Property Testing and Communication Complexity

Lecturer: Nicolas Pena

1 Outline

In this lecture, we provide a brief introduction to property testing. We discuss how to reduce communication complexity problems to property testing problems. We provide some examples of specific reductions that allow concluding lower bounds on the query complexity of testing some function properties. We conclude by mentioning some further work and open problems.

2 Introduction

Suppose that we are given as input an object O and we'd like to know if it has some property P. But querying the object is expensive, so we would like to do a small number of local queries on the object. Usually, deciding with certainty whether an object has a property or not will require a lot of queries. The idea of property testing is to answer an easier question, thus allowing less queries to be made.

In property testing, given the object O, the tester has to distinguish in between objects that satisfy the property P and objects that are far from the property. The notion of "being far" depends on what the object is (distribution, function, graph...). The tester's efficiency is measured by the number of queries it does on the object.

2.1 Applications

Property testing is useful in several settings. First, it is useful in program testing. Usually, it is desirable to test a program, which is designed to calculate some function, in several steps. To do this, the tester first checks whether the program satisfies some properties that are satisfied by the function that the program is supposed to compute. Then, the tester may do a more complete testing of the program.

Second, property testing is useful in probabilistically checkable proofs. In this setting, the concept of testing whether a function is linear or far from linear without looking up too many values of the function is very important. It is well known how to do this: the tester chooses random inputs x and y. The tester queries $f(x)$, $f(y)$, and $f(x \oplus y)$. Then, the tester accepts or rejects according to whether $f(x) \oplus f(y) = f(x \oplus y)$ or not. It is easy to see that the tester will always accept linear functions. It can be proved that the tester has a good chance of rejecting functions that do not look at all like a linear function.

Another context in which property testing is useful is computational learning. Suppose that we are given some pairs of values for an unknown function f: $(x_1, f(x_1))$, $(x_2, f(x_2))$, ... We would like to find a function h that approximates f, and we have two heuristics to do this. One of the heuristics works very well in some cases, but is computationally very intensive. The other one can

be applied to any sample data and is very efficient but does not provide very good results. To decide which of the heuristics to use, we can first test whether the data given has some properties that indicate that it's worthwhile to use the computationally expensive heuristic.

Property testing is also useful to test graphs. Suppose that a graph is represented using an adjacency matrix. Rivest and Vuillemin [7] showed that, to decide any monotone non-trivial graph property on graphs, a deterministic algorithm needs to query $\Omega(n^2)$ entries of the matrix. Yao, King, and Hanjal did work that concluded in a proof [6] of an $\Omega(n^4/3)$ lower bound for randomized algorithms. Thus, to have a chance of doing a small number of queries, we need to relax the question, as is done in property testing.

2.2 Definitions: Testing Functions

In this lecture, we will focus on property testing of functions. We consider properties of functions as sets of functions. Thus, saying that a function satisfies a property means that the function belongs to the set corresponding to the property.

Definition Let $f : \{0,1\}^n \to Y$ be a function and let $P \subseteq \{g \mid g : \{0,1\}^n \to Y\}$ be a property of functions. We say that f is ϵ-far from P if f has to be modified on an ϵ fraction of the inputs to have property P. More precisely, if f and g are functions, define $d(f,g) := |\{x : f(x) \neq g(x)\}|$. Also, if P is a property of functions, define $d(f,P) := \min\{d(f,g) : g \in P\}$. Then we say f is ϵ-far from P if $d(f,P) \geq \epsilon 2^n$.

Definition Let T be a randomized decider (algorithm that can only accept or reject) with input 1^n and oracle access to a function $h : \{0,1\}^n \to Y$. We say that T is a tester for property P if the following two conditions are satisfied:

1. If f satisfies P, then T accepts with probability at least $2/3$.
2. If f is ϵ-far from P, then T rejects with probability at least $2/3$.

Notice that we impose no conditions on functions that do not satisfy P but are close to P. For these, T can behave in any way. Also, the $2/3$ in the definition is arbitrary. By repeatedly running T, this probability may be improved. Finally, notice that no computational constraints are enforced on T. The only reason to give it 1^n as input is so that it knows what the value of n is.

