Monotonicity Testing

by

Sofya Raskhodnikova

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1999

Author . . . .
Department of Electrical Engineering and Computer Science
May 26, 1999

Certified by . . . .
Michael Sipser
Professor of Mathematics
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Monotonicity Testing

by

Sofya Raskhodnikova

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

We present improved algorithms for testing monotonicity of functions. Namely, given the ability to query an unknown function $f : \Sigma^n \mapsto \Xi$, where Σ and Ξ are finite ordered sets, the test always accepts a monotone f and rejects f with high probability if it is ϵ-far from being monotone (i.e., if every monotone function differs from f on more than an ϵ fraction of the domain). For any $\epsilon > 0$, the query complexity of the test is $O((n/\epsilon) \cdot \log|\Sigma| \cdot \log|\Xi|)$. The previous best known bound was $\tilde{O}((n^2/\epsilon) \cdot |\Sigma|^2 \cdot |\Xi|)$.

We also present an alternative test for the boolean range $\Xi = \{0, 1\}$ whose query complexity $O(n^2/\epsilon^2)$ is independent of alphabet size $|\Sigma|$.

Thesis Supervisor: Michael Sipser
Title: Professor of Mathematics

Acknowledgments

I am very grateful to my advisor, Professor Michael Sipser, for indispensable discussions, advice, patience, and support. This work was done in collaboration with Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Dana Ron, and Alex Samorodnitsky. I am especially thankful to Eric Lehman who proposed this problem to me. My sincere gratitude goes to Yevgeniy and Eric for many fruitful discussions and sleepless nights. I am happy to acknowledge Michael Krivilevich and Madhu Sudan whose suggestions are incorporated into this thesis. In particular, Proposition 12 is due to Michael Krivilevich. I cannot find words to thank Boris enough for his support and help in all aspects of my life.

This work was supported by NSF grant 9503322CCR.

Contents

Chapter 1

Introduction

1.1 Gap Property Testing

Gap Property Testing[1] (cf., [13, 9]) is a general formulation of computational tasks in which one is to determine whether a given object has a particular property or is “far” from any object having the property. By “far” we mean that one needs to modify the object at many places to obtain an object with the property. Thus, we exclude from consideration input objects which are borderline, that is, which lack the property but are close to others having it. Allowing arbitrary errors on borderline instances enables us to design much more efficient tests.

Typically, the goal is to perform gap property testing within complexity smaller than the size of the object. If a testing algorithm received an object as its input, just reading the description of the object would take more time then testing. Instead, to avoid that problem, the algorithm is given oracle (black box) access to a natural encoding of the object as a function. The complexity of the algorithm is measured in terms of the number of queries to the oracle.

Since algorithms employed in gap property testing might not have time to read the entire input, it is natural to allow for errors. Therefore, gap property testers are usually probabilistic. Thus, a gap property tester has to determine with high probability whether a function, specified by the oracle, belongs to some class or is “far” from this class.

A primary advantage of gap property testing algorithms is that their complexity can be

[1] Also referred to as probabilistic property testing, property testing, and spot checking.

exponentially smaller than that of exact decision procedures. In applications where some errors are tolerable gap property testers can save a lot of time. They are also useful as preprocessors for procedures that require input with particular properties. In addition, they can be used in program testing to check if the output of the program is close to having required properties (e.g. a list returned by a sorting program is close to being sorted [6]).

Much work in this area was devoted to testing algebraic properties of functions such as linearity (e.g., [5, 1, 4, 3]) and low-degree properties (e.g., [5, 7, 13, 12, 2]). Recently, some attention was given to testing combinatorial properties of functions; first, for functions representing graphs [9, 10, 11], and more recently for functions per se [6, 8]. A natural combinatorial property of functions is monotonicity, and in this thesis we focus on testing monotonicity.

1.2 Formulation of Our Problem

We consider functions with a totally ordered range Ξ, which are defined on n-symbol strings over a totally ordered alphabet Σ. There is a natural partial order $\prec$ defined on the strings in the domain: $x_1x_2\cdots x_n \prec y_1y_2\cdots y_n$ if $x_i \leq y_i$ for every i and $x_i \neq y_i$ for some i, i.e. $x \prec y$ if y can be obtained from x by increasing one or more alphabet symbols.

Definition 1 *A function $f : \Sigma^n \mapsto \Xi$ is* monotone *if $f(x) \leq f(y)$ holds for every $x \prec y$.*

That is, a function is monotone if increasing an input symbol does not decrease the output. The distance of a given function from monotone is measured by the fraction of points in the domain on which it has to be modified to become monotone.

Definition 2 *The* relative distance *of a function $f : \Sigma^n \mapsto \Xi$ from the set of monotone functions, denoted $\epsilon_{\mathrm{M}}(f)$, is the minimum over all monotone $g : \Sigma^n \mapsto \Xi$ of*

$$\frac{|\{x \in \Sigma^n : f(x) \neq g(x)\}|}{|\Sigma|^n}.$$

*A function f is ϵ-*far from monotone *if $\epsilon_{\mathrm{M}}(f) > \epsilon$.*

Our goal is to find a gap property tester for monotonicity of functions.

Definition 3 *A probabilistic oracle machine M is said to be a* tester of monotonicity *if*

$$\begin{array}{llll} \text{Prob}[M^f(\epsilon, n, |\Sigma|, |\Xi|) = 1] & \geq \frac{2}{3} & & \text{for any monotone function } f, \quad \text{AND} \\ \text{Prob}[M^f(\epsilon, n, |\Sigma|, |\Xi|) = 0] & \geq \frac{2}{3} & & \text{for } f \text{ which is } \epsilon\text{-far from monotone.} \end{array}$$

Monotonicity of functions in the context of gap property testing was first considered by Goldreich, Goldwasser, Lehman and Ron [8]. Their main result is a tester of monotonicity for the case $\Sigma = \Xi = \{0,1\}$ having query and time complexities of the form $\text{poly}(n)/\epsilon$. Specifically, the analysis of the query complexity in [8] yields a bound of $\tilde{O}(n^2/\epsilon)$, where $\tilde{O}()$ is the same as $O()$ with suppressed polylogarithmic factors.

The authors also show that $\Omega(n/\epsilon)$ is a lower bound on the query complexity of their algorithm. For general Σ and Ξ, the bounds obtained in [8] are proportional to $|\Sigma|^2 \cdot |\Xi|$.

1.3 Overview of our Results

The main technical contribution of this work is improving both the algorithm and the analysis in [8] to obtain the following.

Theorem 1 (main result): *The monotonicity tester in this work has query complexity*

$$q(n, \epsilon, |\Sigma|, |\Xi|) \stackrel{\text{def}}{=} O\left(\frac{n \cdot (\log |\Sigma|) \cdot (\log |\Xi|)}{\epsilon}\right).$$

The tester works by selecting independently $q(n,\epsilon)/2$ pairs of n-symbol strings over Σ, and comparing the two f-values obtained for the elements of each pair.[2]

Thus, the global feature of being monotone or far from it is determined by a sequence of many independent random local checks. Each local check consists of selecting a pair, (x, y), so that (without loss of generality) $x \prec y$, according to some fixed distribution and checking whether $f(x) \leq f(y)$. If the algorithm ever finds a pair for which this does not hold (i.e., local violation of monotonicity), then it rejects. Otherwise, the algorithm accepts. Thus, it never rejects a monotone function. The challenge is to analyze the dependence of rejection probability on the distance of the given function from being monotone.

[2]Since the algorithm is comparison-based, its complexity depends only on the size of the image of the function. Thus, one may replace Ξ in the above bound by $\Xi_f = \{f(x) : x \in \Sigma^n\}$. In particular, $\log |\Xi_f| \leq n \cdot \log |\Sigma|$; so, by Theorem 1, $q(n, \epsilon)$ is never worse than $O(n^2 \cdot (\log |\Sigma|)^2/\epsilon)$.

The only thing left unspecified in the above description of the testing algorithm is the distribution by which the pairs are selected. In case $\Sigma = \{0,1\}$ there seems to be a very natural choice: Uniformly select dimension $i \in \{1,...,n\}$, independently and uniformly select $z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n \in \{0,1\}$, and set $x = z_1 \cdots z_{i-1} 0 z_{i+1} \cdots z_n$ and $y = z_1 \cdots z_{i-1} 1 z_{i+1} \cdots z_n$. Our improvement over [8] in the case where $\Sigma = \Xi = \{0,1\}$ comes from a better (and in fact tight for $\Xi = \{0,1\}$) analysis of this test, which allows us to obtain the following theorem.

Theorem 2 ($\Sigma = \Xi = \{0,1\}$): *If (x,y) is drawn according to the distribution specified above, then*

$$\text{Prob}[f(x) > f(y)] \geq \frac{2\epsilon_{\text{M}}(f)}{n}.$$

In case of non-binary $\Sigma = \{1,...,d\}$ there seem to be several natural possibilities for the distribution by which the pairs are chosen: Even if we restrict ourselves (as above) to select only pairs of strings which differ on a single coordinate i, there is still the question of how to select the corresponding pair of symbols. We study several natural possibilities of randomly selecting pairs $(k,\ell) \in \Sigma \times \Sigma$.

1. Select uniformly a pair $(k, k+1)$ with $k \in \{1,...,d-1\}$.
2. Select uniformly a pair (k,ℓ) with $k < \ell$.
3. Select uniformly a pair (k,ℓ) from a particular subset of all pairs of size $O(d \log d)$.

