : \{0,1\}^n \to \{0,1\}$ is a* singleton *function, if there exists an $i \in [n]$ such that $f(x) = x_i$ for every $x \in \{0,1\}^n$, or $f(x) = \bar{x}_i$ for every $x \in \{0,1\}^n$.*

We say that f is a monotone k-monomial *for $1 \leq k \leq n$ if there exist k indices $i_1, \ldots, i_k \in [n]$, such that $f(x) = x_{i_1} \wedge \cdots \wedge x_{i_k}$ for every $x \in \{0,1\}^n$. If we allow to replace some of the x_{i_j}'s above with $\bar{x}_{i_j}$, then f is a* k-monomial. *The function f is a* monomial *if it is a k-monomial for some $1 \leq k \leq n$.*

A function f is an ℓ-term DNF *function if it is a disjunction of ℓ monomials. If all monomials are monotone, then it is a* monotone DNF *function.*

When the identity of the function f is clear from the context, we may use the following notation.

Definition 3 *Define $F_0 \stackrel{\rm def}{=} \{x | f(x) = 0\}$ and $F_1 \stackrel{\rm def}{=} \{x | f(x) = 1\}$.*

Definition 4 (Distance between functions) *The distance (according to the uniform distribution) between two functions* $f, g : \{0,1\}^n \to \{0,1\}$ *is denoted by* $\mathrm{dist}(f,g)$*, and is defined as follows:* $\mathrm{dist}(f,g) \stackrel{\mathrm{def}}{=} \Pr_{x\in\{0,1\}^n}[f(x) \neq g(x)]$.

The distance between a function f *and a family of functions* $\mathcal{F}$ *is* $\mathrm{dist}(f,\mathcal{F}) \stackrel{\mathrm{def}}{=} \min_{g\in\mathcal{F}} \mathrm{dist}(f,g)$*. If* $\mathrm{dist}(f,\mathcal{F}) > \epsilon$ *for some* $0 < \epsilon < 1$*, then we say that* f *is* ϵ-far *from* $\mathcal{F}$*. Otherwise, it is* ϵ-close.

Definition 5 (Testing Algorithms) *A testing algorithm for a family of boolean functions* $\mathcal{F}$ *over* $\{0,1\}^n$ *is given a distance parameter* ϵ*,* $0 < \epsilon < 1$*, and is provided with query access to an arbitrary function* $f : \{0,1\}^n \to \{0,1\}$.

If $f \in \mathcal{F}$ *then the algorithm must output* accept *with probability at least* $2/3$*, and if* f *is* ϵ*-far from* $\mathcal{F}$ *then it must output* reject *with probability at least* $2/3$.

3 Testing Singletons

We start by presenting an algorithm for testing singletons. The testing algorithm for k-monomials will generalize this algorithm. More precisely, we present an algorithm for testing whether a function f is a *monotone* singleton. In order to test whether f is a singleton we can check whether either f or $\bar{f}$ passes the monotone singleton test. For the sake of succinctness, in what follows we refer to monotone singletons simply as singletons.

The following characterization of monotone k-monomials motivates our tests. We later show that the requirement of monotonicity can be removed.

Claim 1 *Let* $f : \{0,1\}^n \to \{0,1\}$*. Then* f *is a monotone* k*-monomial if and only if the following three conditions hold:*

1. $\Pr[f = 1] = \frac{1}{2^k}$;
2. $\forall x, y,\ f(x \wedge y) = f(x) \wedge f(y)$;
3. $f(x) = 0$ *for all* $|x| < k$.

Proof: If f is a k-monomial then clearly all the conditions hold. We turn to prove the other direction. Let $y = \bigwedge_{x\in F_1} x$. Using the second item in the claim we get:

$$f(y) = f(\bigwedge_{x\in F_1} x) = \bigwedge_{x\in F_1} f(x) = 1.$$

However, by the third item, $f(x) = 0$ for all $|x| < k$, and thus $|y| \geq k$. Hence, there exist k indices $i_1, ..., i_k$ such that $y_{i_j} = 1$ for all $1 \leq j \leq k$. But $y_{i_j} = \bigwedge_{x\in F_1} x_{i_j}$. Hence, $x_{i_1} = ... = x_{i_k} = 1$ for every $x \in F_1$. The first item now implies that $f(x) = x_{i_1} \wedge ... \wedge x_{i_k}$ for every $x \in \{0,1\}^n$. ■

Definition 6 *We say that* $x, y \in \{0,1\}^n$ *are a* violating pair *with respect to a function* $f : \{0,1\}^n \to \{0,1\}$*, if* $f(x) \wedge f(y) \neq f(x \wedge y)$.

Given the above definition, Claim 1 states that a basic property of (monotone) singletons (and more generally of monotone k-monomials), is that there are no violating pairs with respect to f. A natural candidate for a testing algorithm for singletons would take a sample of uniformly selected pairs x, y, and for each pair verify that it is not violating with respect to f. In addition, the test would check that $\Pr[f = 0]$ is roughly $1/2$ (or else any monotone k-monomial would pass the test).

As we discuss in Section 6, we were unable to give a complete proof for the correctness of this test. Somewhat unintuitively, the difficulty with the analysis lies in the case when the function f is *very far* from being a singleton. More precisely, the analysis is quite simple when the distance δ between f and the closest singleton is bounded away from $1/2$. However, the argument does not directly apply to δ arbitrarily close to $1/2$. We believe it would be interesting to prove that this simple test is in fact correct (or to come up with an example of a function f that is almost $1/2$-far from any singleton, but passes the test).

In the algorithm described below we circumvent the above difficulty by "forcing more structure" on f. Specifically, we first perform another test that only accepts functions that have, or more precisely, that are close to having a certain structure. In particular, every singleton will pass the test. We then perform a slight variant of our original test. Provided that f passes the first test, it will be easy to show that f passes the second test with high probability only if it is close to a singleton function. Details follow.

The algorithm begins by testing whether the function f belongs to a larger family of functions that contains singletons as a sub-family. This is the family of *parity functions*.

Definition 7 *A function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a* parity function *(a linear function over* $\mathrm{GF}(2)$*) if there exists a subset $S \subseteq [n]$ such that $f(x) = \oplus_{i \in S} x_i$ for every $x \in \{0,1\}^n$.*

The test for parity functions is a special case of the linearity test over general fields due to Blum, Luby and Rubinfeld [BLR93]. If the tested function f is a parity function, then the test always accepts, and if f is ϵ-far from any parity function then the test rejects with probability at least $9/10$. The query complexity of this test is $O(1/\epsilon)$. Assuming this test passes, we still need to verify that f is actually close to a singleton function and not to some other parity function. Suppose that the parity test only accepted proper parity functions. Then the following claim would suffice. It shows that if f is a non-singleton parity function, then a constant size sample of pairs x, y would, with high probability, contain a violating pair with respect to f.

Claim 2 *Let $g = \oplus_{i \in S} x_i$ for $S \subseteq [n]$. If $|S|$ is even then*

$$\Pr[g(x \wedge y) = g(x) \wedge g(y)] = \frac{1}{2} + \frac{1}{2^{|S|+1}}$$

and if $|S|$ is odd then

$$\Pr[g(x \wedge y) = g(x) \wedge g(y)] = \frac{1}{2} + \frac{1}{2^{|S|}}.$$

Proof: Let $s = |S|$, and let x, y be two strings such that (i) x has $0 \le i \le s$ ones in S, that is, $|\{\ell \in S : x_\ell = 1\}| = i$; (ii) $x \wedge y$ has $0 \le k \le i$ ones in S; and (iii) y has a total of $j + k$ ones in S, where $0 \le j \le s - i$.

If $g(x \wedge y) = g(x) \wedge g(y)$, then either (1) i is even and k is even, or (2) i is odd and j is even. Let $Z_1 \subset \{0,1\}^n \times \{0,1\}^n$ be the subset of pairs x, y that obey the first constraint, and let $Z_2 \subset \{0,1\}^n \times \{0,1\}^n$ be the subset of pairs x, y that obey the second constraint. Since the two subsets are disjoint,

$$\Pr[g(x \wedge y) = g(x) \wedge g(y)] = 2^{-2n} \cdot (|Z_1| + |Z_2|) \tag{1}$$

It remains to compute the sizes of the two sets. Since the coordinates of x and y outside S do not determine whether the pair x, y belongs to one of these sets, we have

$$|Z_1| = 2^{n-s} \cdot 2^{n-s} \cdot \left(\sum_{i=0, i \ even}^{s} \binom{s}{i} \sum_{k=0, k \ even}^{i} \binom{i}{k} \sum_{j=0}^{s-i} \binom{s-i}{j} \right) \tag{2}$$

and

$$|Z_2| = 2^{n-s} \cdot 2^{n-s} \cdot \left(\sum_{i=0, i \ odd}^{s} \binom{s}{i} \sum_{k=0}^{i} \binom{i}{k} \sum_{j=0, j \ even}^{s-i} \binom{s-i}{j} \right) \quad (3)$$

The first expression equals

$$2^{2n-2s} \cdot (2^{2s-2} + 2^{s-1}) = 2^{2n-2} + 2^{2n-s-1} = 2^{2n} \cdot (2^{-2} + 2^{-(s+1)}).$$

The second sum equals $2^{2n} \cdot (2^{-2} + 2^{-(s+1)})$ if s is odd and 2^{2n-2} if s is even. The claim follows by combining Equations (2) and (3) with Equation (1). ■

Hence, if f is a parity function that is not a singleton (that is $|S| \geq 2$), then the probability that a uniformly selected pair x, y is violating with respect to f is at least $1/8$. In this case, a sample of 16 such pairs will contain a violating pair with probability at least $1 - (1 - 1/8)^{16} \geq 1 - e^{-2} > 2/3$.

However, what if f passes the parity test but is only close to being a parity function? Let g denote the parity function that is closest to f and let δ be the distance between them. (Where g is unique, given that f is sufficiently close to a parity function). What we would like to do is check whether g is a singleton, by selecting a sample of pairs x, y and checking whether it contains a violating pair with respect to g. Observe that, since the distance between functions is measured with respect to the uniform distribution, for a uniformly selected pair x, y, with probability at least $(1 - \delta)^2$, both $f(x) = g(x)$ and $f(y) = g(y)$. However, we cannot make a similar claim about $f(x \wedge y)$ and $g(x \wedge y)$, since $x \wedge y$ is *not* uniformly distributed. Thus it is not clear that we can replace the violation test for g with a violation test for f.

The solution is to use a *self-corrector* for linear (parity) functions [BLR93]. Given query access to a function $f : \{0,1\}^n \rightarrow \{0,1\}$, which is strictly closer than $1/4$ to some parity function g, and an input $x \in \{0,1\}^n$, the procedure Self-Correct(f, x) returns the value of $g(x)$, with probability at least $9/10$. The query complexity of the procedure is constant.

The above discussion suggests the following testing algorithm.