Definition Let P be a property of functions. The query complexity of P is the minimum amount of queries that a tester for P does. We denote it by $Q(P)$. This will usually depend on n and ϵ. In this lecture, we will consider only fixed constant ϵ's, so our $Q(P)$ will only depend on n.

In property testing, several types of testers may be considered. A tester can be *adaptive* or *non-adaptive*. In the non-adaptive case, the algorithm has to give all the queries that it will do before the answers to those queries are provided. Adaptive testers are more powerful because they can choose their queries according to what the oracle answers in previous queries.

The tester can have *one-sided error* or *two-sided error*. A one-sided error tester must always accept whenever it is querying a function that satisfies P. It may still make mistakes when querying a function that is far from P. Two-sided error is defined in the obvious way. Our definition above captures adaptive testers with two-sided error. This is the most powerful model, so lower bounds on these testers imply lower bounds on the other types of testers. In what remains of the lecture, we assume that the testers are adaptive with two-sided error, unless stated otherwise.

3 Reductions From Communication Complexity

Communication complexity can be used to prove lower bounds in property testing. This was first shown by Blais et al [2]. They key idea is that, in both settings, we have parties with unlimited computational power (Alice, Bob, and the tester) that have limited access to some input. We now describe what the reductions from communication complexity to property testing will look like.

Let P be a property that we would like to test. We choose a function C of hard randomized communication complexity. Of course, not any function C will work: the reduction below has to work out properly.

Suppose that Alice is given an input A and Bob is given an input B (these letters will be used only for this purpose throughout the lecture). Before any communication is performed, Alice constructs a function f_A (that depends on A) and Bob constructs a function g_B (depending on B). These two functions together define a test function h_{AB} such that: if $C(A, B) = 1$, then h_{AB} satisfies property P, and if $C(A, B) = 0$ then h_{AB} is far from P. To compute $C(A, B)$ the players can simulate a tester for P on the function h.

One way to accomplish this is to require that for any x in the domain of h_{AB}, $h_{AB}(x)$ can be computed from $f_A(x)$ and $g_B(x)$ – thus if Alice sends $f_A(x)$ and Bob sends $g_B(x)$ then they both know $h_{AB}(x)$. Because the tester is adaptive, both Alice and Bob have to simulate it. Suppose only Alice simulates the tester. Bob can easily know the first x that the tester will query, so he sends $g_B(x)$. Thus, Alice is now able to compute $h(x)$ and continue the simulation. However, Bob does not know $h(x)$, and in particular Bob does not know what is the next query the tester will make (it may depend on $h(x)$). Thus, if the next value queried is y, Alice would have to send y to Bob so that Bob knows that he has to send her $g_B(y)$. This is inefficient in terms of communication complexity, so it will not provide useful lower bounds.

Now we summarize the randomized protocol that Alice and Bob do in order to solve C. If Alice is given A and Bob is given B:

- Alice constructs f_A and Bob constructs g_B.
- Both begin to simulate a tester for h.
- Whenever the tester requires random bits, Alice and Bob use bits from the shared string on the board.
- Whenever the tester does a query for some value x, Alice sends $f_A(x)$ to Bob, and Bob sends $g_B(x)$ to Alice. This way, they can answer the tester's query with $h(x)$.
- At the end, the tester says whether h is far from P or has P. By the properties of h listed above, this tells Alice and Bob whether $C(A, B) = 0$ or $C(A, B) = 1$.

Suppose that the tester does q queries on h and that Alice and Bob can communicate both $f_A(x)$ and $g_B(x)$ using d bits. Then this yields a randomized communication complexity protocol of qd bits that solves C. Therefore, a lower bound of M for the randomized communication complexity of C yields a lower bound of M/d for $Q(P)$. In the reductions we'll do, the size of the ranges of f_A and g_B will be constant, so d will be constant and an $\Omega(M)$ communication complexity lower bound will yield an $\Omega(M)$ property testing lower bound.

