COMS 6998: Communication Complexity **Spring 2026**

Property Testing Bounds Via Communication Complexity

Instructor: Toniann Pitassi *Presenter: Szymon Snoeck*

1 Introduction to Property Testing

Suppose you are given the truth table of some Boolean function, $f : \{0,1\}^n \to \{0,1\}$, and you want to, as quickly as possible, determine whether the function is linear. In your first-approach, you may try to solve this problem by a brute-force check. This solution is not particularly appealing as it requires reading the whole table, which is exponentially large in the size of the function's input, n. A better solution, if we are willing to allow randomness, was given by [BLR93]: simply test if linearity holds on random inputs. Specifically, draw uniformly random strings $x, y \sim \{0,1\}^n$, and check if $f(x) + f(y) = f(x+y)$. If the test fails, the function is for sure not linear, and if it succeeds, it must not be too "far" from linear. Repeating the test a constant number of times boosts the success of the test and gives an algorithm that only reads a constant number of input bits to determine whether f is (close to) linear. This is the central motivation of property testing: design query-efficient schemes to deduce whether an input has a given useful property.

Property testing, in short, can be thought of as an alternative notion of approximation for a decision problem (e.g., is f close to linear?). Due to its simple and broad nature, it appears across learning tasks, sublinear algorithms, the PCP theorem (perhaps the most famous application), and much more. The idea of a property can be formalized as such:

Definition 1. *Let D and R be a finite domain and range, respectively. A **property**, $\mathcal{P}$, is defined as some subset of all functions $f : D \to R$.*

Example 2. *$\mathcal{P} = \{$all linear $f : \{0,1\}^n \to \{0,1\}\}$ where a linear function is any function such that for all $x, y \in \{0,1\}^n$, $f(x+y) = f(x) + f(y)$.*[1]

Example 3. *$\mathcal{P} = \{$all monotonic $f : \{0,1\}^n \to \{0,1\}\}$ where a monotonic function is any function such that for all $x \in \{0,1\}^n$ and $i \in [n]$,*

$$f(x_1 \ \cdots \ x_{i-1} \ 0 \ x_{i+1} \ \cdots \ x_n) \leq f(x_1 \ \cdots \ x_{i-1} \ 1 \ x_{i+1} \ \cdots \ x_n).$$

[1] Note that, as is standard for $x, y \in \{0,1\}^n$, $+$ is bit-wise addition mod 2.

In other words, flipping a bit in the input from 0 *to* 1 *never decreases the output.*

Example 4. $\mathcal{P} = \{$*all* k*-juntas* $f : \{0,1\}^n \to \{0,1\}\}$ *where a function is a* k*-junta if its output is dependent on at most* k*-variables. Formally, that is to say there exist a subsequence* $i_1, \ldots, i_k \subseteq [n]$ *and* $g : \{0,1\}^k \to \{0,1\}$ *such that:*

$$f(x_1 \ \cdots \ x_n) = g(x_{i_1} \cdots x_{i_k}).$$

Properties do not always need to (directly) describe types of functions:

Example 5. $\mathcal{P} = \{$*all bipartite graphs*$\}$ *where here the function* $f : \{(u,v) : u \neq v \in V\} \to \{0,1\}$ *gives the adjacency matrix of a graph.*

In the standard model of property test, one has access to a "black-box" oracle that, for a query $x \in D$, returns $f(x)$. The goal is to test if $f \in \mathcal{P}$ using as few queries as possible. It is not hard to see that if one has to decide exact membership, in other words, $f \in \mathcal{P}$ versus $f \notin \mathcal{P}$, then it is impossible to use fewer than $O(|D|)$ queries. Indeed, if a given function f is equal to some $g \in \mathcal{P}$ on all but one input, deciding $f \in \mathcal{P}$ requires finding the single input where they differ. To overcome this barrier, we relax the notion of $f \notin \mathcal{P}$ to f is far from any $g \in \mathcal{P}$:

Definition 6. *For* $f, g : D \to R$ *and* $\epsilon > 0$*, we say that* f, g *are* ϵ*-close if:*

$$\mathbb{P}_{x \sim D}[f(x) \neq g(x)] < \epsilon,$$

where x *is chosen uniformly from* D*. Symmetrically,* f, g *are* ϵ*-far if:*

$$\mathbb{P}_{x \sim D}[f(x) \neq g(x)] \geq \epsilon.$$

With respect to a property $\mathcal{P}$*, we say that* f *is* ϵ*-far from* $\mathcal{P}$ *if for all* $g \in \mathcal{P}$*,* f *is* ϵ*-far from* g*.*

Intuitively, f ϵ-far from $\mathcal{P}$ tells us that the cost to repair f so that it has property $\mathcal{P}$ is large. Now we are ready to formally state what a property tester is:

Definition 7. *Fix* $\epsilon > 0$*, a randomized decider*[2]*,* T*, is a* ***tester for property*** $\mathcal{P}$ ***with parameter*** ϵ *if:*

1. *For all* $f \in \mathcal{P}$*,* T *accepts with probability* $\geq 2/3$*,*
2. *For all* f ϵ*-far from* $\mathcal{P}$*,* T *rejects with probability* $\geq 2/3$*.*

[2] A randomized algorithm that can only accept or reject.

*The **query complexity of a property** $\mathcal{P}$, $Q_\epsilon(\mathcal{P})$, is the maximum number of queries used by an optimal tester for $\mathcal{P}$.*

As is commonplace in communication complexity and other domains of computational complexity, the choice of $2/3$ is arbitrary, and by repeating independent copies of T, the failure probability can be made arbitrarily small.

Revisiting the linearity test of [BLR93], commonly referred to as the BLR linearity test, we can now rigorously state [BLR93]'s result:

Theorem 8 (BLR Linearity Test)**.** *For any integer $n \in \mathbb{N}$ and $\epsilon > 0$, there exists a tester for $\mathcal{P} = \{$all linear $f : \{0,1\}^n \to \{0,1\}\}$ using $\Theta(1/\epsilon)$ queries.*

The tester simply repeats the test $f(x+y) = f(x) + f(y)$, for $x, y \sim \{0,1\}^n$, $\Theta(1/\epsilon)$ times. If any of the tests fail, it rejects f, and if all tests succeed, it accepts f. The proof of correctness is non-trivial, and for a clean proof using Boolean Fourier analysis, see Chapter 1 of [O'D21].

1.1 Different Types of Tests

Property testers can be categorized in two main ways: adaptive versus non-adaptive, and two-sided versus one-sided error. A test is *adaptive* if the next query is decided using the output of previous queries, and *non-adaptive* if all queries can be chosen up-front. A test has *two-sided error* if it can falsely reject $f \in \mathcal{P}$ (as is the case in Definition 7) and *one-sided error* if it always accepts $f \in \mathcal{P}$. Generally, adaptivity and two-sided error do not reduce the necessary number of queries. For instance, in the case of linearity testing, the BLR linearity test, which is non-adaptive and has one-sided error, achieves the optimal query complexity [Lov08].

There are, however, some cases where adaptivity and two-sided error do help. For the case of two-sided error, consider the property $\mathcal{P}_n = \{$ all simple n-vertex graphs with average degree greater than $k\}$.[3] For some $\epsilon = \Theta(1)$, the property testing question is equivalent to deciding whether a graph has $\geq \frac{kn}{2}$ edges or $\leq \frac{kn}{2} - \epsilon\binom{n}{2}$. The simplest strategy is to sample m random pairs of vertices and compute the empirical probability of an edge. If this empirical estimate is greater than a carefully chosen threshold, accept; otherwise, reject. Clearly, this test has two-sided error, and by application of a Chernoff bound, requires only $\Theta(1)$ queries. On the other hand, any one-sided test must check at least $\frac{kn}{2} - \epsilon\binom{n}{2}$ edges to be sure that it does not reject any graph in $\mathcal{P}$.

[3] Assume that all $f : \{(u,v) : u \neq v \in V\} \to \{0,1\}$ represent some simple graph.