Algorithm 1 Test for Singleton Functions

1. *Apply the parity test to f with distance parameter* $\min(1/5, \epsilon)$.
2. *Uniformly and independently select $m = 32$ pairs of points x, y.*
   - *For each such pair, let b_x = Self-Correct(f, x), b_y = Self-Correct(f, y) and $b_{x \wedge y}$ = Self-Correct$(f, x \wedge y)$.*
   - *Check that $b_{x \wedge y} = b_x \wedge b_y$.*
3. *If one of the above fails* - reject. *Otherwise,* accept.

Theorem 1 *Algorithm 1 is a testing algorithm for monotone singletons. Furthermore, it has one sided error. That is, if f is a monotone singleton, the algorithm always accepts. The query complexity of the algorithm is $O(1/\epsilon)$.*

Proof: Since the testing algorithm for parity functions has one-sided error, if f is a singleton function then it always passes the test. Similarly, in this case the self corrector always returns the value of f on the given input point, and clearly no violating pair can be found. Hence, the test always accepts a singleton.

Assume, without loss of generality, that $\epsilon \leq 1/5$. Consider the case in which f is ϵ-far from any singleton. If it is also ϵ-far from any parity function, then it will be rejected with probability at least $9/10$

in the first step of the algorithm. Otherwise, there exists a unique parity function g such that f is ϵ-close to g. By Claim 2, the probability that a uniformly selected pair x, y is a violating pair with respect to g is at least $1/8$. Given such a pair, the probability that the self-corrector returns the value of g on all the three calls (that is, $b_x = g(x)$, $b_y = g(y)$, and $b_{x \wedge y} = g(x \wedge y)$), is at least $(1 - 1/10)^3 > 7/10$. The probability that the algorithm obtains a violating pair with respect to g *and* all calls to the self corrector return the correct value, is greater than $1/16$. Therefore, a sample of 32 pairs will ensure that a violation $b_{x \wedge y} \neq b_x \wedge b_y$ will be found with probability at least $5/6$. The total probability that f is accepted, despite being ϵ-far from any singleton, is hence at most $1/10 + 1/6 < 1/3$.

The query complexity of the algorithm is dominated by the query complexity of the parity tester which is $O(1/\epsilon)$. The second stage takes constant time. ■

4 Testing Monomials

In this section we describe an algorithm for testing *monotone* k-monomials, where k is provided to the algorithm. We discuss later how to extend this to testing monomials when k is not specified. As for the monotonicity requirement, the following observation and a corollary show that this requirement can be easily removed, if desired.

Observation 3 *Let $f : \{0,1\}^n \rightarrow \{0,1\}$, and let $z \in \{0,1\}^n$. Consider the function $f_z : \{0,1\}^n \rightarrow \{0,1\}$ which is defined as follows: $f_z(x) = f(x \oplus z)$. Then the following are immediate:*

1. *The function f is a k-monomial if and only if f_z is a k-monomial.*
2. *Let $y \in F_1$. If f is a (not necessarily monotone) k-monomial, then $f_{\bar{y}}$ is a monotone k-monomial.*

Corollary 4 *If f is ϵ-far from every (not necessarily monotone) k-monomial, then for every $y \in F_1$, $f_{\bar{y}}$ is ϵ-far from every monotone k-monomial.*

We now present the algorithm for testing monotone k-monomials. The first two steps of the algorithm are an attempt to generalize the application of parity testing in Algorithm 1. Specifically, we test whether F_1 is an *affine subspace*.

Definition 8 (Affine Subspaces) *A subset $H \subseteq \{0,1\}^n$ is an* affine subspace *of $\{0,1\}^n$ if and only if there exist an $x \in \{0,1\}^n$ and a linear subspace V of $\{0,1\}^n$, such that $H = V \oplus x$. That is,*

$$H = \{y \mid y = v \oplus x, for\ some\ v \in V\}.$$

The following is a well known alternative characterization of affine subspaces, which is a basis for our test.

Fact 5 *H is an affine subspace if and only if for every $y_1, y_2, y_3 \in H$ we have $y_1 \oplus y_2 \oplus y_3 \in H$.*

In the following description of the algorithm, there is an explicit (exponential) dependence on the size k of the monomial. However, as we observe momentarily, we can assume, without loss of generality, that $2^k = O(1/\epsilon)$, and so the query complexity of the algorithm is actually polynomial in $1/\epsilon$.

Algorithm 2 Test for monotone k-monomials

1. Size Test: *Uniformly select a sample of* $m = \Theta(2^{2k} + 1/\epsilon^2)$ *strings in* $\{0,1\}^n$. *For each* x *in the sample, obtain* $f(x)$. *Let* α *be the fraction of sample strings* x *such that* $f(x) = 1$. *If* $|\alpha - 2^{-k}| > \min(2^{-k-5}, \epsilon/4)$ *then reject, otherwise continue.*

2. Affinity Test: *Repeat the following* $3k^2 \cdot \max(2/\epsilon^2, 2^{2(k+4)})$ *times:*

   Uniformly select $x, y \in F_1$ *and* $z \in \{0,1\}^n$ *and check whether* $f(x \oplus y \oplus z) = f(x) \oplus f(y) \oplus f(z)$. *If some triple does not satisfy this constraint then reject.*

   (Since $f(x) = f(y) = 1$, we are actually checking whether $f(x \oplus y \oplus z) = f(z)$. As we show in our analysis, this step will ensure that f is close to some function g for which $g(x) \oplus g(y) \oplus g(z) = g(x \oplus y \oplus z)$ for all $x, y, z \in G_1 = \{x | g(x) = 1\}$.)

3. Closure-Under-Intersection Test: *Repeat the following* 2^{k+5} *times:*

   (a) *Uniformly select* $x \in F_1$ *and* $y \in \{0,1\}^n$. *If* x *and* y *are a violating pair, then reject.* (Note that since $x \in F_1$, this test actually checks that $f(y) = f(x \wedge y)$.)

4. *If no step caused rejection, then accept.*

In both the affinity test and the closure-under-intersection test, we need to select strings in F_1 uniformly. This is simply done by sampling from $\{0,1\}^n$ and using only x's for which $f(x) = 1$. This comes at an additional multiplicative cost of $O(2^k)$ in the query complexity. The query complexity of the algorithm is hence $O(2^k \cdot k^2 \cdot \max(1/\epsilon^2, 2^{2k}))$. As the following observation implies, we may assume, without loss of generality, that $\epsilon < 2^{-k+2}$, and so the complexity is actually $\tilde{O}(1/\epsilon^3)$.

Observation 6 *Suppose that* $\epsilon \geq 2^{-k+2}$. *Then:*

1. *If* $\Pr[f = 1] \leq \frac{\epsilon}{2}$, *then* f *is* ϵ*-close to every* k*-monomial (and in particular to every monotone* k*-monomial).*

2. *If* $\Pr[f = 1] > \frac{\epsilon}{4}$, *then* f *is not a* k*-monomial.*

Proof: If $\Pr[f = 1] \leq \frac{\epsilon}{2}$ then for every k-monomial g,

$$\mathrm{dist}(f, g) = \Pr[f = 1 \wedge g = 0] + \Pr[f = 0 \wedge g = 1] \leq \Pr[f = 1] + \Pr[g = 1] \leq \frac{\epsilon}{2} + 2^{-k} \leq \epsilon$$

Since $\epsilon \geq 2^{-k+2}$, if $\Pr[f = 1] > \frac{\epsilon}{4}$ then $\Pr[f = 1] > 2^{-k}$, while by the definition of a k-monomial, $\Pr[f = 1] = 2^{-k}$. ■

We now embark on proving the correctness of the algorithm.

Theorem 2 *Algorithm 2 is a testing algorithm for monotone* k*-monomials. The query complexity of the algorithm is* $\tilde{O}(1/\epsilon^3)$.

The proof of Theorem 2 is based on the following two lemmas whose proofs are provided in Subsections 4.1 and 4.2 respectively.

Lemma 7 *Let* $\eta \stackrel{\mathrm{def}}{=} \Pr_{x,y \in F_1, z \in \{0,1\}^n}[f(x \oplus y \oplus z) \neq f(z)]$. *If* $\eta \leq 2^{-2k-1}$ *and* $\left|\frac{|F_1|}{2^n} - 2^{-k}\right| \leq 2^{-(k+3)}$, *then there exists a function* g *such that* $G_1 \stackrel{\mathrm{def}}{=} \{x : g(x) = 1\}$ *is an affine subspace of dimension* $n - k$ *and which satisfies:*

$$\mathrm{dist}(f, g) \leq \left|\frac{|F_1|}{2^n} - 2^{-k}\right| + k\eta^{\frac{1}{2}}.$$

Lemma 8 *Let $f : \{0,1\}^n \to \{0,1\}$ be a function for which $|\Pr[f = 1] - 2^{-k}| < 2^{-k-3}$. Suppose that there exists a function $g : \{0,1\}^n \to \{0,1\}$ such that:*

1. $\mathrm{dist}(f, g) \leq 2^{-k-3}$.

2. $G_1 \stackrel{\mathrm{def}}{=} \{x : g(x) = 1\}$ *is an affine subspace of dimension $n - k$.*

If g is not a monotone k-monomial, then

$$\Pr_{x \in F_1, y}[f(y) \neq f(x \wedge y)] \geq 2^{-k-3}.$$

Proof of Theorem 2: We assume that the constants in the $\Theta(\cdot)$ notation in the first step of Algorithm 2 are sufficiently large.

If f is a monotone k-monomial, then $\Pr[f = 1] = 2^{-k}$. By Chernoff's bound, the probability that it is rejected in the first step of Algorithm 2 is less than 1/3. By the definition of k-monomials, f always passes the affinity test and the closure-under-intersection test.

Suppose that f is ϵ-far from any monotone k-monomial. We show that it is rejected with probability greater than $2/3$.

1. If $|\Pr[f = 1] - 2^{-k}| > \min\{\epsilon/2, 2^{-(k-4)}\}$, then f is rejected in the first step of the algorithm with probability at least $9/10$.

2. Otherwise, $|\Pr[f = 1] - 2^{-k}| \leq \min\{\epsilon/2, 2^{-(k-4)}\}$. If η, as defined in Lemma 7, is greater than $k^{-2} \cdot \min\{\epsilon/2, 2^{-2k-8}\}$, then f is rejected with probability at least $9/10$ in the second step of the algorithm (the affinity test).

3. Otherwise, both $|\Pr[f = 1] - 2^{-k}| \leq \min\{\epsilon/2, 2^{-(k-4)}\}$ and $\eta \leq k^{-2} \cdot \min\{\epsilon/2, 2^{-2k-8}\}$. Now we can apply Lemma 7 and obtain that there exists a function g as required in Lemma 8. But now, since f is assumed to be ϵ-far from any monotone k-monomial, the function g cannot be a monotone k-monomial. Hence, by Lemma 8, f will be rejected with probability at least $9/10$ in the third step of the algorithm (the closure-under-intersection test).