For lower bounds with adaptive testers and two-sided error, we use communication complexity lower bounds with two-sided error. If the tester has one-sided error, we may use one-sided error

communication complexity (assuming that the sides where the errors may occur do match). If the tester is non-adaptive, then there is no problem with the idea of only one player simulating the tester, so we can use one-way communication complexity lower bounds. That is, only Bob sends messages ($g_B(x)$ for the x values that the tester queries) to Alice, who simulates the tester and at the end knows $C(A, B)$.

4 Examples

On the examples that follow, we start with some property P. We state the function C we will reduce from, how Alice and Bob construct f_A and g_B, and what h is. Then we prove that the desired relationship between C, P, and h holds. Since C will have known lower bounds in communication complexity, this translates to lower bounds in property testing. The examples may be found in [2].

4.1 Testing k-linearity

Definition Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. We say that f is k-linear if it is linear on k components: there exists $S \subseteq [n]$ with $|S| = k$ such that $f(x) = \oplus_{i \in S} x_i$. We say that f is a k-junta if it depends on at most k components: there exists $S \subseteq [n]$ with $|S| = k$ such that for all x and y, if $x_i = y_i$ for all $i \in S$ then $f(x) = f(y)$. Notice that we can view f as having range $\{-1, 1\}$ instead of $\{0, 1\}$ by mapping 0 to 1 and 1 to -1.

Let $g : \{0,1\}^n \to \{-1, 1\}$ be a function (with range $\{-1, 1\}$). It is well-known that g has a unique Fourier representation. That is, there exists a unique function $\hat{g}$ that maps subsets of $[n]$ to numbers such that:

$$g(x) = \sum_{S \subseteq [n]} \hat{g}(S)\chi_S(x)$$

Here, $\chi_S(x)$ is the character function, defined by:

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$$

The Fourier degree of g is the size of the largest S such that $\hat{g}(S) \neq 0$. It is easy to see that being k-linear implies being a k-junta, which implies having Fourier degree at most k.

Definition Let t-UDISJ be unique set disjointness with the additional promise that the sets given to Alice and Bob are of size t. We will use this problem with a large t. It is known that the randomized communication complexity of this problem is $\Omega(t)$.

Theorem 1 *Testing k-linearity (with $\epsilon = 1/2$) requires $\Omega(min\{k, n-k\})$ queries.*

Proof Suppose that $0 < k < n/2$. Let $t = k/2 + 1$. We reduce from t-UDISJ. Alice and Bob are given A and B, and they view them as subsets of $[n]$. Given A, Alice takes $f_A(x) := \oplus_{i \in A} x_i$ and Bob takes $g_B(x) := \oplus_{i \in B} x_i$. They run the k-linearity tester on $h := f_A \oplus g_B$. Now, we will prove that this reduction works: disjoint sets will yield a function far from k-linearity while sets that intersect on one element will yield a k-linear function.

If A and B are disjoint, then h is $(k+2)$-linear, since none of the terms in $f_A \oplus g_B$ are repeated and there are $2t = k + 2$ such terms. The lemma below will show that these functions are far

from k-linearity. The lemma actually shows something stronger: they are far from the property of having Fourier degree at most k.

On the other hand, if A and B intersect at exactly one element, then there will be two repeated terms in $f_A \oplus g_B$. This is because, if $i \in A$, then x_i will appear both in A and in B. These terms cancel out, yielding k distinct terms in h. Thus, in this case, we obtain a k-linear function.

We can reduce the case $k > n/2$ to this case because the query complexity of distinguishing k-linearity versus $k+2$-linearity is equal to that of distinguishing $(n-k)$-linearity versus $(n-k-2)$-linearity. This can be seen by replacing the function h being tested by $h \oplus x_1 \oplus ... \oplus x_n$. We can use a padding argument for $k = n/2$.

