Lecture Notes on Testing Dictatorships, Juntas, and Monomials

Oded Goldreich[*]

May 21, 2016

Summary: We consider testing three basic properties of Boolean functions of the form $f : \{0,1\}^\ell \to \{0,1\}$:

1. Dictatorship: The case where the value of f depends on a single Boolean variable (i.e., $f(x) = x_i \oplus \sigma$ for some $i \in [\ell]$ and $\sigma \in \{0,1\}$).
2. Junta (of size k): The case where the value of f depends on at most k Boolean variables (i.e., $f(x) = f'(x_I)$ for some k-subset $I \subset [\ell]$ and $f' : \{0,1\}^k \to \{0,1\}$).
3. Monomial (of size k): The case where the value of f is the conjunction of exactly k Boolean variables (i.e., $f(x) = \wedge_{i \in I}(x_i \oplus \sigma_i)$ for some k-subset $I \subseteq [\ell]$ and $\sigma_1, ..., \sigma_\ell \in \{0,1\}$).

We present two different testers for dictatorship, where one generalizes to testing k-Juntas and the other generalizes to testing k-Monomials.

These notes are based on the works of Parnas, Ron, and Samorodnitsky [29][1] and Fischer, Kindler, Ron, Safra, and Samorodnitsky [15].

1 Introduction

Boolean functions $f : \{0,1\}^\ell \to \{0,1\}$ that depend on very few of their Boolean variables arise in many applications. Such variables are called *relevant variables*, and they arise in the study of natural phenomena, where there are numerous variables (or attributes) that describe the phenomena but only few of them are actually relevant.

Typically, one does not know *a priori* which of the ℓ variables are relevant, and a natural task is to try to find this out. But before setting out to find the relevant variables, one may want to find out how many variables are actually relevant. Furthermore, in some cases (as shown in the next lecture), just knowing a good upper bound on the number of influential variables is valuable.

Assuming that there are $k \le \ell$ influential variables, finding the set of influential variables requires making $\Omega(2^k + k \log \ell)$ queries to the function, because the number of functions $f : \{0,1\}^\ell \to \{0,1\}$ that have k influential variables is of the order of $\binom{\ell}{k} \cdot 2^{2^k}$, minus a lower order term of $\binom{\ell}{k-1} \cdot 2^{2^{k-1}}$. Our goal is to test whether f has k influential variables (or is ϵ-far from having this property) using

[*]Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.

[1]See discussions regarding the relation to the work of Bellare, Goldreich, and Sudan [4].

only poly(k/ϵ) queries; in particular, the complexity we seek is independent of ℓ, which is especially valuable when ℓ is very large compared to k.

Functions having at most k influential variables will be called *k-juntas*, and in case of $k = 1$ they will be called *dictatorships*. We shall start with the latter case; in Section 2 we present a tester of dictatorships, which views dictatorships as linear functions that depend on one variable. Hence, this tester will first check whether the function is linear, and then check (via self-correction) whether this linear function is a dictatorship. This approach is abstracted in Section 2.3, which is highly recommended.

Section 3 deals with the more general problem of testing whether a function is a k-junta, where $k \geq 1$ is a parameter that is given to the tester. This tester uses different ideas, and thus it yields an alternative tester for dictatorship. (The analysis of this tester is more complex than the analysis of the tester for dictatorship presented in Section 2.)

Teaching note: We suggest leaving the overview section that discusses testing monomials (i.e., Section 2.2) for advanced independent reading.

2 Testing dictatorship via self-correction

Here we consider testing two related properties of Boolean functions $f : \{0,1\}^\ell \to \{0,1\}$, called *dictatorship* and *monotone dictatorship*. First, we note that the object being tested is of size $n = 2^\ell$, and so query complexity that is logarithmic in ℓ (which can be obtained via proper learning (see first lecture))[2] is considered sub-linear. Still, we shall seek testers of lower complexity; specifically, we seek complexity that is independent of the size of the object.

Definition 1 (dictatorship and monotone dictatorship): *A function $f : \{0,1\}^\ell \to \{0,1\}$ is called a* monotone dictatorship *if for some $i \in [\ell]$ it holds that $f(x) = x_i$. It is called a* dictatorship *if for some $i \in [\ell]$ and $\sigma \in \{0,1\}$ it holds that $f(x) = x_i \oplus \sigma$.*

Note that f is a dictatorship if and only if either f or $f \oplus 1$ is a monotone dictatorship. Hence, the set of dictatorships is the union of Π and $\{f : f \oplus 1 \in \Pi\}$, where Π is the set of monotone dictatorships. Using the closure of property testing under unions (see the first lecture), we may reduce testing dictatorship to testing monotone dictatorship.[3] Thus, we shall focus on the latter task.

A detour: dictatorship and the Long Code. The Long Code, which was introduced in [4] and plays a pivotal role in many PCP constructions (see, e.g., [4, 22, 23, 31, 14, 24, 25, 28])[4], encodes k-bit long strings by 2^{2^k}-bit strings such that $x \in \{0,1\}^k$ is encoded by the sequence of the evaluations

[2]This uses the fact that there are only 2ℓ different dictatorship functions.

[3]In fact, we also use the fact that testing $\{f : f \oplus 1 \in \Pi\}$ reduces to testing Π. Indeed, this holds for any property Π of Boolean functions.