Thus, the probability that f is accepted by the algorithm is at most $3/10 < 1/3$, as required. ■

4.1 Analysis of the Affinity Test

In this subsection we prove Lemma 7 using tools from Fourier analysis. An alternative proof, which builds on basic probabilistic principles, is given in Section **??**. One benefit of the alternative proof is that it suggests a self-corrector for functions f that pass the affinity test. However, we tend to believe that the proof described below is simpler, given the basic building blocks provided by Fourier analysis. We start with some needed background and notation concerning Discrete Fourier Transform.

4.1.1 Discrete Fourier Transform

We denote by $\mathbf{E}_{x \in A}[f(x)]$ the expectation of a function f, when x in chosen uniformly in the set $A \subseteq \{0,1\}^n$, namely $\mathbf{E}_{x \in A}[f(x)] = \frac{1}{|A|} \sum_{x \in A} f(x)$. We denote by $\mathbf{E}_x[f(x)]$ the expectation of f over the whole space $\{0,1\}^n$.

For $1 \leq i \leq n$, let $r_i : \{0,1\}^n \to \{-1,1\}$ be a function defined by $r_i(x_1,...,x_n) = 1$ if $x_i = 0$, and $r_i(x_1,...,x_n) = -1$ otherwise. For any $S \subseteq [n]$, define a function $w_S : \{0,1\}^n \to \{-1,1\}$ by $w_S(x) = \prod_{i \in S} r_i(x)$, where $w_\emptyset(x) = 1$. The function w_S is the *Walsh function* indexed by S. [4]

There are 2^n Walsh functions, one for every subset $S \subseteq [n]$, and they are an orthonormal base of the space of real functions on $\{0,1\}^n$, under an inner product given by :

$$\langle f, g \rangle \stackrel{\mathrm{def}}{=} \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \cdot g(x) = \mathbf{E}_x[f(x) \cdot g(x)] \ .$$

Any function $f : \{0,1\}^n \to \Re$ can be represented as a linear combination of Walsh functions. The coefficient of w_S in this representation is called the S-Fourier coefficient of f, and is denoted by $\hat{f}(S)$. Namely, $f = \sum_{S \subseteq [n]} \hat{f}(S) \cdot w_S$. Since Walsh functions are orthonormal, we have $\hat{f}(S) = \langle f, w_S \rangle = \mathbf{E}_x[f(x) \cdot w_S(x)]$.

The *convolution* of two functions f, g is denoted by $f * g$, and is defined by $(f*g)(y) \stackrel{\mathrm{def}}{=} \mathbf{E}_x[f(x) \cdot g(x \oplus y)]$ We will need the following important property of convolution:

$$(\widehat{f * g})(S) = \hat{f}(S) \cdot \hat{g}(S).$$

And we will also need several simple facts about Fourier coefficients. For any two functions f, g:

$$\mathbf{E}_x[f(x) \cdot g(x)] = \sum_{S \subseteq [n]} \hat{f}(S) \cdot \hat{g}(S) \ .$$

In particular, Parseval's equality holds:

$$\mathbf{E}_x[f^2(x)] = \sum_{S \subseteq [n]} \hat{f}^2(S). \qquad (4)$$

By definition, it is also easy to verify that:

$$\mathbf{E}_x[f(x)] = \hat{f}(\emptyset), \quad \sum_S \hat{f}(S) = f(0), \quad \mathbf{E}_x[f(x) \cdot g(x)] = (f * g)(0)$$

4.1.2 Proof of Lemma 7

In order to prove Lemma 7, we need to show the existence of an affine subspace G that is close to F_1 (that is, the symmetric difference between G and F_1 is relatively small). The definition of this affine subspace will based on the location of the large Fourier coefficients of f. Let $p \stackrel{\mathrm{def}}{=} \frac{|F_1|}{2^n}$, and recall that $\eta \stackrel{\mathrm{def}}{=} \Pr_{x,y \in F_1, z \in \{0,1\}^n}[f(x \oplus y \oplus z) \neq f(z)]$. We begin by relating the Fourier coefficients of f to p and η.

Claim 9 $\sum_S \hat{f}^4(S) = p^3 - \frac{\eta}{2} p^2$

Proof: Let $h : \{0,1\}^n \to \{+1,-1\}$ be defined as follows: $h(x) = 1$ for every x such that $f(x) = 0$, and $h(x) = -1$ for every x such that $f(x) = 1$. In other words: $h = 1 - 2f$. Using this notation, the affinity test uniformly picks $x, y \in F_1$ and $z \in \{0,1\}^n$, and checks whether $h(x \oplus y \oplus z) \cdot h(z) = 1$.

[4]Note that Walsh functions are essentially the parity functions on $\{0,1\}^n$, but written in a *multiplicative notation.* If we define $\tilde{w}_S(x)$ to be 0 if $w_S(x) = 1$, and 1 if $w_S(x) = -1$, then the functions $\tilde{w}_S$ are precisely the parity functions on $\{0,1\}^n$.

By the definition of η,

$$\begin{aligned} 1-2\eta &= \mathbf{E}_{x,y\in F_1,z}[h(z)\cdot h(x\oplus y\oplus z)] \\ &= \mathbf{E}_{x,y\in F_1}[\mathbf{E}_z[h(z)\cdot h(x\oplus y\oplus z)]] \\ &= \mathbf{E}_{x,y\in F_1}[(h*h)(x\oplus y)]\ . \end{aligned}$$

By the definition of h,

$$\mathbf{E}_{x,y\in F_1}[(h*h)(x\oplus y)] = \frac{2^{2n}}{|F_1|^2}\cdot \mathbf{E}_{x,y}[f(x)\cdot f(y)\cdot (h*h)(x\oplus y)].$$

Recall that $f=\frac{1-h}{2}$ and hence

$$\begin{aligned} &\frac{2^{2n}}{|F_1|^2}\cdot \mathbf{E}_{x,y}[f(x)\cdot f(y)\cdot (h*h)(x\oplus y)] \\ &= \frac{1}{p^2}\cdot \mathbf{E}_{x,y}\left[\frac{1-h(x)}{2}\cdot\frac{1-h(y)}{2}\cdot (h*h)(x\oplus y)\right] \\ &= \frac{1}{4p^2}\mathbf{E}_{x,y}\left[(h*h)(x\oplus y)\ -\ (h(x)+h(y))\cdot(h*h)(x\oplus y)\ +\ h(x)\cdot h(y)\cdot(h*h)(x\oplus y)\right]\ . \end{aligned}$$

We open the brackets and compute the expectations one by one.

$$\mathbf{E}_{x,y}[(h*h)(x\oplus y) = \mathbf{E}_y[\mathbf{E}_x[(h*h)(x\oplus y)]] = \mathbf{E}_y[\mathbf{E}_z[(h*h)(z)]] = \mathbf{E}_z[(h*h)(z)] = \hat{h}^2(\emptyset),$$

and

$$\mathbf{E}_{x,y}[h(x)\cdot(h*h)(x\oplus y)] = \mathbf{E}_y[\mathbf{E}_x[h(x)\cdot(h*h)(x\oplus y)]] = \mathbf{E}_y[(h*h*h)(y)] = \hat{h}^3(\emptyset)\ .$$

Similarly,

$$\mathbf{E}_{x,y}[h(y)\cdot(h*h)(x\oplus y)]] = \hat{h}^3(\emptyset).$$

Finally,

$$\begin{aligned} \mathbf{E}_{x,y}[h(x)\cdot h(y)\cdot(h*h)(x\oplus y)]] &= \mathbf{E}_y[h(y)\cdot\mathbf{E}_x[h(x)\cdot(h*h)(x\oplus y)]] \\ &= \mathbf{E}_y[h(y)\cdot(h*h*h)(y)] \\ &= (h*h*h*h)(0) = \sum_S \hat{h}^4(S)\ . \end{aligned}$$

Observe that $\hat{h}(\emptyset) = \mathbf{E}_x[h(x)] = 1-\frac{2|F_1|}{2^n} = 1-2p$, and that $h = 1-2f$. Therefore, $\hat{h}(S) = -2\hat{f}(S)$, for all $S\neq\emptyset$. Note also that $\hat{f}(\emptyset)=p$. Taking all this into consideration we have:

$$\begin{aligned} 1-2\eta &= \frac{1}{4p^2}\left((1-2p)^2 - 2(1-2p)^3 + (1-2p)^4 + 16\cdot\sum_S \hat{f}^4(S) - 16p^4\right) \\ &= (1-2p)^2 - 4p^2 + 4\frac{\sum_S \hat{f}^4(S)}{p^2} \\ &= 1-4p+4\frac{\sum_S \hat{f}^4(S)}{p^2}\ . \end{aligned}$$

Claim 9 follows. ■

Assume that η is small, in particular $\eta \le p^2$. In this case, Claim 9 implies that $\sum_S \hat{f}^4(S)$ is very close to p^3. Note that, since f is nonnegative, for any $S \subseteq [n]$:

$$\hat{f}(S) \le |\hat{f}(S)| \le \hat{f}(\emptyset) = p.$$

We now show that for many subsets S, $|\hat{f}(S)|$ is actually very close to p. We later use these subsets to define the affine subspace G that is close to F_1. To this end we define the following collections of subsets:

$$\text{small} \stackrel{\text{def}}{=} \{S : |\hat{f}(S)| < p - \eta^{\frac{1}{2}}\}, \quad \text{big} \stackrel{\text{def}}{=} \{S : |\hat{f}(S)| \ge p - \eta^{\frac{1}{2}}\}.$$

Claim 10 *If* $|p - 2^{-k}| \le 2^{-(k+3)}$ *and* $\eta \le p^2$*, then* $|\text{big}| \ge 2^{k-1} + 1$.

Proof: Using Claim 9, we have:

$$\begin{aligned} p^3 - \frac{p^2\eta}{2} &= \sum_S \hat{f}^4(S) \\ &= \sum_{S \in \text{small}} \hat{f}^4(S) + \sum_{S \in \text{big}} \hat{f}^4(S) \\ &\le (p - \eta^{\frac{1}{2}})^2 \cdot \sum_{S \in \text{small}} \hat{f}^2(S) + p^2 \cdot \sum_{S \in \text{big}} \hat{f}^2(S)\ . \end{aligned}$$

Let $r = \sum_{S \in \text{small}} \hat{f}^2(S)$. Using Equation (4), $\sum_S \hat{f}^2(S) = \mathbf{E}_x[f^2(x)] = p$. Thus,

$$\begin{aligned} p^3 - \frac{\eta}{2}p^2 &\le (p - \eta^{\frac{1}{2}})^2 r + p^2(p - r) \\ &= p^3 - r \cdot (p^2 - (p - \eta^{\frac{1}{2}})^2) \\ &= p^3 - r\eta^{\frac{1}{2}}(2p - \eta^{\frac{1}{2}}) \le p^3 - rp\eta^{\frac{1}{2}}\ . \end{aligned}$$

The last inequality is based on our assumption that $\eta \le p^2$. Therefore, $r \le \frac{\eta^{\frac{1}{2}}}{2}p$. It follows that:

$$p - r = \sum_{S \in \text{big}} \hat{f}^2(S) \ge p - \frac{\eta^{\frac{1}{2}}}{2}p.$$

But $\hat{f}(S) \le p$, and therefore:

$$|\text{big}| \ \ge\ \frac{p - \frac{\eta^{\frac{1}{2}}}{2}p}{p^2} \ =\ \frac{1 - \frac{\eta^{\frac{1}{2}}}{2}}{p} \ \ge\ \frac{1}{p} - \frac{1}{2}.$$

Since $|p - 2^{-k}| < 2^{-(k+3)}$, Claim 10 follows ■

Observe that in a set of size $\ge 2^{k-1} + 1$ there are at least k linearly independent vectors (otherwise the set lies in a span of at most $k - 1$ vectors, which is, obviously, of size 2^{k-1}). We use this in the following claim.