A key result of this work is the reduction of the analysis of testing algorithms as above for any n and Σ, and for $\Xi = \{0,1\}$, to their behavior in the special case of $n = 1$ (where we simply select pairs (k,ℓ) according to one of the above distributions and check the order between $f(k)$ and $f(\ell)$). Using this reduction we derive the following theorem.

Theorem 3 (Monotonicity Testing of Boolean Functions): *Efficiently samplable distributions on pairs $(x,y) \in \Sigma^n \times \Sigma^n$ can be constructed so that for every function $f : \Sigma^n \mapsto \{0,1\}$ the following holds:*

1. *If (x,y) is drawn according to one distribution (which is determined by the third distribution on pairs in Σ^2 mentioned above), then*

$$\text{Prob}[f(x) > f(y)] = \Omega\left(\frac{\epsilon_{\text{M}}(f)}{n \cdot (\log|\Sigma|)}\right).$$

2. If (x,y) is drawn according to the other distribution (which is determined by the second distribution on pairs in Σ^2 mentioned above), then

$$\mathrm{Prob}[f(x) > f(y)] \ = \ \Omega\left(\frac{\epsilon_{\mathrm{M}}(f)^2}{n^2}\right).$$

We note that the first part of the theorem can also be derived by applying our reduction and using an alternative distribution on pairs in Σ^2 which was previously suggested and analyzed for the case $n = 1$ in [6].

The reader may be tempted to say that since our algorithm is comparison-based, the analysis should also hold for non-boolean functions. However, this is not so. For example, by part (2) above, boolean functions over Σ may be tested for monotonicity within complexity independent of $|\Sigma|$. In contrast, a lower bound in [6] asserts that arbitrary functions over Σ (e.g., with $\Xi = \Sigma$) cannot be tested for monotonicity within complexity independent of $|\Sigma|$ but rather require complexity $\Omega(\log|\Sigma|)$ for some fixed distance parameter $\epsilon > 0$. Thus, a natural question arises: *Under what conditions and at what cost can results regarding testing of monotonicity of boolean functions be transformed to results for testing monotonicity of arbitrary functions*? Our most general result is the following.

Theorem 4 (Monotonicity Testing – Range Reduction): *Consider the task of testing monotonicity of functions defined over any partially ordered set S (with partial order $\prec_S$). Suppose that for some distribution on pairs $(x,y) \in S \times S$ with $x \prec_S y$ and for every function $f : S \mapsto \{0,1\}$,*

$$\mathrm{Prob}[f(x) > f(y)] \ \geq \ \frac{\epsilon_{\mathrm{M}}(f)}{C} \ ,$$

where C depends on S only. Then, for every Ξ and every function $f : S \mapsto \Xi$, for pairs selected according to the same distribution,

$$\mathrm{Prob}[f(x) > f(y)] \ \geq \ \frac{\epsilon_{\mathrm{M}}(f)}{C \cdot \log_2|\Xi|} \ .$$

Theorem 1 follows by combining part (1) of Theorem 3 and Theorem 4 with $C = O(n \cdot \log|\Sigma|)$.

1.4 Organization

We start with some preliminaries in chapter 2. Chapter 3 analyzes the performance of our algorithm for functions with binary alphabet Σ and range Ξ. This chapter is intended to give a smooth introduction of the techniques generalized in chapter 5 and to explain additional techniques that we were not able to generalize. Chapter 3 is not required for understanding the rest of the thesis. The algorithms for the case $n = 1$ (each corresponding to a different distribution on pairs in $\Sigma \times \Sigma$) are presented in chapter 4. In chapter 5 we show how the analysis of our algorithm in the binary-range case for arbitrary n and Σ reduces to the case $n = 1$ and prove Theorem 3. A general reduction from an arbitrary range to the binary range and the proof of Theorem 4 appear in chapter 6. Chapters $4 - 6$ can be read independently, even though they reference each other to give a high-level picture. The final chapter presents related open problems.

Chapter 2

Preliminaries

This chapter introduces notation used in the thesis and formally defines our monotonicity test. As described in the introduction, we consider functions $f : \Sigma^n \mapsto \Xi$, where Σ and Ξ are totally ordered sets. We denote the natural partial order on equal-length strings over Σ by $\prec$.

For any pair of functions $f, g : \Sigma^n \mapsto \Xi$, we define the distance between f and g, denoted $\mathrm{dist}(f, g)$, to be the fraction of instance strings in the domain on which $f(x) \neq g(x)$. As in the introduction, we let $\epsilon_{\mathrm{M}}(f)$ denote the minimum distance between f and any monotone function g with the same domain and range. Let us formally define the algorithmic schema studied in this thesis. The schema uses an arbitrary probability distribution $p : \Sigma \times \Sigma \mapsto [0, 1]$. Without loss of generality, we assume that the support of p is restricted to pairs (k, ℓ) with $k < \ell$. The function t, referred to below, depends on p.

Algorithmic schema: Given parameters ϵ, n, Σ, Ξ, and oracle access to an arbitrary function $f : \Sigma^n \mapsto \Xi$, repeat the following local test up to $t(\epsilon, n, |\Sigma|, |\Xi|)$ times:

1. Uniformly select dimension $i \in [n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$, prefix $\alpha \in \Sigma^{i-1}$, and suffix $\beta \in \Sigma^{n-i}$.

2. Select (k, ℓ) according to distribution p. Let $x = \alpha\, k\, \beta$, $y = \alpha\, \ell\, \beta$.

3. If $f(x) > f(y)$ (i.e., the pair (x, y) witnesses that f is not monotone), then reject.

If all iterations were completed without rejecting, then accept.

We focus on the analysis of a single iteration of the above local test. Such an iteration is fully specified by the distribution, denoted $D_p^n : \Sigma^n \times \Sigma^n \mapsto [0, 1]$, by which pairs (x, y)

are selected. That is, $D_p^n(x,y) = \frac{p(k,\ell)}{n\cdot|\Sigma|^{n-1}}$ if $x = \alpha\, k\, \beta$ and $y = \alpha\, \ell\, \beta$, for some α, β, and $D_p^n(x,y) = 0$ otherwise. Observe that $D_p^n(x,y) > 0$ only if $x \prec y$. Let $\textsc{Detect}(f, D_p^n)$ be the probability that a pair (x,y) selected according to D_p^n witnesses that f is not monotone; that is,

$$\textsc{Detect}(f, D_p^n) \stackrel{\text{def}}{=} \text{Prob}_{(x,y)\sim D_p^n}[f(x) > f(y)] \tag{2.1}$$

(where the above definition can of course be applied to any distribution D on pairs $x \prec y$). Our goal is to find distributions D_p^n (determined by the distributions p) for which $\textsc{Detect}(f, D_p^n)$ is "well" lower-bounded as a function of $\epsilon_{\text{M}}(f)$. If D_p^n is such that *for any* $f : \Sigma^n \mapsto \Xi$ with $\epsilon_{\text{M}}(f) \geq \epsilon$ we have that $\textsc{Detect}(f, D_p^n) \geq \delta(\epsilon, n, |\Sigma|, |\Xi|)$, then setting $t(\epsilon, n, |\Sigma|, |\Xi|) = \Theta(1/\delta(\epsilon, n, |\Sigma|, |\Xi|))$ yields a tester for monotonicity.

Thus, in the analysis of our distributions D_p^n and corresponding tests, our goal is to find a good lower bound on $\textsc{Detect}(f, D_p^n)$ in terms of $\epsilon_{\text{M}}(f)$, or equivalently, a good upper bound on $\epsilon_{\text{M}}(f)$ in terms of $\textsc{Detect}(f, D_p^n)$. Chapters 3, 5, and 6 present particular methods for transforming a function f into a monotone function and upper bound $\epsilon_{\text{M}}(f)$ by the distance from f to that function.

The partial order graph: It will be convenient to view the partial order over Σ^n as a directed (acyclic) graph, denoted G_Σ^n. The vertices of G_Σ^n are the strings in Σ^n and directed edges correspond to comparable pairs (i.e. (x,y) is an edge iff $x \prec y$). An edge (x,y) is said to be violated by f if $f(x) > f(y)$. We denote by $\textsc{Viol}(f)$ the set of violated edges of f. We remark that most of the definitions in this section naturally extend to any partially ordered set S in place of Σ^n.

Chapter 3

Binary Alphabet and Range

This chapter covers monotonicity testing results for the special case when the alphabet Σ and the range Ξ are binary. This case is interesting in its own right, and we are going to use it to demonstrate techniques generalized in chapter 5 for treating functions on strings over general alphabets. For the case of the binary alphabet and range, we were able to obtain absolutely tight (as opposed to asymptotically tight) analysis of our test. Even though we were not able to generalize the technique that allowed us to get a tight constant factor in the analysis, we think the technique can potentially be applied elsewhere.

Consider a subgraph of the partial order graph where each vertex is a point in the domain (a binary string of length n) and an edge from x to y represents that $x \prec y$ and x differs from y in exactly one coordinate. This graph is an n-dimensional hypercube, and that is how we are going to refer to it. In this chapter, references to violated edges of function f are references to violated edges of the corresponding hypercube.