Lemma 2 *Any $(k+2)$-linear function is $1/2$-far from functions of Fourier degree at most k.*

Proof Let h be $(k+2)$-linear. Then h is of the form: $h(x) = \oplus_{i \in T} x_i$ where T is a subset of $[n]$ of size $k+2$. So $\hat{h}(T) = 1$ and $\hat{h}(S) = 0$ for all $S \neq T$. If g is a function of Fourier degree at most k, then $\hat{g}(T) = 0$. Therefore, by Parseval's Theorem:

$$E_x[h(x)g(x)] = \sum_{S \subseteq [n]} \hat{h}(S)\hat{g}(S) = 0$$

Since h and g are being viewed as having range $\{-1, 1\}$, then $h(x)g(x) = 1$ whenever $h(x) = g(x)$ and $h(x)g(x) = -1$ whenever $h(x) \neq g(x)$. Since the expected value of the product is 0, then h and g differ on exactly $1/2$ of the inputs. Since g was an arbitrary function of Fourier degree at most k, then h is $1/2$-far from functions of Fourier degree at most k.

Corollary 3 *The properties of being a k-junta and having Fourier degree at most k (with $\epsilon = 1/2$) have $\Omega(min\{k, n-k\})$ query complexity.*

Proof This follows using exactly the same reduction as in the previous theorem. When A and B are disjoint, then h is $(k+2)$-linear, which is far from the property (k-junta or Fourier degree at most k), by the lemma. And when A and B intersect in one element, then h is k-linear, so in particular it is also a k-junta and has Fourier degree at most k.

4.2 Testing Monotonicity

Definition Let Y be a totally ordered set. We say $f : \{0,1\}^n \to Y$ is a monotone function if for all $x, y \in \{0,1\}^n$: if $x \leq y$ then $f(x) \leq f(y)$. Here, by $x \leq y$ we mean component by component: $x_1 \leq y_1, x_2 \leq y_2, ..., x_n \leq y_n$.

Theorem 4 *Testing monotonicity (with $\epsilon = 1/8$) requires $\Omega(min\{n, |Y|^2\})$ queries.*

Proof We will only prove this for $Y = \mathbb{Z}$. This is the most interesting case. When $|Y| = \sqrt{n}$, a range reduction argument may be used (h is modified slightly so that it fits into the range). The case in which $|Y| = o(n)$ can be reduced from the case in which $|Y| = \sqrt{n}$.

Take $Y = \mathbb{Z}$. We use a reduction from DISJ. Recall that this function says whether A and B, viewed as subsets of $[n]$, are disjoint or not. This function is known to have randomized communication complexity of $\Omega(n)$. For the reduction, Alice constructs $f_A(x) := \chi_A(x)$, and Bob constructs $g_B(x) := \chi_B(x)$. They will test monotonicity on h defined by $h(x) := 2|x| + f_A(x) + f_B(x)$

where $|x|$ denotes the Hamming weight (number of 1's) of the vector x. We now need to show that this reduction works: disjoint sets yield a monotone h while sets that intersect yield a function h that is far from being monotone.

Let $i \in [n]$ and $x \in \{0,1\}^n$. Let $x^{i:0}$ be the vector that is obtained by setting x's i-th bit to 0 (it may already be 0, and in that case we do nothing). Let $x^{i:1}$ be the vector obtained by setting x's i-th bit to 1. Whenever $i \notin A$, then x_i will not be used in $\chi_A(x)$ and we have $\chi_A(x^{i:1}) = \chi_A(x^{i:0})$. And whenever $i \in A$, then $x^{i:1}$ will have exactly one more 1-bit than $x^{i:0}$, so $\chi_A(x^{i:1}) = -\chi_A(x^{i:0})$ and $\chi_A(x^{i:1}) - \chi_A(x^{i:0}) = -2\chi_A(x^{i:0})$. An analogous analysis holds for B. We will consider the quantity $h(x^{i:1}) - h(x^{i:0})$. Notice that

$$h(x^{i:1}) - h(x^{i:0}) = 2 + (\chi_A(x^{i:1}) - \chi_A(x^{i:0})) + (\chi_B(x^{i:1}) - \chi_B(x^{i:0}))$$

If A and B are disjoint, we need to prove that h is monotone. We prove the following: for any $i \in [n]$ and for any vector x, $h(x^{i:1}) - h(x^{i:0}) \geq 0$. This implies that h is monotone: given any $y \leq y'$, we can change 0-bits of y that are 1-bits in y', one by one, until we get y', and in each step we are guaranteed to not decrease h. If $i \notin A \cup B$, then $h(x^{i:1}) - h(x^{i:0}) = 2$ (the other terms cancel out because of what was stated above). If $i \in A \setminus B$, then $h(x^{i:1}) - h(x^{i:0}) = 2 - 2\chi_A(x^{i:0}) \geq 0$ because $\chi_A(x^{i:0}) \leq 1$. And if $i \in B \setminus A$, then $h(x^{i:1}) - h(x^{i:0}) = 2 - 2\chi_B(x^{i:0}) \geq 0$. Since A and B are disjoint, these are the only possibilities, so h is monotone.