Adaptivity becomes useful whenever we wish to implement a search strategy. One case where we should expect this to help is in identifying k-juntas. Intuitively, adapting each query to the information learned in the past queries allows us to hone in on the sensitive coordinates faster. Indeed, adaptivity begets a poly-time improvement from $\Theta(k^{3/2}/\epsilon)$ to $O(k\epsilon + k\log k)$ [CST+17][Bla08][Bla09]. The gap between adaptivity and non-adaptivity can even be exponential. For a general analysis of adaptivity, see [CG18] who prove that there exist problems where using even one less adaptive query that optimal can blow up the query complexity.

1.2 A Short History Lesson On Monotonicity Testing

The field of property testing got its unofficial start in 1993 with the creation of the BLR linearity test [BLR93]. This simple yet powerful algorithm inspired Rubinfeld, and Sudan in 1996 and later, Goldreich, Goldwasser, and Ron to formalize the field [RS96][GGR98]. Since the nineties, the literature on property testing has grown vast, see for instance [Ron08]. At the center of much of this literature is the following, seemingly innocent, problem introduced by Goldreich et al. in 2000: determine the query complexity of monotonicity testing for Boolean functions [GGL+00] (see example 3 for precise definition).

In the original paper from 2000, Goldreich et al. showed that a "test locally" strategy, similar to that of the BLR linearity test, could achieve a query complexity of $O(n/\epsilon)$. The test proceeded by picking random $x \sim \{0,1\}^n$, $i \sim [n]$ and testing if:

$$f(x_1 \ \cdots \ x_{i-1} \ 0 \ x_{i+1} \ \cdots \ x_n) \leq f(x_1 \ \cdots \ x_{i-1} \ 1 \ x_{i+1} \ \cdots \ x_n).$$

Repeating this $\Theta(n/\epsilon)$ times was shown to be sufficient to give a monotonicity tester. Despite the success of the similar BLR linearity test, little was known about how optimal Goldreich et al.'s algorithm was. In 2002, [FLN+02] showed a lower bound for non-adaptive one-sided tests of $\Omega(\sqrt{n})$, then for over a decade no progress was made. The stand still was only broken in 2013 by the break through paper by Chakrabarty and Seshadhri which showed that a query complexity of $\tilde{O}(n^{7/8}/\epsilon^{3/2})$ was possible [CS13]. The novel idea was to test monotonicity over many bit flips rather than just one. A "bit" more specifically, the test proceeded by picking $x \sim \{0,1\}^n$ and flipping some carefully chosen random subset of x's 0-bits to get a new string $y \in \{0,1\}^n$ with $x \leq y$. As before, the actual test was to query if $f(x) \leq f(y)$. This idea stuck, and in subsequent refinements by Chen et al. and Khot et al. the bound was lowered to $\tilde{O}(\sqrt{n}/\epsilon^2)$ [CST14][KMS15].

Finding a matching lower bound proved hard due to difficulty of handling adaptive algorithms. In 2015, Chen et al. improved the lower bound for non-adaptive, two-sided algorithms to $\Omega(n^{(1/2)-c})$, for all $c > 0$, (improving on [FLN+02]) however a similar bound for adaptive algorithms evaded researchers. At the time, the best known lower bound for adaptive two-sided testers was $\tilde{\Omega}(n^{1/3})$ (given by Chen et al. of course) [CWX17]. It would take another 7 years before the query complexity of monotonicity testing was finally settled by Chen et al. in 2025 who proved a lower bound for adaptive two-sided testers of $\Omega(n^{(1/2)-c})$ for all $c > 0$ [CCC+25]!

2 Lower Bounds for Monotonicity Testing via Communication Complexity

While there are perhaps many insights and morals to learn from the previously described history, one lesson which is apparent is the challenging nature of proving adaptive two-sided lower bounds. Thus, it was quite a surprise when in 2012 Blais et al. showed a general and powerful framework to turn well known lower bounds in communication complexity into property testing lower bounds for all testers [BBM12]. To demonstrate the usefulness of this method, we will prove the following lower bound:

Theorem 9 ([BBM12])**.** *For $\epsilon = 1/8$, every tester for the property:*

$$\mathcal{P} = \{\textit{all monotonic functions } f : \{0,1\}^n \to \mathbb{Z}\}$$

with parameter ϵ must use $\Omega(n)$ queries.