Notice that our local test for this case simplifies to uniformly selecting an edge from the hypercube and checking if it is violated. As before, the challenge is to analyze the probability that one iteration of the local test finds a violated edge. As discussed in the preliminaries, to analyze that probability, we show a particular method of transforming a function f into a monotone function and upper bound $\epsilon_{\mathrm{M}}(f)$ by the distance from f to that function. This method allows us to alter f on at most one point in the domain for each repaired violated edge. Section 3.1 defines and analyzes the core idea of this method, the shifting operator. Section 3.2 improves this operator, studies the properties of the new shifting operator, and applies it to the analysis of our algorithmic schema to prove Theorem 2.

3.1 Shifting Operator S_i

If a hypercube does not contain violated edges, the corresponding function is monotone. The main idea of the analysis is to transform a given boolean function to a monotone function by repairing violated edges in one dimension at a time. To repair violated edges in dimension i for $1 \leq i \leq n$, we use the shifting operator S_i, defined below, which swaps values of f on endpoints of violated edges in dimension i and leaves f the same everywhere else.

Definition 4 *Let $f : \{0,1\}^n \to \{0,1\}$, $x \in \{0,1\}^n$, y be obtained from x by flipping the i'th bit, and $x \prec y$. Then*

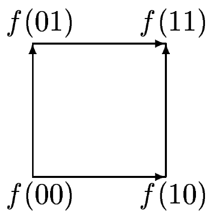
$$\begin{array}{llll} S_i[f](x) = f(y) = 0 & \text{and} & S_i[f](y) = f(x) = 1 & \text{if } f(x) > f(y), \\ S_i[f](x) = f(x) & \text{and} & S_i[f](y) = f(y) & \text{otherwise.} \end{array}$$

For $i \in [n]$, we call a function f monotone in dimension i if there are no violated edges in that dimension. By definition, for each $i \in [n]$, $S_i[f]$ is monotone in dimension i. Let $V_i(f)$ denote the number of violated edges of f in dimension i, and $V(f)$, the total number of violated edges of f. In the next lemma we derive a simple property of the shifting operators.

Lemma 5 *For every $1 \leq i \neq j \leq n$ and every function $f : \{0,1\}^n \mapsto \{0,1\}$, applying the shifting operator S_i does not increase the number of violated edges in dimension j, i.e., $V_j(S_i[f]) \leq V_j(f)$.*

Proof: The important observation is that the statement of the lemma is concerned with only two dimensions, i and j. Ignoring edges in all other dimensions, we can view the hypercube as 2^{n-2} disconnected squares. Each of these squares represents a two-dimensional boolean function, obtained from f by fixing all coordinates other than i and j. To prove the lemma, it is enough to show that it holds for two-dimensional functions. It will demonstrate that the shifting operator S_i does not increase the number of violated edges in dimension j in any of the squares, and hence does not increase the total number of violated edges in dimension j.

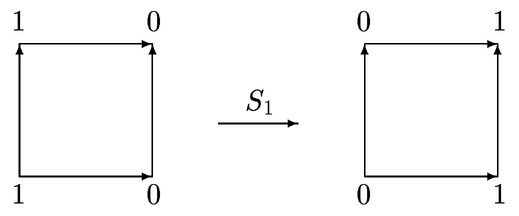
We prove the lemma for two-dimensional functions by case analysis. We depict a two-dimensional function f as shown. A directed edge between $f(x)$ and $f(y)$ indicates that x and y differ in exactly one coordinate and $x \prec y$.



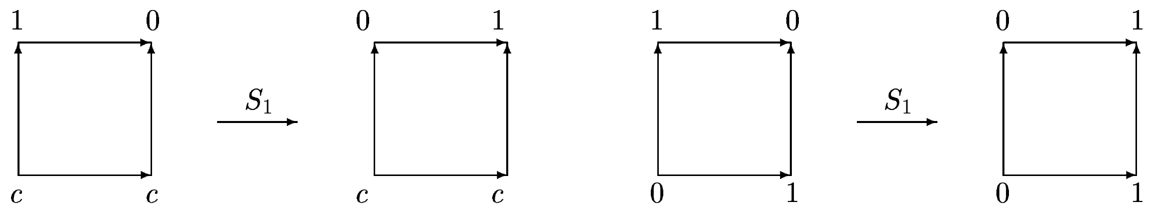
Without loss of generality, let $i = 1$ and $j = 2$. Then the shifting operator swaps edges in the horizontal dimension, and the goal is to prove that the number of violated edges in the vertical dimension does not increase.

Case 1: f does not have violated edges in the horizontal dimension. Then $f \equiv S_i[f]$, and hence $V_j(S_i[f]) = V_j(f)$.

Case 2: Both horizontal edges are violated in f. Then the vertical edges are swapped, and $V_j(S_i[f]) = V_j(f)$.



Case 3: The upper edge is violated; the lower one is not. Consider two possibilities: $f(00) = f(10)$ and $f(00) \neq f(10)$. If $f(00) = f(10)$, then the vertical edges are swapped, and $V_j(S_i[f]) = V_j(f)$. If not, then since the lower edge is not violated, $f(00) = 0$ and $f(10) = 1$. In this case, the vertical violated edge is repaired.



Case 4: The lower edge is violated; the upper one is not. This case is symmetrical to Case 3. ■

Corollary 6 *For every $i \in [n]$, applying the* shifting *operator S_i to any function $f : \{0,1\}^n \mapsto \{0,1\}$ does not increase the number of violated edges in dimensions other than i:*

$$\sum_{j \in [n] \setminus \{i\}} V_j(S_i[f]) \leq \sum_{j \in [n] \setminus \{i\}} V_j(f). \qquad ■$$

3.2 Improved Shifting Operator S_i'

Consider two hypercubes into which the original hypercube is decomposed when all edges in dimension i are removed. We say that all points with $x_i = 0$ and edges connecting them form the left sub-cube, and all points with $x_i = 1$ and edges connecting them form the right cub-cube. Edges in dimension i start at nodes of the left sub-cube and end at nodes of the right sub-cube. Therefore, applying the shifting operator S_i can only change the values of f in two ways:

1. from one to zero on points in the left sub-cube,

2. from zero to one on points in the right sub-cube.

We know that changes of types (1) and (2) together do not increase the total number of violated edges in dimensions other than i. Therefore, since changes (1) and (2) affect only the sub-cube they are made in, either (1) or (2) does not increase the total number of violated edges in dimensions other than i.

To be more precise, we define left and right shifting operators, L_i and R_i, which make the same changes in f as S_i, but restricted to the left and right sub-cubes, respectively. Then we define a new shifting operator S_i', which is equal either to L_i or R_i, depending on the number of violated edges $L_i[f]$ and $R_i[f]$ have.

Definition 5

$$\text{Let } L_i[f](x) = \begin{cases} S_i[f](x) & \text{if } x_i = 0, \\ f(x) & \text{otherwise.} \end{cases} \qquad \text{Let } R_i[f](x) = \begin{cases} S_i[f](x) & \text{if } x_i = 1, \\ f(x) & \text{otherwise.} \end{cases}$$

$$\text{Let } S_i'[f] \equiv \begin{cases} L_i[f] & \text{if } V(L_i[f]) \leq V(R_i[f]), \\ R_i[f] & \text{otherwise.} \end{cases}$$

Lemma 7 (Properties of S_i') *For any $i \in [n]$, the following properties of the new* shifting *operator S_i' hold:*

1. *$S_i'[f]$ is monotone in dimension i.*

2. *Applying the* shifting *operator S_i' to any function f does not increase the number of*

violated edges in the dimensions other than i:

$$\sum_{j\in[n]\setminus\{i\}} V_j(S'_i[f]) \leq \sum_{j\in[n]\setminus\{i\}} V_j(f).$$

Proof:

1. Clearly, both L_i and R_i repair all violated edges in dimension i: $V_i(L_i[f]) = 0$ and $V_i(R_i[f]) = 0$. Since, by definition, S'_i is equal to L_i or to R_i, it also repairs all violated edges in dimension i.

2. By previous discussion, either $V(L_i[f]) \leq V(S_i[f])$ or $V(R_i[f]) \leq V(S_i[f]$. By definition, S'_i is equal to the operator that forces f to have fewer violated edges.

■

Set $D_i(f)$ to be the number of changes from f to $S'_i[f]$, namely,

$$D_i(f) = |\{x, f(x) \neq S'_i(x)\}|.$$

Observe, that $D_i(f) = V_i(f)$. Thus, we can repair all violated edges of f by changing f on at most $V(f)$ points, using the following algorithm.

ALGORITHM: Repeat until f is monotone:

1. Pick non-monotone dimension i.
2. Change f to $S'_i[f]$.

This proves the following lemma.

Lemma 8 $\epsilon_{\mathrm{M}}(f) \leq \frac{V(f)}{2^n}$ *for any* $f : \{0,1\}^n \mapsto \{0,1\}$.

■

Notice that the upper bound above is tight. For each natural number n, we can construct $f : \{0,1\}^n \to \{0,1\}$ for which $\epsilon_{\mathrm{M}}(f) = V(f)/2^n$. Namely, let f be 1 on all points whose first coordinate is zero and 0 on all points whose first coordinate is one. Then violated edges of f are edges of the form $(0x, 1x)$ for all $x \in \{0,1\}^{n-1}$, and hence $V(f) = 2^{n-1}$. To make f monotone, we have to change its value on at least one of the two endpoints of each violated edge. Since all violated edges are disjoint, f has to be changed on at least 2^{n-1} points. In fact, f needs to be changed on exactly 2^{n-1} points: for example, making f zero everywhere requires 2^{n-1} changes. Thus, $\epsilon_{\mathrm{M}}(f) = 1/2 = V(f)/2^n$.