[4]The Long Code is pivotal especially in PCP constructions aimed at optimizing parameters of the query complexity, which are often motivated by the desire to obtain tight inapproximability results. We refer to this line of research as the "second generation" of PCP constructions, which followed the "first generation" that culminated in the establishing of the PCP Theorem [3, 2]. In contrast, the Long Code is not used (or need not be used) in works of the "third generation" that focus on other considerations such as proof length (e.g., [19, 5]), combinatorial constructions (e.g., [13, 11]), and lower error via few multi-valued queries (e.g., [27, 12]).

of all $n = 2^{2^k}$ Boolean functions $g : \{0,1\}^k \to \{0,1\}$ at x. That is, the g^{th} location of the codeword $C(x) \in \{0,1\}^n$ equals $g(x)$. Now, look at $f_x = C(x)$ as a function from $\{0,1\}^{2^k}$ to $\{0,1\}$ such that $f_x(\langle g \rangle) = g(x)$, where $\langle g \rangle \in \{0,1\}^{2^k}$ denotes the truth-table of $g : \{0,1\}^k \to \{0,1\}$ and $\langle g \rangle_x$ denotes the bit corresponding to location x in $\langle g \rangle$ (which means that $\langle g \rangle_x$ equals $g(x)$). Note that the 2^k (bit) locations in $\langle g \rangle$ correspond to k-bit strings. Then, the function $f_x : \{0,1\}^{2^k} \to \{0,1\}$ is a monotone dictatorship, since its value at any input $\langle g \rangle$ equals $\langle g \rangle_x$ (i.e., $f_x(\langle g \rangle) = g(x) = \langle g \rangle_x$ for every $\langle g \rangle$). Hence, the Long Code (encoding k-bit strings) is the set of monotone dictatorship functions from $\{0,1\}^{2^k}$ to $\{0,1\}$, which means that the Long Code corresponds to the case that ℓ is a power of two.

2.1 The tester

One key observation towards testing monotone dictatorships is that these functions are linear; that is, they are parity functions (where each parity function is the exclusive-or of a subset of its Boolean variables). Hence, we may first test whether the input function $f : \{0,1\}^\ell \to \{0,1\}$ is linear (or rather close to linear), and rejects otherwise. Assuming that f is close to the linear function f', we shall test whether f' is a (monotone) dictatorship, by relying on the following dichotomy, where $x \wedge y$ denotes the bit-by-bit AND of the ℓ-bit strings x and y:

- On the one hand, if f' is a monotone dictatorship, then

$$\mathbf{Pr}_{x,y \in \{0,1\}^\ell}[f'(x) \wedge f'(y) = f'(x \wedge y)] = 1. \tag{1}$$

  This holds since if $f'(x) = x_i$, then $f'(y) = y_i$ and $f'(x \wedge y) = x_i \wedge y_i$.

- On the other hand, if $f'(x) = \oplus_{i \in I} x_i$ for $|I| > 1$, then

$$\begin{aligned}
&\mathbf{Pr}_{x,y \in \{0,1\}^\ell}[f'(x) \wedge f'(y) = f'(x \wedge y)] \\
&\quad = \mathbf{Pr}_{x,y \in \{0,1\}^\ell}[(\oplus_{i \in I} x_i) \wedge (\oplus_{i \in I} y_i) = \oplus_{i \in I}(x_i \wedge y_i)] \\
&\quad = \mathbf{Pr}_{x,y \in \{0,1\}^\ell}[\oplus_{i,j \in I}(x_i \wedge y_j) = \oplus_{i \in I}(x_i \wedge y_i)] \qquad (2)
\end{aligned}$$

  Our aim is to show that Eq. (2) is strictly smaller than one. It will be instructive to analyze this expression by moving to the arithmetics of the two-element field. Hence, Eq. (2) can be written as

$$\mathbf{Pr}_{x,y \in \{0,1\}^\ell}\left[\sum_{i,j \in I : i \neq j} x_i \cdot y_j = 0\right] \tag{3}$$

  Observing that the expression in Eq. (3) is a non-zero polynomial of degree two, we conclude that it equals zero with probability at most 3/4 (see Exercise 1). It follows that in this case

$$\mathbf{Pr}_{x,y \in \{0,1\}^\ell}[f'(x) \wedge f'(y) = f'(x \wedge y)] \leq 3/4. \tag{4}$$

The gap between Eq. (1) and Eq. (4) should allow us to distinguish these two cases. However, there is also a third case; that is, the case that f' is the all-zero function. This pathological case can be discarded by checking that $f'(1^\ell) = 1$, and rejecting otherwise.

Of course, we have no access to f', but assuming that f' is close to f, we can obtain the value of f' at any desired point by using self-correction (on f) as follows. When seeking the value of

$f'(z)$, we select uniformly at random $r \in \{0,1\}^\ell$, query f at r and at $r \oplus z$, and use the value $f(r) \oplus f(r \oplus z)$. Indeed, the value $f(r) \oplus f(r \oplus z)$ can be thought of as a random vote regarding the value of $f'(z)$. If f' is ϵ-close to f, then this vote equals the value $f'(z)$ with probability at least $\mathbf{Pr}_r[f'(r) = f(r) \,\&\, f'(r \oplus z) = f(r \oplus z)] \geq 1 - 2\epsilon$, since $f'(z) = f'(r) \oplus f'(r \oplus z)$ (by linearity of f').

This discussion leads to a natural tester for monotone dictatorship, which first checks whether f is linear and if so checks that the linear function f' that is close to f is a monotone dictatorship. We check that f' is a dictatorship by checking that $f'(x \wedge y) = f'(x) \wedge f'(y)$ for uniformly distributed $x, y \in \{0,1\}^\ell$ and that $f'(1^\ell) = 1$, where in both cases we use self-correction (for the values at $x \wedge y$ and 1^ℓ).[5] Indeed, in Step 2 (below), the random strings r and s are used for self-correction of the values at $x \wedge y$ and 1^ℓ, respectively.

Below, we assume for simplicity that $\epsilon \leq 0.01$. This assumption can be made, without loss of generality, by redefining $\epsilon \leftarrow \min(\epsilon, 0.01)$. (It follows that any function f is ϵ-close to *at most one* linear function, since the linear functions are at distance $1/2$ from one another whereas $\epsilon < 0.25$.)[6]

Algorithm 2 (testing monotone dictatorship): *On input $n = 2^\ell$ and $\epsilon \in (0, 0.01]$, when given oracle access to a function $f : \{0,1\}^\ell \to \{0,1\}$, the tester proceeds as follows.*

1. *Invokes the linearity tester on input f, while setting the proximity parameter to ϵ. If the linearity test rejected, then halt rejecting.*

   Recall that the known linearity tester makes $O(1/\epsilon)$ queries to f.