If A and B are not disjoint, then take $i \in A \cap B$. We will prove that, for a lot of vectors whose i-th component is 0, changing that component to 1 decreases h. This will mean that h is far from being monotone. Formally, let $T = \{(x^{i:0}, x^{i:1}) : x \in \{0,1\}^n\}$. Clearly, T is a partition of $\{0,1\}^n$ into 2^{n-1} pairs of vectors (each vector is paired with the vector obtained by changing the i-th component). We prove that, for at least 1/4 of the pairs in T, $h(x^{i:1}) - h(x^{i:0}) < 0$. But notice that, since $i \in A \cap B$, this is equivalent to proving that $2 - 2\chi_A(x^{i:0}) - 2\chi_B(x^{i:0}) < 0$. So we will prove that $\chi_A(x^{i:0}) = \chi_B(x^{i:0}) = 1$ for at least 1/4 of the pairs in T.

To prove this, we consider several cases. Without loss of generality, assume $|A| \leq |B|$. If $A = B = \{i\}$, then all of the pairs in T satisfy $\chi_A(x^{i:0}) = \chi_B(x^{i:0}) = 1$. If $A = \{i\} \subsetneq B$, then $\chi_A(x^{i:0}) = 1$ for all x and $\chi_B(x^{i:0}) = 1$ for half of the pairs: $B \setminus \{i\} \neq \emptyset$, so half of the pairs will have components in $B \setminus \{i\}$ that add up to an even number (so $\chi_B(x^{i:0}) = 1$) and the other half to an odd number (so $\chi_B(x^{i:0}) = -1$). So we get a 1/2 fraction in this case. If $\{i\} \subsetneq A = B$, then half of the pairs in $A \setminus \{i\}$ will have $\chi_A(x^{i:0}) = \chi_B(x^{i:0}) = 1$ (the other half will have $\chi_A(x^{i:0}) = \chi_B(x^{i:0}) = -1$), so we get a 1/2 fraction in this case. Finally, suppose that $\{i\} \subsetneq A$ and $B \setminus A \neq \emptyset$. Half of the pairs satisfy $\chi_A(x^{i:0}) = 1$, and, among these, half satisfy $\chi_B(x^{i:0}) = 1$ (because $A \setminus \{i\}$ and $B \setminus A$ are nonempty), so we get 1/4 in this case.

We have proved that, for at least 1/4 of the pairs in T, $h(x^{i:1}) - h(x^{i:0}) < 0$. To obtain a monotone function from h, at least one of the vectors of each such pair has to be changed. Since there are 2^{n-1} pairs, then at least 1/8 of the inputs have to be changed to obtain a monotone function. Thus, h is 1/8-far from monotone.

4.3 Testing Decision Tree Size

Recall that a decision tree (for a function) is a rooted binary tree where each non-leaf node is labelled with a component (say, x_i, for some i), each non-leaf has exactly two out-edges (one of them labelled with 1 and the other labelled with a 0), and each leaf is labelled with an output. Given x, the tree computes some output for x in the obvious way (following the path determined

by x). Thus, every decision tree computes a function. Also, every function is computed by some decision tree. In this example, we will consider the property of being computable by a decision tree of small size. By size we mean the number of non-leaf nodes of the tree. Let Parity be the function $\text{Parity}(x) = x_1 \oplus ... \oplus x_n$. We will use the following lemma from [1]:

Lemma 5 *Let $0 < \delta < 1$. Then Parity is $(1-\delta)/2$-far from functions computable by a decision tree of size at most $\delta 2^n$.*