The main theorem of this chapter is an easy corollary of the previous lemma. **Proof of Theorem 2:** Let $f : \{0,1\}^n \rightarrow \{0,1\}$ and U be the uniform distribution over the edges of the hypercube. By the previous lemma, $V(f) \geq \epsilon_{\mathrm{M}}(f) \cdot 2^n$. Since there are 2^n vertices and $n2^{n-1}$ edges in an n-dimensional hypercube,

$$\textsc{Detect}(f, U) = \frac{V(f)}{n2^{n-1}} \geq \frac{2\epsilon_{\mathrm{M}}(f)}{n}.$$

■

Chapter 4

Testing Monotonicity on a Line (the $n = 1$ case)

In this chapter, we design algorithms for the case $n = 1$, for any Σ and Ξ. In accordance with our algorithmic schema, the design of such algorithms amounts to the design of a probability distribution $p : \Sigma^2 \mapsto [0, 1]$ (with support only on pairs (k, ℓ) with $k < \ell$). Note that for $n = 1$, we have $D_p^n \equiv p$. As discussed in chapter 2, any lower bound on $\textsc{Detect}(f, p)$, implies an upper bound on the number of times $t(n, \epsilon, |\Sigma|, |\Xi|)$ that the basic iteration of the algorithmic schema should be performed. We present three such distributions, denoted p_1, p_2, and p_3, and provide bounds on $\textsc{Detect}(f, p_j)$, for each j.

4.1 Matching Lemma

The following lemma[1], which relates $\epsilon_{\text{M}}(f)$ to $\textsc{Viol}(f)$, will be used in our analysis of various algorithms. Recall that a *matching* of a graph is a collection of edges that share no common endpoint.

Lemma 9 *For any $f : \Sigma \mapsto \Xi$ the graph $G' = (\Sigma, \textsc{Viol}(f))$ has a matching of size $\epsilon_{\text{M}}(f) \cdot |\Sigma|/2$.*

Proof: Recall that a *vertex cover* of a graph is a subset of vertices such that every edge of the graph has at least one of its endpoints in the subset. We claim that a minimum

[1]While stated for a totally ordered set Σ, both the result and the proof hold for any partially ordered set S.

vertex cover of G' has size at least $\epsilon_{\mathrm{M}}(f) \cdot |\Sigma|$. The lemma directly follows as the size of a maximum matching is at least $1/2$ of the size of the minimum vertex cover. Let $U \subseteq \Sigma$ be any vertex cover of G'. We next show that by modifying the value of f only on points in U, we obtain a monotone function, implying that $|U| \geq \epsilon_{\mathrm{M}}(f) \cdot |\Sigma|$, as claimed.

Let $T = \Sigma \backslash U$. By definition of U, there are no violated edges between pairs of vertices in T. Consider the following iterative process, where in each step we modify the value of f on a single $y \in U$, remove y form U, and add it to T. We maintain the property (which holds initially) that following each step there are no violated edges between vertices in T. The process ends when $U = \emptyset$ and $T = \Sigma^n$, so that the final function is monotone. To redefine the value of f on y, we consider the following two subsets of T: $T_1 = \{x \in T : (x, y) \in \textsc{Viol}(f)\}$ and $T_2 = \{z \in T : (y, z) \in \textsc{Viol}(f)\}$. By transitivity of the partial order, and the fact that there are no violated edges (x, z), for $x, z \in T$, at most one of these subsets is non-empty. If T_1 is non-empty, then we let $f(y) = \max_{x \in T_1}\{f(x)\}$, and if T_2 is non-empty, then $f(y) = \min_{z \in T_2}\{f(y)\}$. In case both are empty (all violated edges incident to y have an end-point in U), the value of y may remain unchanged.

We note that the size of the minimum vertex cover actually equals $\epsilon_{\mathrm{M}}(f) \cdot |\Sigma|$. Consider any set U such that by modifying the value of f only on strings in U we can obtain a monotone function g. Then U must be a vertex cover of G', as otherwise there remain violated edges with respect to g. ■

4.2 Simple Distribution p_1

First we consider a very simple distribution p that only looks at neighboring pairs and show that it gives a too low probability of detecting a violation of monotonicity.

Distribution p_1: This distribution is uniform over pairs $(k, k+1)$. That is,

$$p_1(k, k+1) = 1/(d-1), \text{for } k = 1, \ldots, d-1.$$

Proposition 10 *For any Ξ and $f : \Sigma \mapsto \Xi$,* $\textsc{Detect}(f, p_1) \geq \frac{2}{d-1} \cdot \epsilon_{\mathrm{M}}(f)$.

The lower bound can be shown to be tight even for $\Xi = \{0, 1\}$ (by considering the function f defined by $f(x) = 1$ if $x < d/2$ and $f(x) = 0$ otherwise).

Proof: If $\epsilon_{\rm M}(f) > 0$, then there exists some $k \in \{1, \ldots, d-1\}$ so that $f(k) > f(k+1)$. If there are at least two such k's, then we reject with probability at least $2/(d-1) \geq 2\epsilon_{\rm M}(f)/(d-1)$ as $\epsilon_{\rm M}(f) \leq 1$. Otherwise, there is a unique k that causes us to reject. In this case, $\epsilon_{\rm M}(f) \leq 1/2$ since we can change either all $f(i)$ to $f(k+1)$ for $i \leq k$ or all $f(i)$ to $f(k)$ for $i > k$ in order to make f monotone. Thus, we reject with probability $1/(d-1) \geq 2\epsilon_{\rm M}(f)/(d-1)$ in this case as well. ■

4.3 Optimal Distribution p_2

We see that the above test is too "short-sighted" since it only looks at the neighboring pairs of vertices. We now design a distribution that spots the violated edges much better. As noted before, an alternative distribution which meets the same bound was previously suggested and analyzed in [6].

Distribution p_2: This distribution is uniform on a set $P \subset \Sigma \times \Sigma$ which is defined as follows. The set P consists of pairs (k, ℓ), where $0 < \ell - k \leq 2^t$ and 2^t is the largest power of 2 which divides either k or ℓ. That is, let $\mathsf{power}_2(i) \in \{0, 1..., \log_2 i\}$ denote the largest power of 2 which divides i. Then,

$$P \stackrel{\rm def}{=} \{(k, \ell) \in \Sigma \times \Sigma : 0 < \ell - k \leq 2^{\max(\mathsf{power}_2(k), \mathsf{power}_2(\ell))}\} \tag{4.1}$$

and $p_2(k, \ell) = \frac{1}{|P|}$ for every $(k, \ell) \in P$, and is 0 otherwise.

Proposition 11 *For any Ξ and $f : \Sigma \mapsto \Xi$,* Detect$(f, p_2) \geq \frac{1}{O(\log d)} \cdot \epsilon_{\rm M}(f)$.

Proof: We first show that $|P| = O(d \log d)$. This can be shown by charging each pair $(k, \ell) \in P$ to the element divisible by the larger power of 2 (i.e., to k if $\mathsf{power}_2(k) > \mathsf{power}_2(\ell)$ and to ℓ otherwise), and noting that the charge incurred on each element i is at most $2 \cdot 2^{\mathsf{power}_2(i)}$. It follows that the total charge is at most $\sum_{i=1}^{d} 2^{\mathsf{power}_2(i)+1} = \sum_{j=0}^{\log_2 d} \frac{d}{2^j} \cdot 2^{j+1} = O(d \log d)$.

Since p_2 is uniform over P, the value of Detect(f, p_2) is the ratio between the number of violated edges of f in P and the size of P. Thus, it remains to show that the former is $\Omega(\epsilon_{\rm M}(f) \cdot d)$. In the following argument it will be convenient to view the indices $1, \ldots, d$ as vertices of a graph and the pairs $(k, \ell) \in P$ as directed edges. We refer to this graph as G_P, and note that it is a subgraph if G_Σ^1.

Claim 11.1: For every two vertices k and ℓ in G_P with $k < \ell$, there is a directed path of length at most 2 from k to ℓ in G_P.

Proof of Claim: Let $r = \lceil \log d \rceil$, and consider the binary strings of length r representing k and ℓ. Let $k = (x_{r-1}, \ldots, x_0)$ and $\ell = (y_{r-1}, \ldots, y_0)$. Let j be the highest index such that $x_j = 0$ and $y_j = 1$. Note that $x_i = y_i$ for $j < i < r$. We claim that the vertex $s = (x_{r-1}, \ldots, x_{t+1}, 1, 0, \ldots 0)$ is on a path of length 2 from k to ℓ. This follows from the definition of P, since s is divided by 2^j, while both $s - k = 2^j - \sum_{i=0}^{j-1} x_i 2^i \leq 2^j$ and $\ell - s = \sum_{i=0}^{j-1} y_i 2^i < 2^j$. □

We now apply Lemma 9 to obtain a matching M of size $m \geq (\epsilon_{\rm M}(f) \cdot d/2)$ consisting of violated edges of f. By the above claim, there is path of length at most 2 in G_P between every matched pair. Each edge e of G_P belongs to at most 2 such paths: on at most one path it is the first edge, and on at most one it is the second edge (otherwise, M is not a matching). Since for every $(x, y) \in M$ we have $f(x) > f(y)$ (while $x \prec y$), the length-2 path between x and y must contain a violated edge. Thus, we obtain at least $m/2 \geq (\epsilon_{\rm M}(f) \cdot d/4)$ violated edges in G_P, and the proposition follows. ■

On the Optimality of Distribution p_2. We show that the result of Proposition 11 is optimal (up to a constant factor), even for $\Xi = \{0, 1\}$. The following argument is due to Michael Krivilevich.