2. *Repeat the following check for $O(1/\epsilon)$ times.*[7]

   (a) *Select $x, y, r, s \in \{0,1\}^\ell$ uniformly at random.*

   (b) *Query f at the points x, y, r, s as well as at $r \oplus (x \wedge y)$ and $s \oplus 1^\ell$.*

   (c) *If $f(x) \wedge f(y) \neq f(r) \oplus f(r \oplus (x \wedge y))$, then halt rejecting.*

   (d) *If $f(s) \oplus f(s \oplus 1^\ell) = 0$, then halt rejecting.*

   If none of the iterations rejected, then halt accepting.

(Actually, in Step 2d, we can use r instead of s, which means that we can re-use the same randomization in both invocations of the self-correction.)[8] Recalling that linearity testing is performed by invoking a three-query proximity-oblivious tester for $O(1/\epsilon)$ times, it is begging to consider the following proximity-oblivious tester (POT) instead of Algorithm 2.

Algorithm 3 (POT for monotone dictatorship): *On input $n = 2^\ell$ and oracle access to a function $f : \{0,1\}^\ell \to \{0,1\}$, the tester proceeds as follows.*

[5] Values at these points require self-correction, since these points are not uniformly distributed in $\{0,1\}^\ell$. In contrast, no self-correction is required for the values at the uniformly distributed points x and y. See Section 2.3 for a general discussion of the self-correction technique.

[6] **Advanced comment:** The uniqueness of the linear function that is ϵ-close to f is not used explicitly in the analysis, but the analysis does require that $\epsilon < 1/16$ (see proof of Lemma 4).

[7] Step 2c is self-correction form of the test $f(x) \wedge f(y) \overset{?}{=} f(x \wedge y)$, whereas Step 2d is self-correction form of the test $f(1^\ell) \overset{?}{=} 1$.

[8] This takes advantage of the fact that, in the analysis, for each possible f we rely only on *one* of the three rejection options.

1. *Invokes the three-query proximity-oblivious tester* (of linear detection probability) *for linearity.*[9] *If the linearity test rejected, then halt rejecting.*

2. *Check closure to bit-by-bit conjunction.*

   (a) *Select $x, y, r \in \{0,1\}^\ell$ uniformly at random.*

   (b) *Query f at the points x, y, r and $r \oplus (x \wedge y)$.*

   (c) *If $f(x) \wedge f(y) \neq f(r) \oplus f(r \oplus (x \wedge y))$, then reject.*

3. *Check that f is not the all-zero function.*

   (a) *Select $s \in \{0,1\}^\ell$ uniformly at random.*

   (b) *Query f at the points s and $s \oplus 1^\ell$.*

   (c) *If $f(s) \oplus f(s \oplus 1^\ell) = 0$, then reject.*

If none of the foregoing steps rejected, then halt accepting.

As shown next, Algorithm 3 is a nine-query POT with linear detection probability. The same holds for a four-query algorithm that performs one of the three steps at random (i.e., each step is performed with probability 1/3).[10]

Theorem 4 (analysis of Algorithm 3): *Algorithm 3 is a one-sided error proximity oblivious tester for monotone dictatorship with rejection probability $\varrho(\delta) = \Omega(\delta)$.*

Proof: The proof merely details the foregoing discussion. First, suppose that $f : \{0,1\}^\ell \to \{0,1\}$ is a monotone dictatorship, and let $i \in [\ell]$ such that $f(x) = x_i$. Then, f is linear, and so Step 1 never rejects. (Hence, $f(r) \oplus f(r \oplus z) = f(z)$ for all r, z). Furthermore, $f(x) \wedge f(y) = x_i \wedge y_i = f(x \wedge y)$, which implies that Step 2 never rejects. Lastly, in this case, $f(1^\ell) = 1$, which implies that Step 3 never rejects. It follows that Algorithm 3 always accepts f.

Now, suppose that f is at distance $\delta > 0$ from being a monotone dictatorship. Letting $\delta' = \min(0.9\delta, 0.01)$, we consider two cases.[11]

1. If f is δ'-far from being linear, then Step 1 rejects with probability $\Omega(\delta') = \Omega(\delta)$.

2. If f is δ'-close to being linear, then it is δ'-close to some linear function, denoted f'. Note that f' cannot be a dictatorship function, since this would mean that f is δ'-close to being a monotone whereas $\delta' < \delta$.

   We first note that if f' is the all-zero function, then Step 3 rejects with probability greater than $1 - 2 \cdot 0.01 = \Omega(\delta)$, since

$$\begin{aligned}
\mathbf{Pr}_s[f(s) \oplus f(s \oplus 1^\ell) = 0] &\geq \mathbf{Pr}_s[f(s) = f'(s) \ \& \ f(s \oplus 1^\ell) = f'(s \oplus 1^\ell)] \\
&\geq 1 - \mathbf{Pr}_s[f(s) \neq f'(s)] - \mathbf{Pr}_s[f(s \oplus 1^\ell) \neq f'(s \oplus 1^\ell)] \\
&\geq 1 - 2 \cdot \delta'.
\end{aligned}$$

[9]Such a POT, taken from [8], was presented in a prior lecture.

[10]See Exercise 2.

[11]**Advanced comment:** Any choice of $\delta' \leq 0.06$ such that is $\delta' \in [\Omega(\delta), \delta)$ will do. In fact, 0.06 can be replaced by any constant in $(0, 1/16)$.