Claim 11 *Suppose that* $|p - 2^{-k}| \le 2^{-(k+3)}$ *and* $\eta \le p^2$*. Let* $S_1, \ldots, S_k \in \text{big}$ *be* k *linearly independent vectors. Let* $\beta_i = 0$ *if* $\hat{f}(S_i) > 0$*, and* $\beta_i = 1$ *if* $\hat{f}(S_i) < 0$*. Define*

$$G \stackrel{\text{def}}{=} \{y \ :\ \langle y, S_i \rangle = \beta_i,\ i = 1...k\}$$

where $\langle y, T \rangle = \bigoplus_{j=1}^n y_j \cdot T_j$*. Then* G *is an affine subspace of dimension* $n - k$*, and* $\text{dist}(G, F_1) \le |p - 2^{-k}| + k\eta^{\frac{1}{2}}$*, where* $\text{dist}(G, F_1) \stackrel{\text{def}}{=} 2^{-n} \cdot (|G \setminus F_1| + |F_1 \setminus G|)$.

Proof: Let M be a $k \times n$ matrix with rows $S_1, ..., S_k$. Consider the linear transformation from $\{0,1\}^n$ to $\{0,1\}^k$, taking x to Mx. Since the rows of M are linearly independent, the rank of M is k and the linear transformation is *onto*. The set G is a pre-image of the vector $(\beta_1, ..., \beta_k)$ in $\{0,1\}^k$ and therefore is an affine subspace of dimension $n-k$.

We turn to the second part of the claim. We first show that for any fixed $1 \leq i \leq k$:

$$|F_1 \cap \{y \ : \ \langle y, S_i \rangle \neq \beta_i\}| \leq 2^{n-1}\eta^{1/2}$$

Since f is the characteristic function of F_1,

$$\begin{aligned}
\hat{f}(S_i) = \frac{1}{2^n}\sum_{x \in F_1}(-1)^{\langle x, S_i\rangle} &= \frac{1}{2^n}\left(|F_1| - 2\Big|\{y \in F_1 : \langle y, S_i\rangle = 1\}\Big|\right) \\
&= p - \frac{1}{2^{n-1}}\Big|\{y \in F_1 : \langle y, S_i\rangle = 1\}\Big|.
\end{aligned}$$

Assume that $\beta_i = 0$, and therefore, $\hat{f}(S_i) \geq p - \eta^{1/2}$. It follows that,

$$\Big|\{y \in F_1 \ : \ \langle y, S_i\rangle = 1\}\Big| \ = \ \Big|F_1 \cap \{y : \langle y, S_i\rangle \neq \beta_i\}\Big| \ \leq \ 2^{n-1}\eta^{1/2}.$$

The case that $\beta_i = 1$ and $\hat{f}(S_i) \leq -(p - \eta^{\frac{1}{2}})$ is similar. Therefore, we have:

$$\begin{aligned}
|F_1 \cap G| &= \Big|F_1 \cap \bigcap_{i=1}^{k}\{y \ : \ \langle y, S_i\rangle = \beta_i\}\Big| \\
&\geq |F_1| - \sum_{i=1}^{k}\Big|F_1 \cap \{y \ : \ \langle y, S_i\rangle \neq \beta_i\}\Big| \\
&\geq |F_1| - 2^{n-1}k\eta^{\frac{1}{2}}.
\end{aligned}$$

So

$$\begin{aligned}
\mathrm{dist}(G, F_1) &= \frac{1}{2^n}\left(|G| + |F_1| - 2|G \cap F_1|\right) \\
&\leq \frac{1}{2^n}\left((|G| - |F_1|) + 2^n k\eta^{\frac{1}{2}}\right) \\
&= \frac{1}{2^k} - \frac{|F_1|}{2^n} + k\eta^{\frac{1}{2}} \\
&\leq \left|p - 2^{-k}\right| + k\eta^{\frac{1}{2}}
\end{aligned}$$

and we are done. ■

Proof of Lemma 7: The proof follows immediately from Claims 10 and 11. ■

4.2 Analysis of the Closure-Under-Intersection Test

We first recall several simple properties of affine spaces.

Claim 12 *Let H be an affine subspace such that $H = V \oplus x$, where $x \in \{0,1\}^n$ and $V \subseteq \{0,1\}^n$ is a linear subspace. Then,*

1. $x \in H$.

2. *For every $z \in H$ we have $H = V \oplus z$. By the definition of the $\oplus$ operator, we thus also have that $V = H \oplus z$, for every $z \in H$.*

3. $|H| = |V| = 2^{dimV}$.

Claim 13 *Let H, H' be two affine subspaces of $\{0,1\}^n$, such that $H \not\subseteq H'$. Then:*

$$\frac{|H \cap H'|}{|H|} \leq \frac{1}{2}.$$

Proof: The claim follows from the corresponding property of linear subspaces, namely $V \not\subseteq V'$ implies: $\frac{|V \cap V'|}{|V|} \leq \frac{1}{2}$. ■

The following corollary is immediate:

Corollary 14 *Let H, H' be two affine subspaces of $\{0,1\}^n$ such that $H' \subseteq H$. Then either $H' = H$, or $|H'| \leq |H|/2$.*

Claim 15 *Let H, H' be two affine subspaces of $\{0,1\}^n$ such that $H' \subseteq H$ and let $y \in H'$. Denote by V' the linear subspace such that $H' = V' \oplus y$, and by V the linear subspace such that $H = V \oplus y$. Then:*

1. $V' \subseteq V$.

2. *For any $x \in V$ we have $(H' \oplus x) \subseteq H$, and for any $x \notin V$ we have $(H' \oplus x) \cap H = \emptyset$.*

Proof: By definition, $V' = H' \oplus y \subseteq H \oplus y = V$. This proves the first part of the lemma.

Now let $x \in V$. Since V and V' are linear subspaces and $V' \subseteq V$, then $(V' \oplus x) \subseteq V$. Thus, $H' \oplus x = (V' \oplus y) \oplus x = (V' \oplus x) \oplus y \subseteq V \oplus y = H$. On the other hand, let $x \notin V$. Observe that $(V' \oplus x) \cap V = \emptyset$. Since $H' \oplus x = (V' \oplus y) \oplus x = (V' \oplus x) \oplus y$, we get that $(H' \oplus x) \cap H = (V' \oplus x) \oplus y \cap (V \oplus y) = \emptyset$. This concludes the proof of the claim. ■

To prove Lemma 8 we will need several auxiliary claims. The first claim relates affine spaces that correspond to k-monomials and monotonicity.

Claim 16 *Let H be an affine subspace of $\{0,1\}^n$ of size 2^{n-k}. Assume also that H is monotone. Namely, if $x \in H$ and $y \succeq x$, then $y \in H$. Then $H = \{x : x_{i_1} = 1 ... x_{i_k} = 1\}$, for some subset $i_1 ... i_k$ of coordinates.*

Proof: Let V be an $n-k$ dimensional linear subspace and let $y \in \{0,1\}^n$ be such that $H = V \oplus y$. Let $v_1 ... v_{n-k}$ be a basis of V. Consider an $(n-k) \times n$ matrix with rows $v_1 ... v_{n-k}$. Its rank is $n-k$, and therefore it has $n-k$ linearly independent columns. Without loss of generality, these are the first $n-k$ columns. Therefore the restriction of the rows to the first $n-k$ coordinates is a basis of $\{0,1\}^{n-k}$, and thus it spans all the vectors in $\{0,1\}^{n-k}$ and in particular the first $n-k$ coordinates of y. It follows that there is a vector $v \in V$, namely a linear combination of the rows, that is equal to y on the first $n-k$ coordinates. Therefore, $z = (v \oplus y) \in H$ is 0 on the first $n-k$ coordinates.

Since H is monotone, if $|z| < k$, or there exists a $z' \not\succeq z$ such that $z' \in H$, then $|H| > 2^{n-k}$, contradicting our assumption on H. Hence $H = \{x : x_{i_1} = 1, \cdots, x_{i_k} = 1\}$ where $i_1, \ldots, i_k$ are the coordinates on which z is 1. ■

Recall that by the premise of Lemma 8, there exists a function g such that $\mathrm{dist}(f,g) \leq 2^{-k-3}$, and $G_1 \stackrel{\mathrm{def}}{=} \{x : g(x) = 1\}$ is an affine subspace of dimension $n-k$. Claim 16 implies that if g is not a k-monomial, then the affine subspace G_1 cannot be monotone. We shall use this, together with the fact that f and g are close, to prove that there are many pairs $x \in F_1, y \in \{0,1\}^n$ such that $f(y) \neq f(x \wedge y)$. To this end we define the following subsets.

Definition 9 *Let $x \in \{0,1\}^n$ and $z \in 2^x$. Define $G(x,z) \stackrel{\mathrm{def}}{=} \{y | \; x \wedge y = z\}$.*

We shall show that for many pairs (x,z), with $x \in G_1$ and $z \in 2^x$, the function g is far from constant on $G(x,z)$. Since the functions f and g are close to each other, this will imply the existence of many violating pairs, as desired. First, we prove some properties of the subsets $G(x,z)$.

Claim 17 *For every $x \in \{0,1\}^n$ and $z \in 2^x$, $G(x,z)$ is an affine subspace of $\{0,1\}^n$ of size $2^{n-|x|}$. Furthermore, for every $x \in \{0,1\}^n$, the affine subspaces $\{G(x,z)\}_{z \in 2^x}$ partition $\{0,1\}^n$.*

Proof: These facts about $G(x,z)$ follow easily from the following observation: for a fixed x, the map $m_x : y \to x \wedge y$ is a linear map from $\{0,1\}^n$ to 2^x, and $G(x,z) = m_x^{-1}(z)$. ■

Claim 18 *Let $x \in G_1$ be such that there exists $z \in 2^x$ for which $G(x,z) \subseteq G_1$. Then, $G(x,x) \subseteq G_1$.*

Proof: We first show that $G(x,z) \oplus x \oplus z \subseteq G_1$. Since G_1 is an affine subspace, by Fact 5, is it enough to show that x and z lie in G_1, and that $G(x,z)$ is a subset of G_1. Taking into account the assumptions of the Claim, we only need to show $z \in G_1$. Since $z \preceq x$, we have $z \wedge x = z$. Hence, $z \in G(x,z) \subseteq G_1$.