Proposition 12 *For any distribution $p : \Sigma \times \Sigma \mapsto [0, 1]$, with support only on pairs (k, ℓ) such that $k < \ell$, there exists a non-monotone $f : \Sigma \mapsto \{0, 1\}$ so that*

$$\textsc{Detect}(f, p) \;\leq\; \frac{2}{\log_2 d} \cdot \epsilon_{\rm M}(f)$$

Proof: Let p be a distribution on pairs as above. We define

$$\rho \stackrel{\rm def}{=} \max_{f:\Sigma\mapsto\{0,1\} \text{ s.t. } \epsilon_{\rm M}(f)>0} \left\{ \frac{\textsc{Detect}(f, p)}{\epsilon_{\rm M}(f)} \right\}$$

Our aim is to show that $\rho \leq 2/\log_2 d$. The key observation is that for any consecutive $2a$ indices, p has to assign a probability mass of at least $\rho \cdot a/d$ to pairs (k, ℓ) where k is among the lowest a indices and ℓ is among the higher a such indices. This observation is proven as follows. Let L, H be the low and high parts of the interval in question; that is, $L = \{s+1, ..., s+a\}$ and $H = \{s+a+1, ..., s+2s\}$, for some $s \in \{0, ..., d-2a\}$. Consider the function f defined by $f(i) = 1$ if $i \in L \cup \{s+2a+1, ..., d\}$ and $f(i) = 0$ otherwise. Then $\epsilon_{\mathrm{M}}(f) = a/d$. On the other hand, the only pairs (k, ℓ) with $f(k) > f(\ell)$ are those satisfying $k \in L$ and $\ell \in H$. Thus, by definition of ρ, it must hold that $\rho \leq \Pr_{(k,\ell)\sim p}[k \in L \ \& \ \ell \in H]/(a/d)$, and the observation follows.

The rest of the argument is quite straightforward: Consider $\log_2 d$ partitions of the interval $[1, d]$, so that the i^{th} partition is into consecutive segments of length 2^i. For each segment in the i^{th} partition, probability p assign a probability mass of at least $2^{i-1}\rho/d$ to pairs where one element is in the low part of the segment and the other element is in the high part. Since these segments are disjoint and their number is $d/2^i$, it follows that p assigns a probability mass of at least $\rho/2$ to pairs among halves of segments in the i^{th} partition. These pairs are disjoint from pairs considered in the other partitions and so we conclude that $(\log_2 d) \cdot \frac{\rho}{2} \leq 1$. The proposition follows. ■

4.4 Distribution p_3 for Boolean Range

We now describe a distribution that works well for the boolean range.

Distribution p_3: This distribution is uniform over all pairs (k, ℓ) such that $k < \ell$. That is, $p_3(k, \ell) = 2/((d-1)d)$ for $1 \leq k < \ell \leq d$.

Proposition 13 *For any* $f : \Sigma \mapsto \{0, 1\}$, $\textsc{Detect}(f, p_3) \geq \epsilon_{\mathrm{M}}(f)^2/2$.

A slightly more careful analysis (which we omit) can extend the bound to $\epsilon_{\mathrm{M}}(f)^2$, which is tight. For any integer $e < d/2$, consider the function $f(x) = 0$ if $x \in \{2, 4, 6...2e\}$ and $f(x) = 1$ otherwise. Then $\epsilon_{\mathrm{M}}(f) = e/d$ and $|\textsc{Viol}(f)| = 1 + \ldots + e \approx e^2/2$. Thus, $\textsc{Detect}(f, p_3) \approx (e^2/2)/(d^2/2) = (e/d)^2 = \epsilon_{\mathrm{M}}(f)^2$.

Proof: Let z be the number of zeroes in f and let $2e$ be the number of mismatches between f and its sorted form. Then $\epsilon_{\mathrm{M}}(f) \leq 2e/d$ as by swapping the $2e$ mismatches we make f monotone. On the other hand, considering the e 1-entries in the z-prefix of f and the e 0-entries in the $(d-z)$-suffix, we lower bound the rejection probability by $e^2/((d-1)d/2) > 2(e/d)^2$. Combining the two, we conclude that $\textsc{Detect}(f, p_3) \geq 2 \cdot (\epsilon_{\mathrm{M}}(f)/2)^2$. ■

We remark that the restriction to boolean range in Proposition 13 is important. For any integer $e \leq d/2$, define $f : \Sigma \mapsto \Sigma$ by $f(2i) = 2i - 1$, $f(2i - 1) = 2i$ for $i = 1 \ldots e$, and $f(i) = i$ for $i > 2e$. Clearly, $\epsilon_{\mathrm{M}}(f) = e/d$, while f has only e violated edges: $(2i - 1, 2i)$, $i = 1 \ldots e$. Thus, $\textsc{Detect}(f, p_3) = e/(d(d-1)/2) = 2\epsilon_{\mathrm{M}}(f)/(d-1)$, which is much less than $\epsilon_{\mathrm{M}}(f)^2$ if e is large.

Chapter 5

Dimension Reduction for Boolean Functions

In this chapter we extend the analysis of the performance of our test for the case $n = 1$ (the "line") to boolean functions $f : \Sigma^n \mapsto \{0, 1\}$. Without loss of generality, assume $\Sigma = \{1, \ldots, d\}$, so $|\Sigma| = d$. Recall that our algorithmic schema is fully specified by a probability distribution $D_p^n : \Sigma^n \times \Sigma^n \mapsto [0, 1]$ by which pairs (x, y) are selected. This distribution, in turn, is determined by a probability distribution p on pairs $(k, \ell) \in \Sigma \times \Sigma$.

The main result of this chapter is Theorem 3 proved in section 5.3. It shows that our monotonicity test performs well with distributions $D_{p_2}^n$ and $D_{p_3}^n$, induced by previously analyzed distributions for the line. For both distributions, the theorem lower bounds $\textsc{Detect}(f, D_p^n)$ in terms of $\epsilon_{\mathrm{M}}(f)$, ensuring that functions far from monotone can be readily detected. The proof of this lower bound is facilitated by section 5.1, where the core idea of the dimension reduction, the sorting operator, is introduced, and section 5.2, where the task of obtaining a lower bound for functions with domain Σ^n is reduced to that of obtaining such a lower bound for functions with domain Σ.

Consider a subgraph of the partial order graph where each vertex is a point in the domain (a string of length n over alphabet Σ) and an edge from x to y represents that $x \prec y$ and x differs from y in exactly one coordinate. Nodes of this graph are vertices of n dimensional hypergrid. For each line along the axes of the hypergrid, there is an edge between each pair of points on the line. These are the only violated edges that can be potentially detected by our algorithmic schema. We call this graph an n-dimensional thick hypergrid or, for $n = 2$, simply a thick grid. In this chapter, references to violated edges of function f are references to violated edges of the corresponding thick hypergrid.

5.1 Sorting Operator

If a thick hypergrid does not contain violated edges, the corresponding function is monotone. As in section 3.1, the main idea of the analysis is to transform a given boolean function to a monotone function by repairing violated edges in one dimension at a time. To repair violated edges in dimension i for $1 \le i \le n$, we use the sorting operator S_i which is a generalization of the shifting operator discussed in Chapter 3.

Before defining the sorting operator, we generalize the terms used in the analysis of functions with binary domain and range. For any $i \in [n]$, we say that a function f is monotone in dimension i if there are no violated edges in that dimension, i.e. if for every $\alpha \in \Sigma^{i-1}$, $\beta \in \Sigma^{n-i}$, and $k, \ell \in \Sigma$ such that $k < \ell$, $f(\alpha\, k\, \beta) \le f(\alpha\, \ell\, \beta)$. For a set of indices $T \subseteq [n]$, we say that f is monotone in dimensions T, if for every $i \in T$, the function f is monotone in dimension i. In what follows we describe sorting operators by which any boolean function over Σ^n can be transformed into a monotone function (as we prove below).

Definition 6 *For every $i \in [n]$, the function $S_i[f] : \Sigma^n \mapsto \{0, 1\}$ is defined as follows: For every $\alpha \in \Sigma^{i-1}$ and every $\beta \in \Sigma^{n-i}$, we let $S_i[f](\alpha\, 1\, \beta), \ldots, S_i[f](\alpha\, d\, \beta)$ be assigned the values of $f(\alpha\, 1\, \beta), \ldots, f(\alpha\, d\, \beta)$, in sorted order.*

By definition, for each $i \in [n]$, the function $S_i[f]$ is monotone in dimension $\{i\}$. For every $i \in [n]$ and every pair $(k, \ell) \in \Sigma^2$ with $k < \ell$, we say that an edge is a (k, ℓ)-edge in ith dimension if the ith coordinates of its endpoints are k and ℓ (all other coordinates are the same for both endpoints). Let $V_{i,(k,\ell)}(f)$ denote the number of violated (k, ℓ)-edges of f in ith dimension. Formally,

$$\begin{aligned} V_{i,(k,\ell)}(f) \;\; = \;\; & |\{(x, y) \in \textsc{Viol}(f) \; : \\ & \exists\, \alpha \in \Sigma^{i-1}\, ,\, \beta \in \Sigma^{n-i} \text{ s.t. } x = \alpha\, k\, \beta\, ,\, y = \alpha\, \ell\, \beta\}|. \end{aligned} \tag{5.1}$$

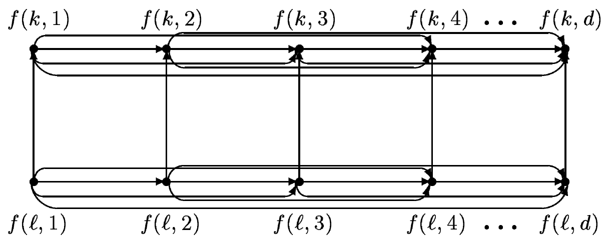
Thus, $\sum_{i,(k,\ell)} V_{i,(k,\ell)}(f)$ is the number of violated edges of f.