Hence, we are left with the case that $f'(x) = \oplus_{i\in I} x_i$, where $|I| \geq 2$. Relying on the hypothesis that f is 0.01-close to f' and using $f'(r) \oplus f'(r \oplus (x \wedge y)) = f'(x \wedge y)$, we observe that the probability that Step 2 rejects equals

$$\begin{aligned}
&\mathbf{Pr}_{x,y,r}[f(x) \wedge f(y) \neq f(r) \oplus f(r \oplus (x \wedge y))] \\
&\quad \geq \mathbf{Pr}_{x,y,r}[f'(x) \wedge f'(y) \neq f'(r) \oplus f'(r \oplus (x \wedge y))] \\
&\qquad - \mathbf{Pr}_{x,y,r}[f(x) \neq f'(x) \vee f(y) \neq f'(y) \vee f(r) \neq f'(r) \vee f(r \oplus (x \wedge y)) \neq f'(r \oplus (x \wedge y))] \\
&\quad \geq \mathbf{Pr}_{x,y}[f'(x) \wedge f'(y) \neq f'(x \wedge y)] - 4 \cdot \mathbf{Pr}_z[f(z) \neq f'(z)] \\
&\quad \geq 0.25 - 4 \cdot 0.01
\end{aligned}$$

where the second inequality uses a union bound as well as $f'(r) \oplus f'(r \oplus (x \wedge y)) = f'(x \wedge y)$, and the last inequality is due to Eq. (4). Hence, in this case, Step 2 rejects with probability greater than $0.2 = \Omega(\delta)$.

To summarize, in each of the two cases, the algorithm rejects with probability $\Omega(\delta)$, and the theorem follows. ■

Digest. Note that self-correction was applied for obtaining the values of $f'(x \wedge y)$ and $f'(1^\ell)$, but not for obtaining $f'(x)$ and $f'(y)$, where x and y were uniformly distributed in $\{0,1\}^\ell$. Indeed, there is no need to apply self-correction when seeking the value of f' at a uniformly distributed point. In contrast, the points 1^ℓ and $x \wedge y$ are *not* uniformly distributed: the point 1^ℓ is fixed, whereas $x \wedge y$ is selected from a distribution of ℓ-bit long strings in which each bit is set to 1 with probability 1/4 (rather than 1/2), independently of all other bits. For further discussion of the self-correction paradigm, see Section 2.3.

2.2 Testing monomials

The ideas that underly the tests of (monotone) dictatorship can be extended towards testing the set of functions that are (monotone) k-monomials, for any $k \geq 1$.

Definition 5 (monomial and monotone monomial): *A function $f : \{0,1\}^\ell \to \{0,1\}$ is called a* k-monomial *if for some k-subset $I \subseteq [\ell]$ and $\sigma = \sigma_1 \cdots \sigma_\ell \in \{0,1\}^\ell$ it holds $f(x) = \wedge_{i\in I}(x_i \oplus \sigma_i)$. It is called a* monotone k-monomial *if $\sigma = 0^\ell$.*

Indeed, the definitions of (regular and monotone) dictatorship coincide with the notions of (regular and monotone) 1-monomials. (In particular, f is a dictatorship if and only if either f or $f'(x) = f(x \oplus 1^\ell) = f(x) \oplus 1$ is a monotone dictatorship).

Teaching note: Alternative procedures for testing (regular and monotone) monomials are presented in the next lecture (on testing via implicit sampling). These alternative procedures are obtained as a simple application of a general paradigm, as opposed to the the direct approach, which is only outlined here. In light of these facts, the reader may skip the current section and procede directly to Section 2.3.

2.2.1 A reduction to the monotone case

Note that f is a k-monomial if and only if for some $\sigma \in \{0,1\}^\ell$ the function $f_\sigma(x) = f(x \oplus \sigma)$ is a monotone k-monomial. Actually, it suffices to consider only σ's such that $f(\sigma \oplus 1^\ell) = 1$, since if f_σ is a monotone monomial, then $f_\sigma(1^\ell) = 1$ must hold. This suggests the following reduction of testing k-monomials to testing monotone k-monomials.

Algorithm 6 (reducing testing monomials to the monotone case): *Given parameters k and ϵ and oracle access to a function $f : \{0,1\}^\ell \to \{0,1\}$, we proceed as follows if $\epsilon < 2^{-k+2}$.*

1. *Select uniformly a random $O(2^k)$-subset of $\{0,1\}^\ell$, denoted S, and for each $\sigma \in S$ query f at $\sigma \oplus 1^\ell$. If for every $\sigma \in S$ it holds that $f(\sigma \oplus 1^\ell) = 0$, then reject. Otherwise, pick any $\sigma \in S$ such that $f(\sigma \oplus 1^\ell) = 1$, and proceed to Step 2.*

2. *Invoke the ϵ-tester for monotone k-monomials, while proving it with oracle access to f' such that $f'(x) = f(x \oplus \sigma)$.*

If $\epsilon \geq 2^{-k+2}$, then use $O(1/\epsilon)$ samples in order to distinguish the case of $|f^{-1}(1)| \leq 0.25\epsilon \cdot 2^\ell$ from the case of $|f^{-1}(1)| \geq 0.75\epsilon \cdot 2^\ell$. Accept in the first case and reject in the second case. (That is, reject if less than a 0.5ϵ fraction of the sample evaluates to 1, and accept otherwise.)

Note that the restriction of the actual reduction to the case that $\epsilon < 2^{-k+2}$ guarantees that the (additive) overhead of the reduction, which is $O(2^k)$, is upper-bounded by $O(1/\epsilon)$. On the other hand, when $\epsilon \geq 2^{-k+2}$, testing is reduced to estimating the density of $f^{-1}(1)$, while relying on the facts that any k-monomial is at distance exactly 2^{-k} from the all-zero function. In both cases, the reduction yields a tester with two-sided error (even when using a tester of one-sided error for the monotone case).