Next, we show that $G(x,x) \subseteq G(x,z) \oplus x \oplus z$. Take $y \in G(x,x)$. Now, define y' as follows. If $z_i = 1$, then $y'_i = 1$ (in this case always $x_i = 1$). If $z_i = 0$ and $x_i = 1$, then $y'_i = 0$, and if $z_i = 0$ and $x_i = 0$, then $y'_i = y_i$. Thus, $y' \wedge x = z$ and so $y' \in G(x,z)$. It is also easy to verify that $y' \oplus x \oplus z = y$. (Note that $y \succeq x$, and therefore $x_i = 1$ implies $y_i = 1$). Hence, $y \in G(x,z) \oplus x \oplus z$. Since we have shown that $G(x,z) \oplus x \oplus z \subseteq G_1$, the claim follows. ■

We shall be interested in the following set:

$$\mathcal{X} \stackrel{\mathrm{def}}{=} \{x \in G_1 : \; G(x,x) \subseteq G_1\} \tag{5}$$

Thus $\mathcal{X}$ consists of those $x \in G_1$ for which every $y \succeq x$ is in G_1. Since, by Claim 16, G_1 is not monotone, necessarily $\mathcal{X} \neq G_1$. As we show momentarily, $\mathcal{X}$ is actually significantly smaller than G_1, and we shall exploit this in our proof.

Claim 19 *The set $\mathcal{X}$ is an affine subspace of G_1. Furthermore, if g is not a k-monomial then $|\mathcal{X}| \leq \frac{1}{2}|G_1|$.*

Proof: By Fact 5, in order to prove the first part of the lemma it suffices to show that for every $x^1, x^2, x^3 \in \mathcal{X}$, we have $x^1 \oplus x^2 \oplus x^3 \in \mathcal{X}$. Let us fix $x^1, x^2, x^3 \in \mathcal{X}$, and let $x = x^1 \oplus x^2 \oplus x^3$. To show that $x \in \mathcal{X}$ we have to show that $G(x,x) \subseteq G_1$. Namely, that for every $y \succeq x$, we have $y \in G_1$. Let $y \succeq x$. Then there exist y^1, y^2, y^3 such that $y = y^1 \oplus y^2 \oplus y^3$, where $y^j \succeq x_j$ for $j = 1...3$. (To verify this, choose a coordinate i: (1) If $y_i = x_i$: set $y^j_i = x^j_i$ for all j. (2) If $y_i = 1$ and $x_i = 0$: Set $y^j_i = 1$ for all j.) That is, $y^j \in G(x^j, x^j) \subseteq G_1$. Therefore $y^j \in G_1$ for all j, and so $y = y^1 \oplus y^2 \oplus y^3 \in G_1$.

By Corollary 14, since $\mathcal{X}$ is an affine subspace of G_1, either $\mathcal{X} = G_1$, or $|\mathcal{X}| \leq \frac{1}{2}|G_1|$. If $\mathcal{X} = G_1$, then for any $x \in G_1$ we have $G(x,x) = \{y : \; y \succeq x\} \subseteq G_1$, namely G_1 is monotone. By Claim 16, g is a k-monomial, which contradicts our assumptions. Therefore, $|\mathcal{X}| \leq \frac{1}{2}|G_1|$. ■

In the next claim we show that for every $x \in G_1 \setminus \mathcal{X}$, the function g is far from constant on $G(x,z)$, for many $z \in 2^x$. Observe that this is trivially true if g is a monotone monomial, since in this case the set $G_1 \setminus \mathcal{X}$ is empty.

Claim 20 *For every $x \in G_1 \setminus \mathcal{X}$, and for any fixed function $h : \{0,1\}^n \to \{0,1\}$,*

$$\frac{1}{2^{|x|}} \cdot \sum_{z \in 2^x} \Pr_{y \in G(x,z)}[g(y) \neq h(z)] \geq 2^{-k} .$$

Proof: Let us fix $x \in G_1 \setminus \mathcal{X}$ and a function h. For every $z \in 2^x$, if $h(z) = 0$ then $\Pr_{y \in G(x,z)}[g(y) \neq h(z)] = \frac{|G(x,z) \cap G_1|}{|G(x,z)|}$ and if $h(z) = 1$ then $\Pr_{y \in G(x,z)}[g(y) \neq h(z)] = \frac{|G(x,z) \setminus G_1|}{|G(x,z)|} = 1 - \frac{|G(x,z) \cap G_1|}{|G(x,z)|}$. Hence,

$$\Pr_{y \in G(x,z)}[g(y) \neq h(z)] \geq \min\left\{\frac{|G(x,z) \cap G_1|}{|G(x,z)|}, 1 - \frac{|G(x,z) \cap G_1|}{|G(x,z)|}\right\} \tag{6}$$

But, for all $z \in 2^x$, $G(x,z) \not\subseteq G_1$ (otherwise, by Claim 18, we would have $G(x,x) \subseteq G_1$, and so $x \in \mathcal{X}$). Thus, by Claim 13, $\frac{|G(x,z) \cap G_1|}{|G(x,z)|} \leq \frac{1}{2}$. Combining this with Equation (6),

$$\begin{aligned} \frac{1}{2^{|x|}} \cdot \sum_{z \in 2^x} \Pr_{y \in G(x,z)}[g(y) \neq h(z)] \quad &\geq \quad \frac{1}{2^{|x|}} \cdot \sum_{z \in 2^x} \frac{|G(x,z) \cap G_1|}{|G(x,z)|} \\ &= \quad 2^{-n} \cdot \sum_{z \in 2^x} |G(x,z) \cap G_1| \;=\; 2^{-n} \cdot |G_1| \;=\; 2^{-k} \end{aligned}$$

In the last sequence of steps we have used the following: (1) $|G(x,z)| = 2^{n-|x|}$ for every z (Claim 17); (2) For every x, the subsets $G(x,z)$ form a partition of $\{0,1\}^n$ (Claim 17); (3) G_1 is of size 2^{n-k}. ■

If the closure-under-intersection test in Step 3 of Algorithm 2 was performed on g and not on f, we would be done. Indeed, Claims 19 and 20 imply that if g is not a k-monomial then $\Pr_{x \in G_1, y \in \{0,1\}^n}[g(y) \neq g(x \wedge y)] \geq 2^{-(k+1)}$. Therefore, uniformly picking $x \in G_1, y \in \{0,1\}^n$ and checking that $g(y) = g(x \wedge y)$, we would detect a violation with probability at least $2^{-(k+1)}$.

However, the test is performed on f, and g and f might differ (though the distance between them is bounded). Consequently, we need to relate between two different probabilities. This is done in the following claim.

Claim 21 $\Pr_{x \in F_1, y}\left(f(y) \neq f(x \wedge y)\right] \geq \frac{1}{2}\left(\Pr_{x \in F_1 \cap G_1, y}\left[g(y) \neq f(x \wedge y)\right] - 2^{-k-3}\right)$.

Proof: Recall that $\mathrm{dist}(f,g) \leq 2^{-k-3}$, and that $|F_1| \geq 2^{n-k} - 2^{n-k-3}$. Thus:

$$\begin{aligned} \Pr_{x \in F_1, y}\left[f(y) \neq f(x \wedge y)\right] \quad &\geq \quad \Pr_{x \in F_1, y}\left[f(y) \neq f(x \wedge y),\ x \in G_1\right] \\ &= \quad \Pr_{x \in F_1 \cap G_1, y}\left[f(y) \neq f(x \wedge y)\right] \cdot \frac{|F_1 \cap G_1|}{|F_1|} \\ &> \quad \frac{1}{2} \cdot \Pr_{x \in F_1 \cap G_1, y}\left[f(y) \neq f(x \wedge y)\right]. \end{aligned}$$

Also,

$$\begin{aligned} \Pr_{x \in F_1 \cap G_1, y}\left[f(y) \neq f(x \wedge y)\right] \quad &\geq \quad \Pr_{x \in F_1 \cap G_1, y}\left[f(y) \neq f(x \wedge y),\ f(y) = g(y)\right] \\ &= \quad \Pr_{x \in F_1 \cap G_1, y}\left[g(y) \neq f(x \wedge y),\ f(y) = g(y)\right]. \end{aligned}$$

On the other hand,

$$\begin{aligned}&\Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y)\right]\\ &\quad\leq\quad \Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y),\ f(y)=g(y)\right]+\Pr_{y}\left[f(y)\neq g(y)\right]\\ &\quad\leq\quad \Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y),\ f(y)=g(y)\right]+2^{-k-3}.\end{aligned}$$

Hence,

$$\begin{aligned}\Pr_{x\in F_1,y}\left[f(y)\neq f(x\wedge y)\right] &> \frac{1}{2}\Pr_{x\in F_1\cap G_1,y}\left[f(y)\neq f(x\wedge y)\right]\\ &\geq \frac{1}{2}\left(\Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y)\right]-2^{-k-3}\right).\end{aligned}$$

■

We are now ready to prove Lemma 8.

Proof of Lemma 8: By Claim 21, it suffices to bound the following probability:

$$\Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y)\right] \;=\; \frac{1}{|F_1\cap G_1|}\sum_{x\in F_1\cap G_1}\frac{1}{2^{|x|}}\sum_{z\in 2^x}\Pr_{y\in G(x,z)}\left[g(y)\neq f(z)\right]. \tag{7}$$

Let $\mathcal{X}$ be the set defined in Equation (5). If we replace the summation over $x\in F_1\cap G_1$ in Equation (7) with a summation over $x\in(F_1\cap G_1)\setminus\mathcal{X}$, then, by Claim 20, we obtain

$$\begin{aligned}\Pr_{x\in F_1\cap G_1,y}\left[g(y)\neq f(x\wedge y)\right] &\geq \frac{1}{|F_1\cap G_1|}\sum_{x\in(F_1\cap G_1)\setminus\mathcal{X}}2^{-k}\\ &= \frac{|(F_1\cap G_1)\setminus\mathcal{X}|}{|F_1\cap G_1|}\cdot 2^{-k}.\end{aligned}\tag{8}$$

However, by Claim 19, $|\mathcal{X}|\leq\frac{|G_1|}{2}=2^{n-k-1}$. Also $\mathcal{X}\subseteq G_1$ and $\Pr[f\neq g]=\frac{1}{2^n}\left(|F_1\setminus G_1|+|G_1\setminus F_1|\right)\leq 2^{-k-3}$. Hence,

$$|(F_1\cap G_1)\setminus\mathcal{X}|=|(G_1\setminus\mathcal{X})\setminus(G_1\setminus F_1)|\geq 2^{n-k-1}-2^{-k-3}\cdot 2^n=2^{n-k-1}-2^{n-k-3}.$$

Since $|F_1\cap G_1|\leq|G_1|=2^{n-k}$ we have

$$\frac{|(F_1\cap G_1)\setminus\mathcal{X}|}{|F_1\cap G_1|}\geq\frac{2^{n-k-1}-2^{n-k-3}}{2^{n-k}}=\frac{3}{8}. \tag{9}$$

Combining Equation (8) with Equation (9):

$$\Pr_{x\in F_1\cap G_1,y}(g(y)\neq f(x\wedge y))\geq 3\cdot 2^{-k-3}.$$

Thus, by Claim 21:

$$\begin{aligned}\Pr_{x\in F_1,y}\left[f(y)\neq f(x\wedge y)\right] &\geq \frac{1}{2}\left[\Pr_{x\in F_1\cap G_1,y}\left(g(y)\neq f(x\wedge y)\right]-2^{-k-3}\right)\\ &\geq 2^{-k-3},\end{aligned}$$

and we are done. ■

4.3 Testing Monomials when k is Unspecified

Suppose that we want to test whether a function f is a monomial without the size of the monomial, k, being specified. In this case we start by finding k. We obtain an estimate α to $\Pr[f = 1]$, by taking a sample of size $\Theta(1/\epsilon)$. By the multiplicative Chernoff bound, such a sample ensures that, with high probability, if $\Pr[f = 1] \geq \epsilon/2$ then $\alpha \geq \epsilon/4$, while if $\Pr[f = 1] < \epsilon/8$, then $\alpha < \epsilon/4$. Hence, if $\alpha < \epsilon/4$ then we can immediately accept. This is true, since we may assume that $\Pr[f = 1] < \epsilon/2$, and so f is close to every monomial that contains at least $\log(2/\epsilon)$ literals.