Proof Let T be a decision tree of size at most $\delta 2^n$ and let q be the function computed by this tree. Let L_1 be the set of leaves of T such that not all components are queried on the path from the root to the leaf. Let L_2 be the other leaves, that is, those where the path to the leaf includes all components. For $i \in \{1,2\}$, let S_i be the set of all $x \in \{0,1\}^n$ such that a leaf in L_i is reached when taking the path determined by x. The leaves in L_2 are only associated to a single input (because all components are queried in the path to the leaf). Since $|L_2| \leq \delta 2^n$, then $|S_2| \leq \delta 2^n$, so $|S_1| \geq (1-\delta)2^n$. For any leaf in L_1, there is at least one component not queried in the path to the leaf, so Parity will differ from q on at least half of the values associated to this leaf (notice that q must be constant on the set of all inputs arriving to that leaf). Summing over all leafs in L_1, we get that Parity differs from q in at least $\frac{|S_1|}{2} \geq \frac{(1-\delta)2^n}{2}$ of the inputs.

Definition Let $\Delta(A,B)$ be the Hamming distance between A and B (it may also be seen as the size of the symmetric difference). Gap equality with parameter t, or $GEQ(t)$, consists of the promise problem of deciding whether $A = B$ ($\Delta(A,B) = 0$) or $\Delta(A,B) = t$. It can be proved that, if $t = \Omega(n)$ is even, the one-sided error communication complexity, in which the protocol may only make mistakes when $A = B$, is $\Omega(n)$. Notice that t being even is necessary for the problem to be hard: if t is odd, then Alice can send $|A|$, Bob can send $|B|$, and since $\Delta(A,B) = |A|+|B|-2|A \cap B|$, they can obtain the parity of $\Delta(A,B)$ from this.

Theorem 6 *Let P be the property of having a decision tree of size at most k. Then the query complexity of testing P with one-sided error testers is $\Omega(k)$.*

Proof Take $s = 32/15k$ as the size of the inputs given to Alice and Bob, and suppose that s is a power of 2, say $s = 2^{n-1}$. Fix a bijection between the sets $\{x \in \{0,1\}^n \mid \text{Parity}(x) = 0\}$ and $[s]$ (both have size 2^{n-1}). This bijection associates to each x such that $\text{Parity}(x) = 0$ an index so that it makes sense to say A_x or B_x. We do a reduction from $GEQ(s/8)$. Alice and Bob take:

$$f_A(x) := \begin{cases} A_x & \text{if Parity}(x) = 0 \\ 0 & \text{if Parity}(x) = 1 \end{cases} \qquad g_B(x) = \begin{cases} B_x & \text{if Parity}(x) = 0 \\ 1 & \text{if Parity}(x) = 1 \end{cases}$$

They test $h := f_A \oplus g_B$. Let us see that, if $A = B$, then h is far from functions with small decision tree size (h will be Parity in this case) and that, if $\Delta(A,B) = s/8$, then it has a decision tree of size at most k.

If $A = B$ then $A_x = B_x$ for all x such that $\text{Parity}(x) = 0$. Therefore, if $\text{Parity}(x) = 0$, $h(x) = 0$. Also, if $\text{Parity}(x) = 1$, then $h(x) = 1$ (this happens always, regardless of what A and B are). Thus, h is Parity, which by the lemma (with $\delta = 15/16$) is $1/32$-far from functions with decision tree size at most $\frac{15}{16}2^n = 15/32s = k$.

If $\Delta(A,B) = s/8$: consider the complete decision tree where at the i-th level x_i is queried. This tree has size $2^n - 1$. Let x be a vector with $\text{Parity}(x) = 0$ such that $A_x \neq B_x$. There are $s/8$ of such vectors, and they satisfy $h(x) = 1$. Consider the leaf l that is associated to x. The sibling is a

leaf associated to the vector with the n-th component changed, so that vector has Parity 1 and the output associated to that leaf is also 1. Thus, we can replace the parent of l (a node that is labelled with x_n) with a leaf whose output is 1. This reduces the number of non-leaf nodes by 1. Because we can do this for all x such that $A_x \neq B_x$, we obtain a tree of size $2^n - 1 - s/8 < \frac{15}{16}2^n = k$.