Theorem 7 (analysis of Algorithm 6): *If the ϵ-tester for monotone k-monomials invoked in Step 2 has error probability at most $1/4$, then Algorithm 6 constitutes a tester for k-monomials.*

Proof: We start with the (main) case of $\epsilon < 2^{-k+2}$. Note that if $|f^{-1}(1)| < 2^{\ell-k}$, then f cannot be a k-monomial, and it is OK to reject it. Otherwise (i.e., $|f^{-1}(1)| \geq 2^{\ell-k}$), with probability at least 0.9, Step 1 finds σ such that $f(\sigma \oplus 1^\ell) = 1$. Now, if f is a k-monomial, then f' as defined in Step 2 (i.e., $f'(x) = f(x \oplus \sigma)$) is a monotone k-monomial, since all strings in $f^{-1}(1)$ agree on the values of the bits in location I, where I denotes the indices of the variables on which f depends.[12] Thus, any k-monomial is accepted by the algorithm with probability at least $0.9 \cdot 0.75 > 2/3$.

On the other hand, if f is ϵ-far from being a k-monomial, then either Step 1 rejects or (as shown next) f' is (also) ϵ-far from being a (monotone) k-monomial, and Step 2 will reject it with probability at least $3/4 > 2/3$. To see that f' is ϵ-far from being a k-monomial (let alone ϵ-far from being a monotone k-monomial), we consider a k-monomial g' that is supposedly ϵ-close to f' and derive a contradiction by considering the function g such that $g(x) = g'(x \oplus \sigma)$, where σ is as in Step 2 (i.e., $f'(x) = f(x \oplus \sigma)$). Specifically, g maintains the k-monomial property of g', whereas $\delta(f,g) = \delta(f',g') \leq \epsilon$ (since $f(x) = f'(x \oplus \sigma)$).

[12] To see this claim, let $f(x) = \wedge_{i\in I}(x_i \oplus \tau_i)$, for some k-set $I \subseteq [\ell]$ and $\tau \in \{0,1\}^\ell$. Then, $f(\sigma \oplus 1^\ell) = 1$ if and only if $\wedge_{i\in I}(\sigma_i \oplus 1 \oplus \tau_i) = 1$, which holds if and only if $\sigma_i = \tau_i$ for every $i \in I$. Hence, $f'(x) = f(x \oplus \sigma) = f(x \oplus \tau)$ is a monotone monomial.

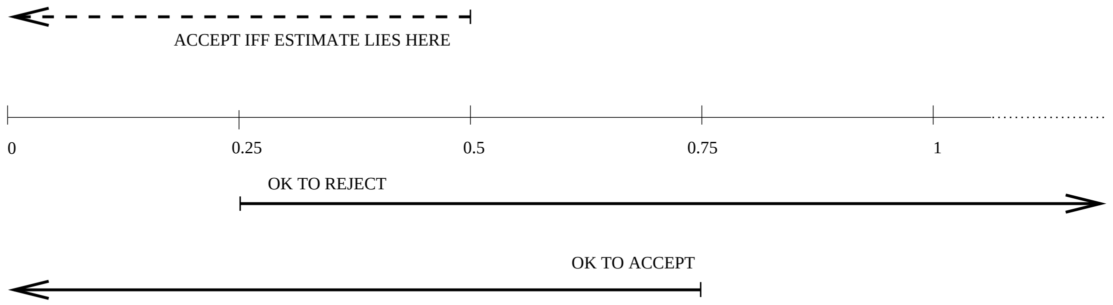

Figure 1: Detail for the proof of Theorem 7. The algorithmic decision is depicted by a dashed arrow that refers to the estimated value (in multiples of $\epsilon 2^\ell$), and the analysis is depicted by solid arrows that refers to the real value of $|f^{-1}(1)|$.

We complete the proof by considering the case of $\epsilon \geq 2^{-k+2}$. In this case, if $|f^{-1}(1)| > 0.25\epsilon \cdot 2^\ell$, which implies $|f^{-1}(1)| > 2^{\ell-k}$, then f is not a k-monomial, and it is OK to reject it. On the other hand, if $|f^{-1}(1)| \leq 0.75\epsilon \cdot 2^\ell$, then f is 0.75ϵ-close to the all-zero function, which is 2^{-k}-close to a k-monomial, and so it is OK to accept f, because f is ϵ-close to a k-monomial (since $0.75\epsilon + 2^{-k} \leq \epsilon$). Indeed, when $|f^{-1}(1)| \in (0.25\epsilon 2^\ell, 0.75\epsilon 2^\ell]$, any decision is fine (see Figure 1). Hence, it suffices to guarantee rejection (w.p. 2/3) when $|f^{-1}(1)| \geq 0.75\epsilon 2^\ell$ and acceptance (w.p. 2/3) when $|f^{-1}(1)| \leq 0.25\epsilon 2^\ell$, as the algorithm does. ■

2.2.2 Testing monotone k-monomials – an overview

We start by interpreting the dictatorship tester in a way that facilitates its generalization. If f is a monotone dictatorship, then $f^{-1}(1)$ is an $(\ell-1)$-dimensional affine subspace (of the ℓ-dimensional space $\{0,1\}^\ell$). Specifically, if $f(x) = x_i$, then this subspace is $\{x \in \{0,1\}^\ell : x_i = 1\}$. In this case, the *linearity tester* could be thought of as testing that $f^{-1}(1)$ is an arbitrary $(\ell-1)$-dimensional subspace, whereas the "conjunction test" verifies that this subspace is an affine translation by 1^ℓ of a linear space that is spanned by $\ell-1$ unit vectors (i.e., vectors of Hamming weight 1).[13]

Now, if f is a monotone k-monomial, then $f^{-1}(1)$ is an $(\ell-k)$-dimensional affine subspace. So the idea is to first test that $f^{-1}(1)$ is an $(\ell-k)$-dimensional affine subspace, and then to test that it is an affine subspace of the right form (i.e., it has the form $\{x \in \{0,1\}^\ell : (\forall i \in I)\ x_i = 1\}$, for some k-subset I). Following are outlines of the treatment of these two tasks.

Testing affine subspaces. Supposed that the alleged affine subspace H is presented by a Boolean function h such that $h(x) = 1$ if and only if $x \in H$. (Indeed, in our application, $h = f$.) We wish to test that H is indeed an affine subspace.