Note that the fact that the range $\mathbb{Z}$ is infinite is irrelevant since the image of f is finite and all that matters is the ordering of the range. Identically one could rewrite the property in terms of functions $f : \{0,1\}^n \to [2^n]$ however for the ease of notation, we will proceed with the range being $\mathbb{Z}$.

It is also worth while to point out that, while the above is a lower bound for monotonicity testing, it is not a lower bound for the boolean case where the function's output range is $\{0,1\}$. Hence it does not applying to the previously discussed history.

In section 2.1, we prove Theorem 9. In section 2.2, we will show how the proof can be generalized to other property testing problems, and in the final section, open directions in the field will be discussed.

2.1 Proof of Theorem 9

The following proof is adapted directly from Tim Roughgarden's presentation of the proof in Chapter 9 of [Rou15]. All mistakes introduced during adaptation are my own.

As always, the first thing to try is a reduction from disjointness, with the query complexity somehow translating to the communication cost. At first this might seem weird — there's only one "player" in property testing, so where do Alice and Bob come from? But as we've seen over and over again, starting with our applications to streaming lower bounds, it can be useful to invent two parties just for the sake of standing on the shoulders of communication complexity lower bounds. To implement this, we need to show how a low-query tester for monotonicity leads to a low-communication protocol for disjointness.

It's convenient to reduce from a "promise" version of disjointness that is just as hard as the general case. In the unique-disjointness problem, the goal is to distinguish between inputs where Alice and Bob have sets A and B with $A \cap B = \emptyset$, and inputs where $|A \cap B| = 1$. On inputs that satisfy neither property, any output is considered correct. Razborov's famous proof that every randomized protocol for disjointness with two-sided error requires $\Omega(n)$ communication inadvertently also proves a $\Omega(n)$ lower bound for unique-disjointness as the hard probability distribution in this proof makes use only of inputs with intersection size 0 or 1 [Raz92].

Key to the proof of Theorem 9 is the following lemma.

Lemma 10. *Fix sets $A, B \subseteq U = [n]$. Define the function $h_{AB} : 2^U \to \mathbb{Z}$ by*

$$h_{AB}(S) = 2|S| + (-1)^{|S \cap A|} + (-1)^{|S \cap B|}. \tag{1}$$

Then:

(i) If $A \cap B = \emptyset$, then h is monotone.

(ii) If $|A \cap B| = 1$, then h is $\frac{1}{8}$-far from monotone.

Note that this function can be identically recast as a function $h_{AB} : \{0,1\}^n \to \mathbb{Z}$ simply by the typical identification of a subset $A \subseteq [n]$ with a string $A \in \{0,1\}^n$ such that $A_{i \in [n]} = 1$ iff $i \in A$.

We'll prove the lemma shortly; let's first see how to use it to prove Theorem 9. Let T be a tester that distinguishes between monotone functions from $\{0,1\}^n$ to $\mathbb{Z}$ and functions that are $\frac{1}{8}$-far from monotone. We proceed to construct a (public-coin randomized) protocol for the unique-disjointness problem.

Suppose Alice and Bob have sets $A, B \subseteq \{1, 2, \ldots, n\}$. The idea is for both parties to run local copies of the tester T to test the function h_{AB}, communicating with each other as needed to carry out these simulations. In more detail, Alice and Bob first use the public coins to agree on a random string to be used with the tester T. Given this shared random string, T is deterministic. Alice and Bob then simulate local copies of T query-by-query:

1. Until T halts:
   (a) Let $S \subseteq [n]$ be the next query that T asks about the function h_{AB}.
   (b) Alice sends $(-1)^{|S \cap A|}$ to Bob.
   (c) Bob sends $(-1)^{|S \cap B|}$ to Alice.
   (d) Both Alice and Bob evaluate the function h_{AB} at S, and give the result to their respective local copies of T.
2. Alice (or Bob) declares "disjoint" if T accepts the function h_{AB}, and "not disjoint" otherwise.