Otherwise, we may assume that $\Pr[f = 1] \geq \epsilon/8$, and the multiplicative Chernoff bound implies that, with high probability, $(1 - 1/4) \cdot \Pr[f = 1] < \alpha < (1 + 1/4) \cdot \Pr[f = 1]$. Now, we look for an integer k for which $4/5\alpha \leq 2^{-k} \leq 4/3\alpha$. If there is no such integer, we reject. If there is, there is at most one, and we chose it as our estimate for k. If f is in fact a monomial, then this estimate of k is correct with high probability. Given this k, we proceed as before.

5 Testing Monotone DNF Formulae

In this section we describe an algorithm for testing whether a function f is a monotone DNF formula with at most ℓ terms, for a given integer ℓ.

In other words, we test whether $f = T_1 \vee T_2 \vee \cdots \vee T_{\ell'}$, where $\ell' \leq \ell$, and each term T_i is of the form $T_i = x_{j_1} \wedge x_{j_2} \wedge \cdots \wedge x_{j_{k(i)}}$. Note that we allow the size of the terms to vary. We assume, without loss of generality, that no term contains the set of variables of any other term (or else we can ignore the more specific term), though the same variable can of course appear in several terms. The basic idea underlying the algorithm is to test whether the set $F_1 \stackrel{\rm def}{=} \{x : f(x) = 1\}$ can be "approximately covered" by at most ℓ terms (monomials). To this end, the algorithm finds strings $x^i \in \{0,1\}^n$ and uses them to define functions f^i that are tested for being monomials. If the original function f is in fact an ℓ-term DNF, then, with high probability, each such function f^i corresponds to one of the terms of f.

The following notation will be useful. Let f be a monotone ℓ-term DNF, and let its terms be $T_1, \ldots, T_\ell$. Then, for any $x \in \{0,1\}^n$, we let $S(x) \subseteq \{1, \ldots, \ell\}$ denote the subset of indices of the terms satisfied by x. That is:

$$S(x) \stackrel{\rm def}{=} \{i : \; T_i(x) = 1\} \; .$$

In particular, if $f(x) = 0$ then $S(x) = \emptyset$. This notion extends to a set $R \subseteq F_1$, were $S(R) \stackrel{\rm def}{=} \bigcup_{x \in R} S(x)$. We observe that if f is a monotone ℓ-term DNF, then for every $x, y \in \{0,1\}^n$

$$S(x \wedge y) = S(x) \cap S(y) \quad .$$

We shall also need the following definitions.

Definition 10 (Single-Term Representatives) *Let f be a monotone ℓ-term DNF. We say that $x \in F_1$ is a* single-term representative *for f if $|S(x)| = 1$. That is, x satisfies only a single term in f.*

Definition 11 (Neighbors) *Let $x \in F_1$. The* set of neighbors of *x, denoted $N(x)$, is defined as follows:*

$$N(x) \stackrel{\rm def}{=} \{y \mid f(y) = 1 \textit{ and } f(x \wedge y) = 1\} \quad .$$

The notion of neighbors extends to a set $R \subseteq F_1$, where $N(R) \stackrel{\rm def}{=} \bigcup_{x \in R} N(x)$.

Consider the case in which x is a single-term representative of f, and $S(x) = \{i\}$. Then, for every neighbor $y \in N(x)$, we must have $i \in S(y)$ (or else $S(x \wedge y)$ would be empty, implying that $f(x \wedge y) = 0$). Notice that the converse statement holds as well, that is, $i \in S(y)$ implies that x and y are neighbors. Therefore, the set of neighbors of x is exactly the set of all strings satisfying the term T_i. The goal of the algorithm will be to find at most ℓ such single-term representatives $x \in \{0,1\}^n$, and for each such x to test that its set of neighbors $N(x)$ satisfies some common term. We shall show that if f is in fact a monotone ℓ-term DNF, then all these tests pass with high probability. On the other hand, if all the tests pass with high probability, then f is close to some monotone ℓ-term DNF.

We start with a high-level description of the algorithm, and then show how to implement its main step of finding single-term representatives.

Algorithm 3 Test for Monotone ℓ-term DNF

1. $R \leftarrow \emptyset$. R is designated to be a set of single-term representatives for f.

2. *For $i = 1$ to $\ell + 1$ (Try to add ℓ single-term representatives to R):*

   (a) *Take a uniform sample U^i of size $m_1 = \Theta\left(\frac{\ell \log \ell}{\epsilon}\right)$ strings. Let $W^i = (U^i \cap F_1) \setminus N(R)$. That is, W^i consists of strings x in the sample such that $f(x) = 1$, and x is not a neighbor of any string already in R.*

   Observe that if the strings in R are in fact single-term representatives, then every $x \in W^i$ satisfies only terms not satisfied by the representatives in R.

   (b) *If $i = \ell + 1$ and $W^i \neq \emptyset$, then* reject.

   If there are more than ℓ single term representatives for f then necessarily f is not an ℓ-term DNF.

   (c) *Else, if $\frac{|W^i|}{m_1} < \frac{\epsilon}{4}$ then* go to Step 3.

   The current set of representatives already "covers" almost all of F_1.

   (d) *Else $\left(\frac{|W^i|}{m_1} \geq \frac{\epsilon}{4} \text{ and } i \leq \ell\right)$, use W^i in order to find a string x^i that is designated to be a single-term representative of a term not yet represented in R. This step will be described subsequently.*

3. *For each string $x^i \in R$, let the function $f^i : \{0,1\}^n \mapsto \{0,1\}$ be defined as follows: $f^i(y) = 1$ if and only if $y \in N(x^i)$.*

   As observed previously, if x^i is in fact a single-term representative, then f^i is a monomial.

4. *For each f^i, test that it is monomial, using distance parameter $\epsilon' = \frac{\epsilon}{2\ell}$ and confidence $1 - \frac{1}{6\ell}$ (instead of $\frac{2}{3}$ — this can simply be done by $O(\log \ell)$ repeated applications of each test).*

   Note that we do not specify the size of the monomial, and so we need to apply the appropriate variant of our test, as described in Subsection 4.3.

5. *If any of the tests fail then* reject*, otherwise* accept.

The heart of the algorithm lies in finding a new representative in each iteration of Step 2. This procedure will be described and analyzed shortly. In particular, we shall prove the following lemma.

Lemma 22 *Suppose that f is an ℓ-term monotone DNF, and let $R \subset \{0,1\}^n$ be a subset of single-term representatives for f such that* $\Pr[x \in F_1 \setminus N(R)] \geq \epsilon/8$. *Let U^i be a uniformly selected sample of $m_1 = \Theta\left(\frac{\ell \log \ell}{\epsilon}\right)$ strings, and let $W^i = (U^i \cap F_1) \setminus N(R)$. Then there exists a procedure that receives W^i as input, for which the following holds:*

1. *With probability at least $1-\frac{1}{6\ell}$, taken over the choice of U^i and the internal coin flips of the procedure, the procedure returns a string x^i that is a single term representative for f of a term not yet represented in R. That is, $|S(x^i)| = 1$ and $S(x^i) \cap S(R) = \emptyset$.*

2. *The query complexity of the procedure is $O(\ell \log^2 \ell/\epsilon)$.*

Conditioned on the above lemma we can prove the following theorem.

Theorem 3 *Algorithm 3 is a testing algorithm for ℓ-term DNF. The query complexity of the algorithm is $\tilde{O}(\ell^4/\epsilon^3)$.*

Proof: We shall use the following notation: for any set $R \subset \{0,1\}^n$, let $\bar{p}(R) \stackrel{\text{def}}{=} \Pr[x \in F_1 \setminus N(R)]$.

Suppose f is a monotone ℓ-term DNF, and consider each iteration of Step 2. By Lemma 22, if all strings in R are single-term representatives for f and $\bar{p}(R) \geq \epsilon/8$, then with probability at least $1 - \frac{1}{6\ell}$, the procedure for finding a single term representative in fact returns a new representative (of a term not yet represented in R). Hence, the probability that, for some iteration i, the string x^i returned by the procedure is not a single-term representative, is at most $1/6$. Conditioned on such an event not occurring, the algorithm completes Step 2 with a set R that contains at most ℓ single-term representatives for f.

In such a case, by the definition of single-term representatives, each f^i defined in Step 3 is a (monotone) monomial. For each fixed f^i, the probability that it fails the monomial test is at most $\frac{1}{6\ell}$. By applying a union bound, the probability that any one of the f^i's fail, is at most $\frac{1}{6}$. Adding up the error probabilities, we obtain that f is accepted with probability at least $2/3$.

We now turn to the case in which f is ϵ-far from being a monotone ℓ term DNF. Consider the value of $\bar{p}(R)$ at the start of each iteration i of Step 2. Observe that $\bar{p}(R)$ does not increase with i. If $\bar{p}(R) > \epsilon/2$, then, by the multiplicative Chernoff bound, the probability that $\frac{|W^i|}{m_1} \leq \epsilon/4$ (causing the algorithm to exit Step 2) is smaller than $\frac{1}{6\ell}$. Hence, the probability that the algorithm completes Step 2 without rejecting and with a set R for which $\bar{p}(R) > \epsilon/2$, is at most $1/6$.