(Actually, we are interested in testing that H has a given dimension, but this extra condition can be checked easily by estimating the density of H in $\{0,1\}^\ell$, since we are willing to have complexity

[13]That is, we requires that this subspace has the form $\left\{1^\ell + \sum_{j\in([\ell]\setminus\{i\})} c_j e_j : c_1, ..., c_\ell \in \{0,1\}\right\}$, where $e_1, ..., e_\ell \in \{0,1\}^\ell$ are the ℓ unit vectors (i.e., vectors of Hamming weight 1).

that is inversely proportional to the designated density (i.e., 2^{-k}).)[14]

This task is related to linearity testing and it was indeed solved in [29] using a tester and an analysis that resembles the standard linearity tester of [8]. Specifically, the tester selects uniformly $x, y \in H$ and $z \in \{0,1\}^\ell$ and checks that $h(x+y+z) = h(z)$ (i.e., that $x+y+z \in H$ if and only if $z \in H$). Indeed, we uniformly sample H by repeatedly sampling $\{0,1\}^\ell$ and checking whether the sampled element is in H.

Note that, for co-dimension $k > 1$, the function h is not affine (e.g., $h(x) = h(y) = 0$, which means $x, y \notin H$, does not determine the value of $h(x+y)$ (i.e., whether $x+y \in H$)). Still, testing affine subspaces can be reduced to testing linearity, providing an alternative to the presentation of [29] (see [20, Sec. 4] or Exercises 8–10).

Testing that an affine subspace is a translation by 1^ℓ of a linear subspace spanned by unit vectors. Suppose that an affine subspace H' is presented by a Boolean function, denoted h', and that we wish to test that H' has the form $\left\{1^\ell + \sum_{i\in[\ell]\setminus I} c_i e_i : c_1, ..., c_\ell \in \{0,1\}\right\}$, where $e_1, ..., e_\ell \in \{0,1\}^\ell$ are unit vectors, and $I \subseteq [\ell]$ is arbitrary. That is, we wish to test that $h'(x) = \wedge_{i\in I} x_i$.

This can be done by picking uniformly $x \in H'$ and $y \in \{0,1\}^\ell$, and checking that $h'(x \wedge y) = h'(y)$ (i.e., $x \wedge y \in H'$ if and only if $y \in H'$). Note that if H' has the form $1^\ell + L$, where L is a linear subspace spanned by the unit vectors $\{e_i : i \in [\ell] \setminus I\}$ for some I, then $h'(z) = \wedge_{i\in I} z_i$ holds for all $z \in \{0,1\}^\ell$ and $h'(x \wedge y) = h'(x) \wedge h'(y)$ holds for all $x, y \in \{0,1\}^\ell$. On the other hand, as shown in [29], if H' is an affine subspace that does not have the foregoing form, then the test fails with probability at least 2^{-k-1}.

However, as in the case of $k = 1$, we do not have access to h' but rather to a Boolean function h that is (very) close to h'. So we need to obtain the value of h' at specific points by querying h at uniformly distributed points. Specifically, the value of h' at z is obtained by uniformly selecting $r, s \in h^{-1}(1)$ and using the value $h(r+s+z)$. In other words, we self-correct h at any desired point by using the value of h at random elements of $h^{-1}(1)$, while actually hoping that these points reside in the affine subspace H'. This hope is likely to materialize when h is $0.01 \cdot 2^{-k}$-close to h'.

The foregoing is indeed related to the conjunction check performed in Step 2 of Algorithm 3, and the test and the analysis in [29] resemble the corresponding parts in Section 2.1. An alternative approach, which essentially rediuces the general case (of any $k \geq 1$) to the special case (of $k = 1$), appears in [20, Sec. 5].

Conclusion. To recap, the overall structure of the resulting tester resembles that of Algorithm 3, with the exception that we perform a density test in order to determine the dimension of the affine subspace. We warn, however, that the analysis is significantly more involved (and the interested reader is referred to [29]). Lastly, we stress that the tester of monotone k-monomials has two-sided error probability, which is due to its estimation of the density of the affine subspace. We wonder whether this is inherent.

Open Problem 8 (one-sided error testers for monomials): *For any $k \geq 2$, is there a one-sided error test for monotone k-monomials with query complexity that only depends on the proximity parameter? Ditto for testing monotone k-monomials.*

[14]Recall that if $\epsilon < 2^{-k+2}$, then $O(2^k) = O(1/\epsilon)$, and otherwise (i.e., for $\epsilon \geq 2^{-k+2}$) we can proceed as in Algorithm 6.

It seems that when the arity of the monomial (i.e., k) is not specified, one-sided testing is possible by modifying the tester of [29] such that the size check is avoided. Indeed, in such a case, one may fail to sample $f^{-1}(1)$ using $O(1/\epsilon)$ random queries, but we can avoid rejection in this case because it occurs with noticeable probability only when the function f is 0.5ϵ-close to the all-zero function, which implies that f is ϵ-close to the monotone ℓ-monomial (provided that $2^{-\ell} \leq 0.5\epsilon$).[15]

2.3 The self-correction paradigm: an abstraction

Recall that self-correction was used in the analysis of the linearity and low-degree tests, but in Section 2.1 we used this paradigm as part of the tester. We now abstract the self-correction paradigm as an algorithmic paradigm (rather than as a tool of analysis).

In general, the self-correction of a function f that is close to a function g is based on a "random self-reduction" feature of g, which is the ability to easily recover the value of g at any fixed z in the g's domain based on the value of g on few uniformly distributed points in g's domain. We stress that each of these points is uniformly distributed in g's domain, but *they are not necessarily independent of one another.*

The foregoing description of random self-reduction is lacking, because, for a fixed function g, nothing prevents the recovery algorithm from just computing $g(z)$. In the context of complexity theory this is avoided by requiring the recovery algorithm to have lower *computational complexity* than any algorithm computing g. In the current context, where the focus is information theoretic (i.e., on the query complexity), we can not use this possibility. Instead, we define random self-reducibility for sets of functions.