Notice that the tester may only make mistakes when h is far from having a decision tree of size k. This translates to a communication protocol in which Alice and Bob may only make mistakes when $A = B$, as desired.

5 New Directions

5.1 Other Work

The paper that introduces the connection between property testing and communication complexity [2] also gives some other reductions and lower bounds. It provides an alternative proof of some known lower bounds for two sided-error adaptive testers of properties concerning concise representations: decision tree size, DNFs, branching programs, and boolean formulas. It also proves new lower bounds on the query complexity of testing s-sparse polynomials.

A later paper by Goldreich [5] suggests thinking of the reduction in a more direct fashion (instead of thinking on f_A and g_B, construct the function directly from A and B). His approach can be applied if a property is seen as a subset of a set of the form $\{0,1\}^l$. He shows an application of his method on codeword testing.

After the excellent initial results, involving both improved lower bounds and simpler proofs, there seemed to be a halt to the progress. Bhrushnundi et al [3] introduce a possible intermediate step for the reductions. The paper proposes to go from communication complexity to parity decision trees (a generalization of decision trees, where nodes are now labelled with linear functions), and from there to property testing. This could lead to new lower bounds, because the resulting problem now consists of proving a lower bound on the randomized parity decision tree complexity of a function. Using this method, they characterize properties of linear functions testable with constant queries and prove that testing affine isomorphism (to IP) requires $\Omega(n^2)$ queries, thus obtaining a tight bound (an $O(n^2)$ bound is known).

Tell [9] generalized this idea to linear access models. Here, the model is viewed using randomized oracle machines (that can do linear queries) instead of trees, and the field may be any finite field (instead of just the field of two elements). Besides considering linear access models just as an intermediate step, he also proposed proving property testing lower bounds directly from linear access lower bounds.

5.2 Open problems

There are many open problems in property testing. One consists of closing the gap between the upper and lower bounds for property testing. The following are the lower bounds known, versus the upper bounds known, for the examples we discussed:

- k-linearity: $\Omega(k)$ versus $O(k \log k)$. This is only for adaptive testers. For non-adaptive, a tight $\Theta(k \log k)$ bound is known.
- Fourier degree at most k: $\Omega(k)$ versus $2^{O(k)}$.

- Size s decision tree of size at most k (two-sided error): $\Omega(\log k)$ versus $O(k)$. Note that our bound was for one-sided error.

Another important open problem is to obtain a lower bound on the query complexity of testing monotonicity on $\{0,1\}$. The bound we proved says nothing about this case. There has been recent progress in showing $O(n^{5/6})$ testers, but the best lower bound known is $\Omega(\log n)$ [4].

Finally, property testing can also be defined on graphs (the tester queries edges, and the distance is defined in terms of edges). There are no known proofs of lower bounds for testing graph properties that use communication complexity reductions. This is quite surprising. Why is this reduction method so powerful for functions, but seemingly useless when we try to apply it to graphs? Or is there some clever reduction that does provide lower bounds for some graph property?

References

[1] E. Blais. Testing properties of boolean functions. PhD thesis (2012).

[2] E. Blais, J. Brody, K. Matulef. Property testing lower bounds via communication complexity. CCC (2011).

[3] A. Brushnundi, S. Chakraborty, R. Kulkarni. Property testing bounds for linear and quadratic functions via parity decision trees. ECCC (2013).

[4] X. Chen, R. Servedio, L. Yang. New algorithms and lower bounds for monotonicity testing. FOCS (2014).

[5] O. Goldreich. On the communication complexity methodology for proving lower bounds on the query complexity of property testing. ECCC (2013).

[6] P. Hajnal. An $\Omega(n^{4/3})$ lower bound on the randomized complexity of graph properties. Combinatorica (1991).

[7] R. L. Rivest, J. Vuillemin. On recognizing graph properties from adjacency matrices. Theoretical Computer Science (1976).

[8] D. Ron. Property testing. Handbook of Randomization (2000).

[9] R. Tell. Deconstructions of Reductions from Communication Complexity to Property Testing using Generalized Parity Decision Trees. ECCC (2014).