Lemma 14 *For every $1 \leq i \neq j \leq n$, function $f : \Sigma^n \mapsto \{0,1\}$, and $1 \leq k < \ell \leq d$, applying the* sorting *operator S_i does not increase the number of violated (k,ℓ)-edges in dimension j, i.e.*

$$V_{j,(k,\ell)}(S_i[f]) \leq V_{j,(k,\ell)}(f).$$

Proof: As in the proof of Lemma 5, the important observation is that the statement of the lemma is concerned with only two dimensions, i and j. Ignoring edges in all other dimensions, we can view the n-dimensional thick hypergrid as many disconnected two-dimensional thick grids. Each of these thick grids represents a two-dimensional boolean function, obtained from f by fixing all coordinates other than i and j. To prove the lemma, it is enough to show that it holds for all two-dimensional functions. It will demonstrate that the sorting operator S_i does not increase the number of violated (k,ℓ)-edges in dimension j in any of the thick grids, and hence does not increase the total number of violated (k,ℓ)-edges in dimension j.

Now we prove the lemma for two-dimensional functions. Without loss of generality, let $i = 1$ and $j = 2$. Then the sorting operator sorts values along the horizontal dimension, and the goal is to prove that the number of violated vertical (k,ℓ)-edges does not increase. The picture shows vertical (k,ℓ)-edges and all horizontal edges between their endpoints.



The current claim amounts to saying that for any $2 \times d$ zero-one matrix

$$\begin{pmatrix} f(k,1) & f(k,2) & \cdots & f(k,d) \\ f(\ell,1) & f(\ell,2) & \cdots & f(\ell,d) \end{pmatrix},$$

sorting the rows of the matrix does not increase the number of unsorted columns.

Let Q be any d-by-2 zero-one matrix, let o_1 (respectively, o_2) denote the number of ones in the first (respectively, second) row of Q, and let Q' be the matrix resulting from sorting the rows of Q. If $o_1 \leq o_2$, then Q' has no unsorted columns and we are done. Otherwise, Q' has exactly $(o_1 - o_2)$ unsorted columns. But out of o_1 columns of Q that have 1 in the first row, at most o_2 can be sorted, so Q should have at least $(o_1 - o_2)$ unsorted columns as well. This proves the lemma. ■

Corollary 15 *If f is monotone in dimensions $T \subseteq [n]$, then $S_i[f]$ is monotone in dimensions $T \cup \{i\}$.*

Proof: By the previous lemma, applying the sorting operator S_i does not increase the number of violated (k, ℓ)-edges in dimension j for all $k < \ell$ in $[n]$ and all j in T. Hence, it does not increase the total number of violated edges in dimensions T. ■

5.2 Dimension Reduction

With Lemma 14 at our disposal, we are ready to state and prove that the analysis of the algorithmic schema (for any n) reduces to its analysis for the special case $n = 1$. Let A denote one iteration of the algorithmic schema, p be any distribution on pairs $(k, \ell) \in \Sigma \times \Sigma$ such that $k < \ell$, and D_p^n be the corresponding distribution induced on edges of the thick hypergrid. The dimension reduction lemma upper bounds $\epsilon_{\mathrm{M}}(f)$ and lower bounds $\textsc{Detect}(f, D_p^n)$ by the corresponding quantities for $n = 1$.

Lemma 16 (Dimension Reduction for Boolean Functions) *Let $f : \Sigma^n \mapsto \{0,1\}$. Then there exist functions $f_{j,\alpha,\beta} : \Sigma \mapsto \{0,1\}$, for $j \in [n]$, $\alpha \in \{0,1\}^{j-1}$ and $\beta \in \{0,1\}^{n-j}$, so that the following holds (all expectations below are taken uniformly over $j \in [n]$, $\alpha \in \{0,1\}^{j-1}$ and $\beta \in \{0,1\}^{n-j}$):*

1. $\epsilon_{\mathrm{M}}(f) \leq 2n \cdot \mathbf{E}_{j,\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta}))$.
2. $\textsc{Detect}(f, D_p^n) \geq \mathbf{E}_{j,\alpha,\beta}(\textsc{Detect}(f_{j,\alpha,\beta},\ p))$.

Proof: For $j = 1, \ldots, n+1$, we define $f_j = S_{j-1} \cdots S_1[f]$ and let $f_1 = f$. By Lemma 14, we have that f_{n+1} is monotone. It follows that

$$\epsilon_{\mathrm{M}}(f) \;\leq\; \mathrm{dist}(f, f_{n+1}) \;\leq\; \sum_{j=1}^{n} \mathrm{dist}(f_j, f_{j+1}). \tag{5.2}$$

Next, for $j = 1, \ldots, n$, $\alpha \in \{0,1\}^{j-1}$ and $\beta \in \{0,1\}^{n-j}$, define the function $f_{j,\alpha,\beta} : \Sigma \mapsto \{0,1\}$, by $f_{j,\alpha,\beta}(x) = f_j(\alpha\, x\, \beta)$, for $x \in \Sigma$. Throughout the proof, $\sum_{\alpha,\beta}$ refers to summing over all (α,β)'s in $\Sigma^{j-1}\times\Sigma^{n-j}$, and $\mathbf{E}_{\alpha,\beta}$ refers to expectation over uniformly distributed $(\alpha,\beta)\in\Sigma^{j-1}\times\Sigma^{n-j}$. We claim that

$$\mathrm{dist}(f_j,f_{j+1}) \;\le\; 2\cdot\mathbf{E}_{\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})). \tag{5.3}$$

This inequality is proven (below) by observing that f_{j+1} is obtained from f_j by sorting, separately, the elements in each $f_{j,\alpha,\beta}$. (The factor of 2 is due to the relationship between the distance of a vector to its sorted form and its distance to monotone.) We have,

$$\begin{aligned}
d^n\cdot\mathrm{dist}(f_j,f_{j+1}) &= \sum_{\alpha,\beta}|\{x\in\Sigma : f_j(\alpha\,x\,\beta)\neq f_{j+1}(\alpha\,x\,\beta)\}| \\
&= \sum_{\alpha,\beta}|\{x\in\Sigma : f_{j,\alpha,\beta}(x)\neq f_{j+1,\alpha,\beta}(x)\}| \\
&= \sum_{\alpha,\beta}|\{x\in\Sigma : f_{j,\alpha,\beta}(x)\neq S_j[f_{j,\alpha,\beta}](x)\}| \\
&\le \sum_{\alpha,\beta}2d\cdot\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta}) \\
&= 2d^n\cdot\mathbf{E}_{\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})),
\end{aligned}$$

where the inequality is justified as follows. Consider a vector $v\in\{0,1\}^d$, and let $S(v)$ denote its sorted version. Then $S(v)=0^z1^{d-z}$, where z denotes the number of zeros in v. Thus, for some $e\ge 0$, the vector v has e 1-entries within its z-prefix and e 0-entries in its $(d-z)$-suffix. Therefore, the number of locations on which v and $S(v)$ disagree is exactly $2e$. On the other hand, consider an arbitrary perfect matching of the e 1-entries in the prefix and the e 0-entries in the suffix. To make v monotone one must alter at least one entry in each matched pair; thus, $\epsilon_{\mathrm{M}}(v)\ge e/d$.