Definition 9 (random self-reduction): *Let Π be a set of functions ranging over D. We say that* functions in Π are randomly self-reducible by q queries *if there exist a randomized* (query generating) *algorithm Q and a* (recovery) *algorithm R such that for every $g \in \Pi$ and every $z \in D$ the following two conditions hold:*

1. Recovery: *For every sequence of queries $(r_1, ..., r_q)$ generated by $Q(z)$, it holds that*
$$R(z, r_1, ..., r_q, g(r_1), ..., g(r_q)) = g(z).$$
2. Query distribution: *For each $i \in [q]$, the i^{th} element in $Q(z)$ is uniformly distributed in D; that is, for every $e \in D$, it holds that*
$$\mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[r_i = e] = \frac{1}{|D|}.$$

Indeed, various generalizations are possible.[16] For example, we may allow the recovery algorithm to be randomized and (only) require that it is correct with probability 2/3.

The self-correction paradigm amounts to using such a random self-reduction, while observing that if f is only ϵ-close to $g \in \Pi$ (rather than f being in Π), then, with probability at least $1 - q \cdot \epsilon$,

[15]Otherwise (i.e., if $\epsilon < 2^{-\ell+1}$), we can just recover f by making $2^\ell = O(1/\epsilon)$ queries.

[16]**Advanced comment:** One generalization, which only matters when one considers the computational efficiency of the recovery algorithm R, is providing R with the coins used by Q (rather than with the generated q-long sequence of queries). Needless to say, the recovery algorithm still gets z as well as the oracle answers $g(r_1), ..., g(r_q)$. That is, denoting by $Q(z;\omega)$ the output of Q on input z, when using coins ω, we replace the recovery condition by $R(z, \omega, g(r_1), ..., g(r_q)) = g(z)$ for every ω, where $(r_1, ..., r_q) = Q(z;\omega)$.

the value obtained by applying R on the answers obtained by querying f on $Q(z)$ matches $g(z)$. This observation is captured by the following theorem.

Theorem 10 (self-correction): *Let Π be a set of functions ranging over D. Suppose that functions in Π are randomly self-reducible by q queries, and denote the corresponding query-generating and recovery algorithms by Q and R, respectively. Then, for every f that is ϵ-close to some $f' \in \Pi$ and for every $z \in D$, it holds that*

$$\mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[R(z, r_1, ..., r_q, f(r_1), ..., f(r_q)) = f'(z)] \geq 1 - q \cdot \epsilon.$$

It follows that f cannot be at distance smaller than $1/2q$ from two different functions in Π. Hence, functions in Π must be at mutual distance of at least $1/q$; that is, *if Π is random self-reducible by q queries, then for every distinct $f, g \in \Pi$ it holds that $\delta(f, g) \geq 1/q$* (see Exercise 3).

(Indeed, Theorem 10 and its proof are implicit in Section 2.1 as well as in the analysis of the linearity and low-degree tests.)

Proof: By the (recovery condition in the) hypothesis, we know that for every sequence of queries $(r_1, ..., r_q)$ generated by Q, it holds that $R(z, r_1, ..., r_q, f'(r_1), ..., f'(r_q)) = f'(z)$. Hence,

$$\begin{aligned}
&\mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[R(z, r_1, ..., r_q, f(r_1), ..., f(r_q)) = f'(z)] \\
&\quad\geq\ \mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[(\forall i \in [q])\ \ f(r_i) = f'(r_i)] \\
&\quad\geq\ 1 - \sum_{i\in[q]} \mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[f(r_i) \neq f'(r_i)] \\
&\quad=\ 1 - q \cdot \mathbf{Pr}_{r\in D}[f(r) \neq f'(r)],
\end{aligned}$$

where the equality uses the (the query distribution condition in the) hypothesis by which each of the queries generated by Q is uniformly distributed in D. Recalling that f is ϵ-close to f', the claim follows. ■

An archetypical application. In the following result, we refer to the general notion of solving a promise problem. Recall that a promise problem is specified by two sets, P and Q, where P is the promise and Q is the question. The problem, denoted (P, Q), is define as *given an input in P, decide whether or not the input is in Q* (where standard decision problems use the trivial promise in which P consists of the set of all possible inputs). Equivalently, the problem consists of distinguishing between inputs in $P \cap Q$ and inputs in $P \setminus Q$, and indeed promise problems are often presented as pairs of non-intersecting sets (i.e., the set of YES-instances and the set of NO-instances). Lastly, note that here we consider solving such promise problems by probabilistic oracle machines, which means that the answer needs to be correct (only) with probability at least 2/3.

Specifically, we shall refer to the promise problem (Π', Π''), where Π' is randomly self-reducible and *testable* within some given complexity bounds. We shall show that if (Π', Π'') is *solvable* within some complexity, then $\Pi' \cap \Pi''$ is *testable* within complexity that is related to the three given bounds.

Theorem 11 (testing intersection with a self-correctable property): *Let Π' and Π'' be sets of functions ranging over D. Suppose that functions in Π' are randomly self-reducible by q queries, that Π' is ϵ-testable using $q'(\epsilon)$ queries, and that the promise problem (Π', Π'') can be solved in* *query complexity* q'' (i.e., a probabilistic q''-query oracle machine can distinguish between inputs in $\Pi' \cap \Pi''$ and inputs in $\Pi' \setminus \Pi''$). *Then,* $\Pi' \cap \Pi''$ *is* ϵ*-testable using* $O(q'(\min(\epsilon, 1/3q))) + q \cdot \widetilde{O}(q'')$ *queries.*

(Indeed, Theorem 11 and its proof are implicit in Section 2.1.) We stress that Theorem 11 does not employ a tester for Π'', but rather employs a decision procedure for the promise problem (Π', Π''). However, as shown in Exercise 4, such a decision procedure is implied by any $(1/q)$-tester for Π'', since Π' has distance at least $1/q$ (see Exercise 3).[17]

Proof: We propose the following tester for $\Pi' \cap \Pi''$. On input f, the tester proceeds in two steps:

1. It invokes the $\min(\epsilon, 1/3q)$-tester for Π' on input f and rejects if this tester rejects.
2. Otherwise, it invokes the decision procedure for the promise problem (Π', Π''), while providing this procedure with answers obtained from f via the self-correction procedure (for Π') guaranteed by Theorem 10. Specifically, let Q and R be the query-generating and recovery algorithms guaranteed by Theorem 10. Then, the query z is answered with the value $R(z, r_1, ..., r_q, f(r_1), ..., f(r_q))$, where $(r_1, ..., r_q) \leftarrow Q(z)$. Needless to say, the tester decides according to the verdict of the decision procedure.