Conditioned on such an event not occurring, consider the functions f^i defined in Step 3. We claim that at least one of these functions is $\frac{\epsilon}{2\ell}$-far from being a monomial. To verify this, assume in contradiction that all these $|R| \leq \ell$ functions are $\frac{\epsilon}{2\ell}$-close to being monomials. For each such function, let g^i be a closest monomial and let $g = g^1 \vee g^2 \vee \ldots \vee g^{|R|}$. Then $\text{dist}(f, g) \leq |R| \cdot \frac{\epsilon}{2\ell} + \bar{p}(R) \leq \epsilon$, contradicting the fact that f is ϵ-far from any ℓ-term DNF. Thus, let f^t be one of the f^i's that is $\frac{\epsilon}{2\ell}$-far from being a monomial. The probability that the the monomial test does not reject f^t is at most $\frac{1}{6\ell}$. Adding up the error probabilities, f is rejected with probability at least $2/3$.

Finally, we bound the query complexity of the algorithm. There are at most $\ell + 1$ iterations in Step 2. In each iteration, $m_1 = O(\ell \log \ell/\epsilon)$ strings are queried in Step 2a. By Lemma 22, $O(\ell \log^2 \ell/\epsilon)$ strings are queried by the procedure for finding a new representative that is called in Step 2d. By Theorem 2, testing each of the at most ℓ functions f^i requires $\tilde{O}(1/(\epsilon')^3) \cdot O(\log \ell) = \tilde{O}(\ell^3/\epsilon^3)$ queries. Therefore, the total number of queries is $\tilde{O}(\ell^4/\epsilon^3)$. ■

5.1 Finding New Representatives

Suppose that f is a monotone ℓ-term DNF, and consider an arbitrary iteration i in Step 2 of the algorithm. Assume that $R \subset \{0,1\}^n$ is a subset of single-term representatives for f, such that $\Pr[x \in F_1 \setminus N(R)] \geq \epsilon/8$. Let $\overline{N}(R) \stackrel{\text{def}}{=} F_1 \setminus N(R)$ be the set of all the strings that are not neighbors of any string in R, and let $\overline{S}(R) \stackrel{\text{def}}{=} \{1, \ldots, \ell\} \setminus S(R)$ be the set of indices of terms not yet represented in R. By definition, $W^i \subseteq \overline{N}(R)$, and for every $x \in W^i$ we have $S(x) \subseteq \overline{S}(R)$.

Given a string $x_0 \in W^i$, we shall try to "remove" terms from $S(x_0)$, until we are left with a single term. More precisely, we produce a sequence of strings $x_0, \ldots, x_r$, where $x_0 \in W^i$, such that $\emptyset \neq S(x_{j+1}) \subseteq S(x_j)$, and in particular $|S(x_r)| = 1$. The aim is to decrease the size of $S(x_j)$ by a constant factor for most j's. This will ensure that for $r = \Theta(\log \ell)$, the final string x_r is a single-term representative as desired. How is such a sequence obtained? Given a string $y_j \in N(x_j)$, define $x_{j+1} = x_j \wedge y_j$. Then $f(x_{j+1}) = 1$ (i.e., $S(x_{j+1}) \neq \emptyset$), and $S(x_{j+1}) = S(x_j) \cap S(y_j) \subseteq S(x_j)$. The string y_j is acquired by uniformly selecting a sufficiently large sample from $\{0,1\}^n$, and picking the first string in the sample that belongs to $N(x_j)$, if such exists. The exact procedure follows.

Procedure for finding a new representative, given $W^i \subseteq \overline{N}(R)$

1. *Let the strings in W^i be denoted $w_1, \ldots, w_{|W^i|}$.*
2. *Uniformly and independently select $r = \Theta(\log \ell)$ samples, $Y_0, \cdots, Y_{r-1}$, each consisting of $m_2 = O(\ell \log \ell / \epsilon)$ strings from $\{0,1\}^n$.*
3. *$found \leftarrow FALSE$; $t \leftarrow 0$;*
4. *While $found \neq TRUE$ and $t < |W^i|$ do:*
   (a) *$t \leftarrow t + 1$; $x_0 \leftarrow w_t$.*
   (b) *For $j = 1$ to r*
      i. *If $Y_{j-1} \cap N(x_{j-1}) = \emptyset$ then exit the "for" loop and go to (4a).*
      ii. *Otherwise, pick the first string $y_{j-1} \in Y_{j-1} \cap N(x_{j-1})$, and let $x_j = x_{j-1} \wedge y_{j-1}$.*
   (c) *If $j = r$ then $found \leftarrow TRUE$.*
5. *if $found = TRUE$, return x_r, else return an arbitrary string.*

We first prove that if Y_j intersects $N(x_j)$, the probability that the size of $S(x_{j+1})$ is significantly smaller than that of $S(x_j)$ is bounded away from 0. Observe that since the sample Y_j is uniformly distributed in $\{0,1\}^n$, $Y_j \cap N(x_j)$ is uniformly distributed in $N(x_j)$.

Claim 23 *Let x_j be a fixed string. With probability at least $\frac{1}{3}$ over the uniform choice of a string $y_j \in N(x_j)$, if $x_{j+1} = x_j \wedge y_j$ then $|S(x_{j+1})| \leq 1 + \frac{3}{4} \cdot (|S(x_j)| - 1)$.*

Proof: Without loss of generality, let $S(x_j) = \{1, \ldots, t\}$. We partition the set of neighbors $N(x_j)$ into disjoint subsets $N_i(x_j)$, for $1 \leq i \leq t$, where $N_i(x_j) = \{y : \; i \in S(y) \text{ and for every } i' < i, \; i' \notin S(y)\}$. Since y_j is uniformly distributed in $N(x_j)$, we can view it as being selected by first choosing i with probability $\frac{|N_i(x_j)|}{N(x_j)}$, and then selecting y uniformly in $N_i(x_j)$.

Consider the case $y_j \in N_1(x_j)$. In order to select a string uniformly in $N_1(x_j)$, we first set to 1 all bits corresponding to the variables in T_1, and then let the remaining bits to be 0 or 1 with equal probability. Since for every $i \neq 1$ there is at least one variable that appears in T_i and not in T_1, we have that

$$\Pr\left[T_i(y_j) = 0 \mid y_j \in N_1(x_j)\right] \geq \frac{1}{2}.$$

It follows that the expected number of indices $i \in S(x_j)$, $i \neq 1$, for which $T_i(y_j) = 1$ is at most $\frac{t-1}{2}$. By Markov's inequality, the probability that there are more than $(1-\alpha)(t-1)$ terms T_i, $i \neq 1$, satisfied by a uniformly selected $y_j \in N_1(x_j)$, is at most $\frac{1}{2(1-\alpha)}$. Setting $\alpha = 1/4$, we get that, with probability at least $\frac{1}{3}$ over the choice of a uniformly selected $y_j \in N_1(x_j)$, we have $|S(x_{j+1})| \leq 1 + \frac{3}{4} \cdot (|S(x_j)| - 1)$. It is easy to see that for any $N_i(x_j)$, $i > 1$, this probability is at least as large. In particular, note that for $i = t$, for any $y_j \in N_t(x_j)$, $|S(x_{j+1})| = 1$. ■

The next corollary follows directly from Claim 23 and the fact that $|S(x_0)| \leq \ell$.

Corollary 24 *Let $r = c \cdot \log \ell$, where c is a sufficiently large constant, and let x_0 be a fixed string in W^i. Consider the following process, consisting of r steps, where in the j's step we uniformly and independently select a string $y_{j-1} \in N(x_{j-1})$ and set $x_j = x_{j-1} \wedge y_{j-1}$. Then, with probability at least $1 - \text{poly}(1/\ell)$ over the choice of $y_0, \ldots, y_{r-1}$, we obtain $|S(x_r)| = 1$.*

Finally, we bound the size of a sample Y_j sufficient for acquiring a string $y_j \in N(x_j)$ with high probability. We first define a "good initial string" x_0. This is a string that satisfies only "large" monomials.

Definition 12 *A string x_0 will be called a* good initial string *if for every $i \in S(x_0)$, $\Pr[T_i = 1] \geq \frac{\epsilon}{16\ell}$. Let* $\text{Good} \stackrel{\text{def}}{=} \left\{x \in \overline{N}(R) \text{ and } x \text{ is a good initial string}\right\}$.

Claim 25 *Suppose $\Pr[x \in \overline{N}(R)] \geq \frac{\epsilon}{8}$. Then the probability, taken over the choices of U^i, that W^i does not contain any good initial strings, is at most $\frac{1}{18\ell}$.*

Proof: Recall that $\bar{p}(R) \stackrel{\text{def}}{=} \Pr[x \in \overline{N}(R)]$. For any $i \in \overline{S}(R)$, consider the event

$$E_i \stackrel{\text{def}}{=} \left\{x \in \overline{N}(R) \text{ and } T_i(x) = 1\right\}.$$

By definition, $\bar{p}(R) = \Pr\left[\bigcup_{i \in \overline{S}(R)} E_i\right]$. Let

$$\overline{S}_{\text{small}}(R) = \left\{i \in \overline{S}(R) \text{ and } \Pr[E_i] \leq \frac{\bar{p}(R)}{2\ell}\right\}.$$

Clearly¡ for any term i, $\Pr[T_i = 1] \geq \Pr[E_i]$. Therefore, if $x \in \left(\bigcup_{i \in \overline{S}(R)} E_i\right) \setminus \left(\bigcup_{i \in \overline{S}_{\text{small}}(R)} E_i\right)$ then $S(x) \subseteq \overline{S}(R) \setminus \overline{S}_{\text{small}}(R)$, and therefore for all $i \in S(x)$ we have $\Pr[T_i(x) = 1] \geq \Pr[E_i] \geq \frac{\bar{p}(R)}{2\ell} \geq \frac{\epsilon}{16\ell}$. Thus, $x \in \text{Good}$. Therefore,

$$\begin{aligned}
\Pr[\text{Good}] &\geq \Pr\left[\left(\bigcup_{i \in \overline{S}(R)} E_i\right) \setminus \left(\bigcup_{i \in \overline{S}_{\text{small}}(R)} E_i\right)\right] \\
&\geq \Pr\left[\bigcup_{i \in \overline{S}(R)} E_i\right] - \Pr\left[\bigcup_{i \in \overline{S}_{\text{small}}(R)} E_i\right] \\
&\geq \bar{p}(R) - \ell \cdot \frac{\bar{p}(R)}{2\ell} \quad = \quad \frac{\bar{p}(R)}{2}.
\end{aligned}$$

Since $\bar{p}(R) \geq \frac{\epsilon}{8}$, and the size of the sample U^i is $\Theta(\ell \log \ell/\epsilon)$, the probability that W^i does not contain *any* initial good strings is, for a sufficiently large constant in the $\Theta(\cdot)$ notation, smaller than $\frac{1}{18\ell}$. ■

The next claim follows from the definition of a good initial string.