Combining Eq. (5.2) and (5.3), the first part of the lemma follows:

$$\epsilon_{\mathrm{M}}(f) \;\le\; \sum_{j=1}^{n}\mathrm{dist}(f_j,f_{j+1}) \le 2\cdot\sum_{j=1}^{n}\mathbf{E}_{\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})) = 2n\cdot\mathbf{E}_{j,\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})).$$

In order to prove the second part, we use the definition of algorithm A. For any event E, let $\chi(E)=1$ if E holds and $\chi(E)=0$ otherwise.

$$\textsc{Detect}(f,D_p^n)$$

$$= \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{\alpha\,,\beta} \mathrm{Prob}_{(k,\ell)\sim p}[f(\alpha\, k\, \beta) > f(\alpha\, \ell\, \beta)] \tag{5.4}$$

$$= \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{\alpha\,,\beta} \sum_{(k,\ell)} p(k,\ell) \cdot \chi[f(\alpha\, k\, \beta) > f(\alpha\, \ell\, \beta)] \tag{5.5}$$

$$= \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{(k,\ell)} p(k,\ell) \cdot \sum_{\alpha\,,\beta} \chi[f(\alpha\, k\, \beta) > f(\alpha\, \ell\, \beta)] \tag{5.6}$$

$$= \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{(k,\ell)} p(k,\ell) \cdot V_{j,(k,\ell)}(f) \tag{5.7}$$

Using Lemma 14, we have

$$V_{j,(k,\ell)}(f) \;\geq\; V_{j,(k,\ell)}(S_1[f]) \;\geq\; \cdots \;\geq\; V_{j,(k,\ell)}(S_{j-1} \cdots S_1[f]) = V_{j,(k,\ell)}(f_j)$$

Unwinding steps (5.4)-(5.7) with f_j in place of f and recalling the definition of $f_{j,\alpha,\beta}$, we get

$$\begin{aligned}
\textsc{Detect}(f, D_p^n) &\geq \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{(k,\ell)} p(k,\ell) \cdot V_{j,(k,\ell)}(f_j) \\
&= \frac{1}{n \cdot d^{n-1}} \sum_{j=1}^{n} \sum_{\alpha\,,\beta} \mathrm{Prob}_{(k,\ell)\sim p}[f_j(\alpha\, k\, \beta) > f_j(\alpha\, \ell\, \beta)] \\
&= \mathbf{E}_{j,\alpha,\beta}(\textsc{Detect}(f_{j,\alpha,\beta},\ p)),
\end{aligned}$$

and the second part of the lemma follows. ■

5.3 Proof of Theorem 3

In this section we combine Lemma 16 with the results for the case $n = 1$ provided in section 4, and derive Theorem 3 (see section 4 for the definitions of distributions p_2 and p_3).

Combining Lemma 16 and Proposition 11 (applied only to $\Xi = \{0, 1\}$), we have

$$\begin{aligned}
\textsc{Detect}(f, D_{p_2}^n)) &\geq \mathbf{E}_{j,\alpha,\beta}(\textsc{Detect}(f_{j,\alpha,\beta},\ p_2)) && \text{[By Part 2 of the lemma]} \\
&\geq \mathbf{E}_{j,\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})/O(\log d)) && \text{[By the proposition]} \\
&\geq \tfrac{\epsilon_{\mathrm{M}}(f)}{2n \cdot O(\log d)} = \Omega(\tfrac{\epsilon_{\mathrm{M}}(f)}{n \log d}) && \text{[By Part 1 of the lemma]}
\end{aligned}$$

which establishes the first part of the theorem.

Combining Lemma 16 and Proposition 13, we have

$$\begin{aligned}
\textsc{Detect}(f, D^n_{p_3})) &\geq \mathbf{E}_{j,\alpha,\beta}(\textsc{Detect}(f_{j,\alpha,\beta},\ p_3)) && \text{[By Part 2 of the lemma]} \\
&\geq \mathbf{E}_{j,\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta})^2/2) && \text{[By the proposition]} \\
&\geq [\mathbf{E}_{j,\alpha,\beta}(\epsilon_{\mathrm{M}}(f_{j,\alpha,\beta}))]^2/2 && \text{[as } \mathbf{E}(X^2) \geq [\mathbf{E}(X)]^2\text{]} \\
&\geq (\epsilon_{\mathrm{M}}(f)/2n)^2/2 = \Omega(\epsilon_{\mathrm{M}}(f)^2/n^2) && \text{[By Part 1 of the lemma]}
\end{aligned}$$

which establishes the second part of the theorem.

Chapter 6

Testing Monotonicity over General Ranges

We now reduce the problem of testing arbitrary-range functions to the simpler problem of testing boolean-range functions, which was considered in the preceding chapter. This reduction works not only for functions with domain Σ^n, but more generally when the domain is any partially ordered set S. The reduction is characterized by Theorem 4, which states that a certain type of monotonicity test for functions of the form $f : S \mapsto \{0,1\}$ also works well for functions of the form $f : S \mapsto \Xi$. Here Ξ is a finite totally ordered set of size r, which we can regard as the integers in the interval $[0, r-1]$. Furthermore, for simplicity, we assume that $r = 2^s$ for some integer s. All references to "edges" are references to edges of the partial order graph, whose vertices are strings in the domain S and directed edges correspond to ordered comparable pairs (i.e. (x, y) is an edge iff $x \prec y$).

To ensure that a function far from monotone can be readily detected by our test, we lower bound $\textsc{Detect}(f, D)$ in terms of $\epsilon_{\mathrm{M}}(f)$. Equivalently, we are looking for a good upper bound on $\epsilon_{\mathrm{M}}(f)$ in terms of $\textsc{Detect}(f, D)$. We reduce the task of obtaining an upper bound for functions with an arbitrary range to that of obtaining such an upper bound for functions with binary range.

The general idea of the reduction is to incrementally transform a function f into a monotone function, while ensuring that for each repaired violated edge, the value of the function is changed at only a few points. This transformation allows us to find a monotone function close to f and to upper bound $\epsilon_{\mathrm{M}}(f)$ by the distance from f to that function.

The transformation produces the following chain of functions: $f \mapsto f_1 \mapsto f_2 \mapsto f_3$, where f_3 is monotone. The distance between any two consecutive functions in the chain is equal to the distance to monotone of some auxiliary function with a smaller range. Thus we obtain an upper bound for $\epsilon_{\mathrm{M}}(f)$ in terms of the distance to monotone of smaller-range functions. In addition, edges violated by the auxiliary functions are also violated by f, and $\textsc{Detect}(f, D)$ can be lower bounded by the corresponding probability for the smaller-range auxiliary functions. Finally, we obtain an upper bound on $\epsilon_{\mathrm{M}}(f)$ in terms of $\textsc{Detect}(f, D)$, using an analogous bound for binary-range functions and the two bounds described above.

In Section 6.1 we describe and analyze operators SQUASH, MONO, and CLEAR later used to define functions f_1, f_2, and f_3 described above. In Section 6.2 we prove the range reduction lemma which upper bounds $\epsilon_{\mathrm{M}}(f)$ and lower bounds $\textsc{Detect}(f, D)$ by the corresponding quantities for smaller range functions. We conclude this chapter by Section 6.3 where we prove Theorem 4.

6.1 Operators SQUASH, MONO, and CLEAR

We start by defining operators that will be useful for obtaining functions f_1, f_2, and f_3 related to f.

Definition 7 *The operators* SQUASH, MONO, *and* CLEAR *each map a function $f : S \mapsto [0, r-1]$ to a related function with the same domain and the same or smaller range. In particular,* $\textsc{mono}[f]$ *is some arbitrary monotone function at distance $\epsilon_{\mathrm{M}}(f)$ from the function f. The operators* SQUASH *and* CLEAR *are defined below; in these definitions a and b are elements of $[0, r-1]$ and $a < b$.*

$$\textsc{squash}[f,a,b](x) = \begin{cases} a & \textit{if } f(x) \le a \\ b & \textit{if } f(x) \ge b \\ f(x) & \textit{otherwise} \end{cases}$$

$$\textsc{clear}[f,a,b](x) = \begin{cases} \textsc{mono}[\textsc{squash}[f,a,b]] & \\ \qquad \textit{if } \textsc{mono}[\textsc{squash}[f,a,b]](x) \neq \textsc{squash}[f,a,b](x) & \\ f(x) \quad \textit{otherwise} & \end{cases}$$

SQUASH operator simply "squashes" the range of f to $[a, b]$. Notice that if an edge is not violated by f, it is not violated by SQUASH$[f, a, b]$.

Claim 17 VIOL(SQUASH$[f, a, b]$) $\subseteq$ VIOL(f), *i.e.* SQUASH *does not introduce any new violated edges.* ■

CLEAR operator first "squashes" the range to $[a, b]$, then alters the resulting smaller-range function at some points to obtain the closest monotone function, and finally "unsquashes" the function at unaltered points to the original values. This leads to the following simple claim:

Claim 18 dist$(f,$ CLEAR$[f, a, b]) = \epsilon_{\mathrm{M}}($SQUASH$[f, a, b])$.

Proof: By definitions of the CLEAR and MONO operators:

$$\begin{aligned}\mathrm{dist}(f, \textsc{clear}[f,a,b]) &= \mathrm{dist}(\textsc{mono}[\textsc{squash}[f,a,b]], \textsc{squash}[f,a,b]) \\ &= \epsilon_{\mathrm{M}}(\textsc{squash}[f,a,b]).\end{aligned}$$

■

Define the *interval of a violated edge* (x, y) *with respect to function* f to be the interval $[f(y), f(x)]$ (since the edge is violated by f, $f(x) > f(y)$). We say that two intervals *cross* if they intersect in more than one point. Intuitively, if we consider $f(x) - f(y)$ as a measure of how violated an edge (x, y) is, then we can say that CLEAR$[f, a, b]$ partially repairs violated edges of f whose intervals cross $[a, b]$ without worsening other violated edges. The following lemma formalizes important properties of CLEAR.

Lemma 19 *The function* CLEAR$[f, a, b]$ *has the following properties for all* $f : S \mapsto [0, r-1]$ *and all* $a, b \in [0, r-1]$ *such that* $a < b$*:*

1. VIOL(CLEAR$[f, a, b]$) $\subseteq$ VIOL(f), *i.e.* CLEAR *does not introduce any new violated edges.*
2. CLEAR$[f, a, b]$ *has no violated edges whose intervals cross* $[a, b]$.
3. *The interval of a violated edge with respect to* CLEAR$[f, a, b]$ *is contained in the interval of this edge with respect to* f.