We first observe that the protocol is well defined. Since Alice and Bob use the same random string and simulate T in lockstep, both parties know the (same) relevant query S to h_{AB} in every iteration, and thus are positioned to send the relevant bits ($(-1)^{|S \cap A|}$ and $(-1)^{|S \cap B|}$) to each other. Given these bits, they are able to evaluate h_{AB} at the point S (even though Alice doesn't know B and Bob doesn't know A).

The communication cost of this protocol is twice the number of queries used by the tester T, and it doesn't matter if T is adaptive or not. Correctness of the protocol follows immediately from Lemma 10, with the error of the protocol the same as that of the tester T. Because every randomized protocol (with two-sided error) for unique-disjointness has communication complexity $\Omega(n)$, we conclude that every (possibly adaptive) tester T with two-sided error requires $\Omega(n)$ queries for monotonicity testing. This completes the proof of Theorem 9.

Proof of Lemma 10. For part (i), assume that $A \cap B = \emptyset$ and consider any set $S \subseteq [n]$ and $i \notin S$. Because A and B are disjoint, i does not belong to at least one of A or B. Recalling (1), in the expression $h_{AB}(S \cup \{i\}) - h_{AB}(S)$, the difference between the first terms is 2, the difference in either the second terms (if $i \notin A$) or in the third terms (if $i \notin B$) is zero, and the difference in the remaining terms is at least -2. Thus, $h_{AB}(S \cup \{i\}) - h_{AB}(S) \geq 0$ for all S and $i \notin S$, and h_{AB} is monotone.

For part (ii), let $A \cap B = \{i\}$. For all $S \subseteq [n] \setminus \{i\}$ such that $|S \cap A|$ and $|S \cap B|$ are both even, $h_{AB}(S \cup \{i\}) - h_{AB}(S) = -2$. If we choose such an S uniformly at random, then $\mathbb{P}|S \cap A|$ is even is 1 (if $A = \{i\}$) or $\frac{1}{2}$ (if A has additional elements, using the Principle of Deferred Decisions). Similarly, $\mathbb{P}|S \cap B|$ is even $\geq \frac{1}{2}$. Since no potential element of $S \subseteq \{1, 2, \ldots, n\} \setminus \{i\}$ is a member of both A and B, these two events are independent and hence $\mathbb{P}|S \cap A|, |S \cap B|$ are both even $\geq \frac{1}{4}$. Thus, for at least $\frac{1}{4} \cdot 2^{n-1} = 2^n/8$ choices of S, $h_{AB}(S \cup \{i\}) < h_{AB}(S)$. Since all of these monotonicity violations involve different values of h_{AB} (indeed each $S, S \cup \{i\}$ pair is unique) fixing all of them requires changing h_{AB} at $2^n/8$ values. We conclude that h_{AB} is $\frac{1}{8}$-far from a monotone function.

■

2.2 A General Framework for Property Testing Lower Bounds

The proof of Theorem 9 elucidates a more general framework for transforming property testing problems into communication problems. The general framework for testing a property $\mathcal{P}$ is as follows:

1. For a suitable communication problem Π and $\epsilon > 0$, construct a map from inputs (A, B) to functions $h_{(A,B)}$ such that:
   - If $\Pi(A, B) = 1$, then $h_{(A,B)}$ has property $\mathcal{P}$;
   - If $\Pi(A, B) = 0$, then $h_{(A,B)}$ is ϵ-far from $\mathcal{P}$.
2. Construct a protocol such that both Alice and Bob can evaluate $h_{(A,B)}$ using d bits of communication.

Given such a mapping of inputs to functions, it is not hard to see that the simulation argument from Theorem 9 shows that the query complexity of $\mathcal{P}$ is at least $\frac{1}{d}$·the communication complexity of Π.

Using the above model [BBM12] proved the following theorem and corollary:

Theorem 11 ([BBM12]). *For $\epsilon = 1/2$, every tester for the property:*[4]

$$\mathcal{P} = \{\textit{all linear } k\textit{-juntas } f : \{0,1\}^n \to \{0,1\}\}$$

with parameter ϵ must use $\Omega(\min(k, n-k))$ queries.

[4]See example 2 and example 4 for a definition of linear functions and k-juntas.