Claim 26 *Let $m_2 = c \cdot \ell \log \ell/\epsilon$, where c is a sufficiently large constant, and suppose x_0 is a good initial string. Then, for each $1 \leq j \leq r$, the probability that a sample Y_j of m_2 strings intersects $N(x_j)$ is at least $1 - \frac{1}{18\ell^2}$.*

Proof of Lemma 22: By the premise of the lemma, $\Pr[x \in \overline{N}(R)] \geq \epsilon/8$. By Claim 25, the set W^i contains a good initial string with probability at least $1 - \frac{1}{18\ell}$. Conditioned on this event, let us fix such a string x_0, and consider the execution of Step 4b in the procedure. By Claim 26, the probability that there exists $j \leq r$ for which the sample Y_j does not contain a string in $N(x_j)$ is at most $\frac{1}{18\ell}$. Since the strings in Y_j are uniformly selected from $\{0,1\}^n$, the strings in $Y_j \cap N(x_j)$ are uniformly distributed in $N(x_j)$. Hence, conditioned on each Y_j containing a string from $N(x_j)$, we can apply Corollary 24 and get that with probability at least $1 - \frac{1}{18\ell}$, $|S(x_r)| = 1$. Since $x_0 \in \overline{N}(R)$, necessarily $x_r \in \overline{N}(R)$. Therefore, with probability at least $1 - 3 \cdot \frac{1}{18\ell} = 1 - \frac{1}{6\ell}$, taken over the choices of U^i and the samples Y_j, the procedure returns a string x_r that is a single-term representative for f of a term not yet represented in f.
The number of queries performed is $r \cdot m_2 = O(\ell \log^2 \ell/\epsilon)$. ■

6 Testing Singletons Without Testing Linearity

Recall that by Claim 1 an alternative characterization of singletons is that $Pr[f = 1] = 1/2$, and furthermore that there are no violating pairs $x, y \in \{0,1\}^n$. That is, there are no x, y such that $f(x \wedge y) \neq f(x) \wedge f(y)$. We show that the following simple algorithm that checks these properties, is a testing algorithm for singletons if f is not too far from a singleton function. Let $\mathcal{F}_{\text{SING}}$ denote the class of singletons. The algorithm will receive a value γ_0 such that $\min_{g \in \mathcal{F}_{\text{SING}}} \text{dist}(f, g) \leq \frac{1}{2} - \gamma_0$. That is, γ_0 is a lower bound on the difference between $1/2$ and the distance of f to the closest singleton. We shall think of γ_0 as a constant.

Algorithm 4 Test for Singleton with lower bound γ_0

1. Size Test: *Uniformly select a sample of $m = \Theta(1/\epsilon^2)$ strings in $\{0,1\}^n$. For each x in the sample, obtain $f(x)$. Let α be the fraction of sample strings x such that $f(x) = 1$. If $|\alpha - 1/2| > \frac{\epsilon}{4}$ then* reject*, otherwise continue.*
2. Closure-Under-Intersection Test: *Repeat the following $\Theta(\epsilon^{-1}\gamma_0^{-1})$ times: Uniformly select $x, y \in \{0,1\}^n$. If x and y are a violating pair, then reject.*
3. *If no step caused rejection, then accept.*

Theorem 4 *If f is a singleton, then Algorithm 4 accepts with probability at least $2/3$. If f is ϵ-far from any singleton where ϵ is bounded away from $1/2$, then the algorithm rejects with probability at least $2/3$. The query complexity of the algorithm is $O(1/\epsilon^2)$.*

Proof: If f is a singleton then $\Pr[f = 1] = 1/2$. By an additive Chernoff bound, and for the appropriate constant in the $\Theta(\cdot)$ notation, the probability that it is rejected in the first step of Algorithm 4 is less than $1/3$. By the definition of singletons, f always passes the closure-under-intersection test.

Suppose that f is ϵ-far from any singleton and let δ be its distance to the closest singleton. Thus $\epsilon < \delta \leq 1/2 - \gamma_0$. We show that f is rejected with probability greater than $2/3$.

1. If $|\Pr[f = 1] - 1/2| > \frac{\epsilon}{2}$, then f is rejected in the first step of the algorithm with probability at least $5/6$.

2. Otherwise, $|\Pr[f = 1] - 1/2| \leq \frac{\epsilon}{2} < \frac{\delta}{2}$. In this case, as we show shortly in Lemma 27, the probability of obtaining a violating pair is at least $\frac{\delta}{4}(\frac{1}{2} - \delta) \geq \frac{\epsilon}{4} \cdot \gamma_0$. Therefore, f will be rejected with probability of at least $5/6$ in the second step of the algorithm (the closure-under-intersection test).

Thus, the probability that f is accepted by the algorithm is at most $1/3$, as required. ■

Lemma 27 *Let δ be the distance of f to the closest singleton. If $\Pr[f(x) = 1] \geq \frac{1}{2} - \frac{\delta}{2}$, then the probability of obtaining a violating pair is at least $\frac{\delta}{4}(\frac{1}{2} - \delta)$.*

Proof: Let x_i be the closest singleton to f, so that $\Pr[f(x) \neq x_i] = \delta$. Define

$$G_1 = \{x | f(x) = 1, x_i = 1\}, \quad B_1 = F_1 \setminus G_1$$

$$G_0 = \{x | f(x) = 0, x_i = 0\}, \quad B_0 = F_0 \setminus G_0$$

A simple counting argument shows that there are $(\frac{1}{2} - \delta)2^n$ disjoint pairs x, x', such that: (1) $x \in G_1$, $x' \in G_0$; (2) x and x' differ only on the i'th bit. To see why this is true, simply match each $x \in G_1$ to a point x', which differs with x only on the i'th bit. Thus, there are at least $|G_1| - |B_1|$ points $x \in G_1$ that must be matched to points $x' \in G_0$. But $|G_1| + |B_0| = 2^{n-1}$, and $|B_1| + |B_0| = \delta 2^n$ and therefore $|G_1| - |B_1| = (\frac{1}{2} - \delta)2^n$.

Now consider any point $y \in B_1$, and let $x \in G_1$, $x' \in G_0$ be a matched pair as defined above. Then $x \wedge y = x' \wedge y$, but $f(x) \wedge f(y) = 1$ while $f(x') \wedge f(y) = 0$. Therefore, either $f(x \wedge y) \neq f(x) \wedge f(y)$ or $f(x' \wedge y) \neq f(x') \wedge f(y)$, and so either y and x are a violating pair, or y and x' are a violating pair.

Since $\Pr[f(x) = 1] \geq \frac{1}{2} - \frac{\delta}{2}$, then $|G_1| + |B_1| \geq 2^n(\frac{1}{2} - \frac{\delta}{2})$. Using again the fact that $|G_1| - |B_1| = (\frac{1}{2} - \delta)2^n$, we get that $|B_1| \geq \delta 2^{n-2}$. It follows that the probability of obtaining a violating pair, is at least $\frac{\delta}{4}(\frac{1}{2} - \delta)$. ■

The above analysis breaks when f is actually almost $1/2 - far$ from every singleton, since in this case δ is close to $1/2$, and the probability $\frac{\delta}{4}(\frac{1}{2} - \delta)$ of obtaining a violating pair is not bounded from below. Another disadvantage of Algorithm 4 is the two sided error probability for testing singletons, as opposed to the one sided error we achieved in Algorithm 1 when we added the parity test. However, Algorithm 4 can be generalized to testing k-monomials, with a query complexity of only $O(1/\epsilon^2)$, in comparison to the $\tilde{O}(1/\epsilon^3)$ query complexity of Algorithm 2. The probability of choosing a violating pair can be shown to be at least $\frac{\delta}{4}(\frac{1}{2^k} - \delta)$. Thus the requirement here is that δ will be strictly smaller than $\frac{1}{2^k}$. Notice that it is not a problem that δ is even more restricted here, since we first must test whether $\Pr[f(x) = 1]$ is approximately $\frac{1}{2^k}$.

Another alternative test for singletons is to replace the relatively expensive test of checking whether $\Pr[f(x) = 1]$ is approximately $1/2$, by extending the notion of a violating pair. We will say that $x, y \in \{0,1\}^n$ are a violating pair if $f(x \wedge y) \neq f(x) \wedge f(y)$ or if $f(x \vee y) \neq f(x) \vee f(y)$. Then in a similar way to the proof of Lemma 27, it can be shown that the probability of obtaining a violating pair is at least $\frac{\delta}{2}(\frac{1}{2} - \delta)$ (In this case the size of either B_0 or B_1 is at least $\delta 2^{n-1}$. Therefore choosing $y \in B_1$ or $y \in B_0$ and x, x' as before, will result in a violating pair either to the $\wedge$ test or to the $\vee$ test). The query complexity of this algorithm will be only $O(1/\epsilon)$, and it will have a one-sided error. Unfortunately this algorithm does not extend to testing monomials.

Further Research

Our results raise several questions that we believe may be interesting to study.

- Our algorithms for testing singletons and, more generally, monomials, apply two tests. The role of the first test is essentially to facilitate the analysis of the second, natural test (the closure under intersection test). The question is whether the first test is necessary.
- Our algorithm for testing monomials has a cubic dependence on $1/\epsilon$, as opposed to the linear dependence of the singleton testing algorithm. Can this dependence be improved?
- The query complexity of our algorithm for testing ℓ-term DNF grows like ℓ^4. While some dependence on ℓ seems necessary, we conjecture that a lower dependence is achievable. In particular, suppose we slightly relax the requirements of the testing algorithm and only ask that it rejects functions that are ϵ-far from any monotone DNF with at most $c \cdot \ell$ (or possibly ℓ^c) terms, for some constant c. Is it possible, under this relaxation, to devise an algorithm that has only polylogarithmic dependence on ℓ?
- Finally, can our algorithm for testing monotone DNF functions be extended to testing general DNF functions?

References

[AFKS99] N. Alon, E. Fischer, M. Krivelevich, and M Szegedy. Efficient testing of large graphs. In *Proceedings of FOCS*, pages 645–655, 1999.

[AHRS99] Y. Aumann, J. Håstad, M. Rabin, and M. Sudan. Linear consistency testing. In *Proceedings of RANDOM*, pages 109–120, 1999.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[BCH+95] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of FOCS*, pages 432–441, 1995.

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, April 1987.

[BJT99] N. Bshouty, J. Jackson, and C. Tamon. More efficient PAC-learning of DNF with membership queries under the uniform distribution. In *Proceedings of COLT*, pages 286–295, 1999.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *JACM*, 47:549–595, 1993.

[DGL+99] 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.

[GGL+00] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

[GGR98] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998.

[GR97] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of STOC*, pages 406–415, 1997. To appear in *Algorithmica*.

[Jac97] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *JCSS*, 55:414–440, 1997.

[KS99] A. Klivans and R. Servedio. Boosting and hard-core sets. In *Proceedings of FOCS*, pages 624–633, 1999.

[Ron00] D. Ron. Property testing. To appear in the Handbook on Randomization. Currently available from: `http://www.eng.tau.ac.il/~danar`, 2000.

[RS96] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[Rub99] R. Rubinfeld. Robust functional equations and their applications to program testing. *SIAM Journal on Computing*, 28(6):1972–1997, 1999.