Proof: For brevity, define $g = \textsc{mono}[\textsc{squash}[f, a, b]]$ and $h = \textsc{clear}[f, a, b]$. Let (x, y) be an edge violated by h; that is, $h(x) > h(y)$. By its definition, g is monotone and takes values in $[a, b]$. Also notice that $h(x) = f(x)$ if $h(x) \notin [a, b]$, and $h(x) = g(x)$ if $h(x) \in [a, b]$. We consider four cases where each of $h(x)$ and $h(y)$ is either inside or outside the interval $[a, b]$.

- Case 1: $h(x), h(y) \in [a, b]$. This case cannot occur: (x, y) cannot be violated by h because $h(x) = g(x), h(y) = g(y)$ and g is monotone.
- Case 2: $h(x), h(y) \notin [a, b]$. Since f and h agree on both x and y, it follows that (x, y) is violated by f and $[h(y), h(x)] = [f(y), f(x)]$. This proves parts (1) and (3). To show that $[h(y), h(x)]$ does not cross $[a, b]$, it remains to prove that the case when $h(x) > b$ and $h(y) < a$ cannot happen. But in such a case we must have $g(x) = b$ and $g(y) = a$ and that contradicts the monotonicity of g.
- Case 3: $h(x) \notin [a, b]$, $h(y) \in [a, b]$. Since (x, y) is violated, $h(x) > b$. Consequently, $f(x) = h(x) > b$ and, thus, $g(x) = b$. Since g is monotone, $g(y) \geq g(x) = b$, and hence $h(y) = g(y) = b$. This proves that $[h(y), h(x)]$ intersects $[a, b]$ in at most one point (b), establishing part 2. If $f(y) = h(y) = b$, then f and h agree on both x and y, and parts (1) and (3) follow. If not, then $b = g(y) \neq \textsc{squash}[f, a, b](y)$. Thus, $\textsc{squash}[f, a, b](y) < b$, and hence $f(y) < b = h(y)$. Since $f(x) = h(x) > b$, parts (1) and (3) follow.
- Case 4: $h(x) \in [a, b], h(y) \notin [a, b]$. This case is symmetrical to Case 3.

■

6.2 Range Reduction

We are now ready to define functions in the chain $f \mapsto f_1 \mapsto f_2 \mapsto f_3$, as well as auxiliary smaller-range functions f_1', f_2', and f_3'. These functions and their properties are summarized in the following lemma.

The transition from f to f_1 transforms violated edges with one endpoint in the lower half of the range and the other endpoint in the upper half into edges with both endpoints in the same half of the range. After that we repair violated edges with both endpoints in the lower half of the range to obtain f_2 and then, upper half of the range to obtain f_3.

Lemma 20 (Range Reduction) *Define six functions in terms of $f : S \mapsto [0, r-1]$ as follows.*

$$f' = \textsc{squash}[f, \tfrac{r}{2}-1, \tfrac{r}{2}] \qquad f_1 = \textsc{clear}[f, \tfrac{r}{2}-1, \tfrac{r}{2}]$$
$$f_1' = \textsc{squash}[f_1, 0, \tfrac{r}{2}-1] \qquad f_2 = \textsc{clear}[f_1, 0, \tfrac{r}{2}-1]$$
$$f_2' = \textsc{squash}[f_2, \tfrac{r}{2}, r-1] \qquad f_3 = \textsc{clear}[f_2, \tfrac{r}{2}, r-1]$$

These functions have the following properties, for every probability distribution D.

1. $\textsc{Detect}(f, D) \geq \textsc{Detect}(f', D)$
2. $\textsc{Detect}(f, D) \geq \textsc{Detect}(f_1', D) + \textsc{Detect}(f_2', D)$
3. $\epsilon_{\mathrm{M}}(f) \leq \epsilon_{\mathrm{M}}(f') + \epsilon_{\mathrm{M}}(f_1') + \epsilon_{\mathrm{M}}(f_2')$

Proof: All references to "parts" are references to parts of Lemma 19.

(1) The SQUASH operator never adds new violated edges by Claim 17. Therefore, $\textsc{Viol}(f') \subseteq \textsc{Viol}(f)$, and the claim follows.

(2) We show that $\textsc{Viol}(f_1')$ and $\textsc{Viol}(f_2')$ are *disjoint* subsets of $\textsc{Viol}(f)$, and the claim follows. First, note that $\textsc{Viol}(f_1')$ and $\textsc{Viol}(f_2')$ are subsets of $\textsc{Viol}(f)$, because f_1' and f_2' are constructed from f using a sequence of CLEAR and SQUASH operators, which never add new violated edges by Claim 17 and part (1).

All that remains is to prove that $\textsc{Viol}(f_1')$ and $\textsc{Viol}(f_2')$ are disjoint. By part (2), there is no edge violated by f_1 whose interval crosses $[\frac{r}{2}-1, \frac{r}{2}]$. Therefore, the edges violated by f_1 are partitioned into two disjoint subsets: "low" edges with intervals contained in $[0, \frac{r}{2}-1]$ and "high" edges with intervals contained in $[\frac{r}{2}, r-1]$. The edges violated by f_1' are a subset of the low edges, because the SQUASH operation repairs all the high violated edges and adds no new violated edges by Claim 17. The edges violated by f_2' are a subset of the high edges, because the CLEAR operation used to form f_2 repairs all the low violated edges by parts (2) and (3), and no new violated edges are added by Claim 17 and part (1).

(3) First, we show that f_3 is monotone. Since the function f_3 is constructed from f using a sequence of three CLEAR operations, parts (2) and (3) imply that there is no edge violated by f_3 whose interval crosses any of the intervals $[\frac{r}{2}-1, \frac{r}{2}]$, $[0, \frac{r}{2}-1]$, or $[\frac{r}{2}, r-1]$. Therefore, f_3 violates no edges at all and is monotone.

Now the distance from f to the set of monotone functions is at most the distance from

f to the particular monotone function f_3, and we can reason as follows:

$$\begin{aligned}
\epsilon_{\mathrm{M}}(f) &\leq \mathrm{dist}(f, f_3) \\
&\leq \mathrm{dist}(f, f_1) + \mathrm{dist}(f_1, f_2) + \mathrm{dist}(f_2, f_3) \\
&= \epsilon_{\mathrm{M}}(f') + \epsilon_{\mathrm{M}}(f_1') + \epsilon_{\mathrm{M}}(f_2').
\end{aligned}$$

The last step uses Claim 18. ■

6.3 Proof of Theorem 4

In this section we use the results of the preceding lemma to prove Theorem 4. The proof is by induction on s with the inductive hypothesis that for every function $f : S \mapsto \Xi$ where $|\Xi| = 2^s$,

$$\epsilon_{\mathrm{M}}(f) \leq C \cdot \textsc{Detect}(f, D) \cdot s.$$

In the base case where $s = 1$, the hypothesis holds by the assumption stated in the theorem. Now assume that the hypothesis holds for $s - 1$ to prove that it holds for s. We can reason as follows:

$$\begin{aligned}
\epsilon_{\mathrm{M}}(f) &\leq \epsilon_{\mathrm{M}}(f') + \epsilon_{\mathrm{M}}(f_1') + \epsilon_{\mathrm{M}}(f_2') \\
&\leq C \cdot \textsc{Detect}(f', D) \\
&+ C \cdot \textsc{Detect}(f_1', D) \cdot (s-1) + C \cdot \textsc{Detect}(f_2', D) \cdot (s-1) \\
&\leq C \cdot (\textsc{Detect}(f, D) + \textsc{Detect}(f, D)(s-1)) \\
&= C \cdot \textsc{Detect}(f, D) \cdot s
\end{aligned}$$

The first inequality was proved in part (3) of Lemma 20. The second inequality uses the induction hypothesis; recall that the range of f' has size 2^1, and the ranges of f_1' and f_2' have size $r/2 = 2^{s-1}$. The third step uses parts (1) and (2) of Lemma 20, and the final step is simplification. This completes the proof.

Chapter 7

Open Problems

There are two problems in testing monotonicity of functions that we were not able to resolve so far. The first one is designing a gap tester for monotonicity whose complexity is independent of the size of the range (or proving that one does not exist). The second one is designing testers for monotonicity of functions defined on any partially ordered set.

It would also be interesting to see gap property testers for other properties and gap property testing itself applied in various areas of computer science.

Bibliography

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *JACM*, 45(3):501–555, 1998.

[2] S. Arora and S. Sudan. Improved low degree testing and its applications. In *Proceedings of STOC97*, pages 485–495, 1997.

[3] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of FOCS95*, pages 432–441, 1995.

[4] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of STOC93*, pages 294–304, 1993.

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

[6] F. Ergun, S. Kannan, S. R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. In *Proceedings of STOC98*, pages 259–268, 1998.

[7] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of STOC91*, pages 32–42, 1991.

[8] O. Goldreich, S. Goldwasser, E. Lehman, and D. Ron. Testing monotonicity. In *Proceedings of FOCS98*, 1998.

[9] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, 1998. An extended abstract appeared in the proceedings of FOCS96.

[10] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of STOC97*, pages 406–415, 1997.

[11] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. In *Proceedings of STOC98*, pages 289–298, 1998. To appear in *Combinatorica*, 1999.

[12] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC97*, pages 475–484, 1997.

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