Corollary 12 ([BBM12])**.** *For $\epsilon = 1/2$, every tester for the property:*

$$\mathcal{P} = \{\textit{all } k\textit{-juntas } f : \{0,1\}^n \to \{0,1\}\}$$

with parameter ϵ must use $\Omega(\min(k, n-k))$ queries.

Here, the communication framework gives close to optimal bounds: [Bla09] showed $O(k \log k)$ queries are sufficient for testing k-juntas. Thus, this framework provides an easy and general way to obtain property testing lower bounds from well-known communication complexity results. The only drawback is that the properties for which these methods seem to work best can sometimes be slightly orthogonal to those we really care about. For instance, Theorem 9 shows strong lower bounds for monotonicity testing functions with large ranges but breaks down when we try to apply it to the more popular case of monotonicity testing Boolean functions.

For more examples of applications of this technique see [BBM12].

2.3 Open Questions

Since the lauded paper of [BBM12], there have been several lines of work trying to build on the lower bound framework with respect to a variety of settings:

- In the most general of the extensions, [Gol20] showed how the framework can be made easier to work with and how it can be used specifically for non-adaptive testers.
- Another line of work is to expand the communication framework to achieve lower bounds for property testing on graphs. See [ER18] for some of the background and current work on the subject.
- There has also been interest in extending the framework to the realm of distribution testing. In particular, [BCG19] has shown that there is a connection between distribution testing, the simultaneous message passing (SMP) communication mode, and functional analysis. See also [DGKR19].

References

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Comput. Complex.*, 21(2):311–358, June 2012. 2, 9, 2.2, 11, 12, 2.2, 2.3

[BCG19] Eric Blais, Clément L. Canonne, and Tom Gur. Distribution testing lower bounds via reductions from communication complexity. *ACM Trans. Comput. Theory*, 11(2), February 2019. 2.3

[Bla08] Eric Blais. Improved bounds for testing juntas. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 317–330, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 1.1

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 151–158, New York, NY, USA, 2009. Association for Computing Machinery. 1.1, 2.2

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, December 1993. 1, 1, 1.2

[CCC+25] Mark Chen, Xi Chen, Hao Cui, William Pires, and Jonah Stockwell. Boolean function monotonicity testing requires (almost) $n^{1/2}$ queries, 2025. 1.2

[CG18] Clément L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Comput. Complex.*, 27(4):671–716, December 2018. 1.1

[CS13] Deeparnab Chakrabarty and C. Seshadhri. A o(n) monotonicity tester for boolean functions over the hypercube. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 411–418, New York, NY, USA, 2013. Association for Computing Machinery. 1.2

[CST14] Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, page 286–295, USA, 2014. IEEE Computer Society. 1.2

[CST+17] Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the Query Complexity of Non-Adaptive Junta Testing. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:19, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 1.1

[CWX17] Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 523–536, New York, NY, USA, 2017. Association for Computing Machinery. 1.2

[DGKR19] Ilias Diakonikolas, Themis Gouleakis, Daniel Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions, 06 2019. 2.3

[ER18] Talya Eden and Will Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2.3

[FLN+02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 474–483, New York, NY, USA, 2002. Association for Computing Machinery. 1.2

[GGL+00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20, 06 2000. 1.2

[GGR98] Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, July 1998. 1.2

[Gol20] Oded Goldreich. *On the Communication Complexity Methodology for Proving Lower Bounds on the Query Complexity of Property Testing*, pages 87–118. 04 2020. 2.3

[KMS15] Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric type theorems. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, page 52–58, USA, 2015. IEEE Computer Society. 1.2

[Lov08] Shachar Lovett. Lower bounds for adaptive linearity tests. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science*, volume 1 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 515–526, Dagstuhl, Germany, 2008. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 1.1

[O'D21] Ryan O'Donnell. Analysis of boolean functions, 2021. 1

[Raz92] A.A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. 2.1

[Ron08] Dana Ron. Property testing: A learning theory perspective. *Found. Trends Mach. Learn.*, 1(3):307–402, March 2008. 1.2

[Rou15] Tim Roughgarden. Communication complexity (for algorithm designers). *CoRR*, abs/1509.06257, 2015. 2.1

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials withapplications to program testing. *SIAM J. Comput.*, 25(2):252–271, February 1996. 1.2