By using error reduction, we may assume that both the tester of Π' and the solver of (Π', Π'') have error probability at most 0.1. Likewise, we assume that the self-correction procedure has error probability at most $0.1/q''$ (when invoked on any input that is $1/3q$-close to Π').[18] Hence, Step 1 can be implemented using $O(q'(\min(\epsilon, 1/3q)))$ queries, whereas Step 2 can be implemented using $q'' \cdot O(q \cdot \log q'')$ queries.

We now turn to the analysis of the proposed tester. If $f \in \Pi' \cap \Pi''$, then Step 1 rejects with probability at most 0.1, and otherwise we proceed to Step 2, which accepts with probability at least 0.9 (since in this cases all answers provided by the self-correction procedure are always correct). On the other hand, if f is ϵ-far from $\Pi' \cap \Pi''$, then we consider two cases.

Case 1: f is $\min(\epsilon, 1/3q)$-far from Π'. In this case, Step 1 rejects with probability at least 0.9.

Case 2: f is $\min(\epsilon, 1/3q)$-close to Π'. Let $f' \in \Pi'$ be $\min(\epsilon, 1/3q)$-close to f, and note that $f' \notin \Pi''$ (since otherwise f would have been ϵ-close to $\Pi' \cap \Pi''$). Hence, the decision procedure employed in Step 2 would have rejected f' with probability at least 0.9, since $f' \in \Pi' \setminus \Pi''$. However, this procedure is not invoked with f', but is rather provided with answers according to the self-correction procedure for Π'. Still, since f is $1/3q$-close to f', each of these answers agrees with f' with probability at least $1 - 0.1/q''$, which implies that with probability at least 0.9 all q'' answers agree with $f' \in \Pi' \setminus \Pi''$. We conclude that Step 2 rejects with probability at least $0.9 \cdot 0.9$.

Combining the two cases, we infer that any function that is ϵ-far from $\Pi' \cap \Pi''$ is rejected with probability greater than 0.8, and the theorem follows. ∎

[17] **Advanced comment:** In light of the latter fact, we would have gained nothing by considering a promise problem version of testing Π'' when promised that the input is in Π' (rather than a tester for Π'' or a solver for (Π', Π'')). By such a version we mean the task of distinguishing between inputs in $\Pi' \cap \Pi''$ and inputs in Π' that are ϵ-far from Π'', where ϵ is a given proximity parameter as in standard testing problems. As stated above, if $\epsilon \leq 1/q$, then all inputs in $\Pi' \setminus \Pi''$ are ϵ-far from $\Pi'' \cap \Pi'$.

[18] Specifically, we invoke the self-correction procedure for $O(\log q'')$ times and take the value that appears most frequently. Note that each invocation returns the correct value with probability at least $1 - q \cdot 1/3q = 2/3$.

Detour: on the complexity of testing self-correctable properties. An interesting feature of self-correctable properties is that the complexity of testing them is inversely proportional to the proximity parameter. This is due to the fact that ϵ-testing a property that consists of functions that are randomly self-reducible by t queries reduces to $1/2t$-testing this property.[19]

Theorem 12 (proximity parameter reduction for self-correctable properties): *Suppose that the functions in Π are randomly self-reducible by t queries, and that Π has a tester of query complexity $q : \mathbb{N} \times (0,1] \to \mathbb{N}$. Then, Π has a tester of query complexity $q' : \mathbb{N} \times (0,1] \to \mathbb{N}$ such that $q'(n,\epsilon) = q(n,1/2t) + O(t/\epsilon)$. Furthermore, one-sided error probability is preserved.*

Proof Sketch: On input f, the new tester proceeds as follows.[20]

1. Invoke the tester (hereafter denoted T) guaranteed by the hypothesis with proximity parameter $1/2t$. If T reject, then the new tester rejects.

2. Uniformly select a sample S of $O(1/\epsilon)$ elements in the domain of f, and compare the value of f on each of these points to the value obtained via the self-correction procedure (which relies on the random self-reducibility of Π).

   Specifically, let Q and R denote the query-generating and recovery algorithms guaranteed by the hypothesis. Then, for each $x \in S$, we compare the value of $f(x)$ to $R(x, r_1, ..., r_t, f(r_1), ..., f(r_t))$, where $(r_1, ..., r_t) \leftarrow Q(x)$, and accept if and only if no mismatch is found.

Note that when Step 2 is employed to any $f \in \Pi$, no mismatch is ever found. On the other hand, any function that is $1/2t$-far from Π is rejected in Step 1 with probability at least $2/3$. Lastly, suppose that the distance of f from Π denoted δ, resides in the interval $(\epsilon, 1/2t]$. Let $f' \in \Pi$ be at distance δ from f, and let D denote the domain of f. In this case, we have

$$\begin{aligned}
&\mathbf{Pr}_{x\in D,\,(r_1,...,r_t)\leftarrow Q(x)}[f(x) \neq R(x, r_1, ..., r_t, f(r_1), ..., f(r_t))] \\
&\quad\geq\ \mathbf{Pr}_{x\in D,\,(r_1,...,r_t)\leftarrow Q(x)}[f(x) \neq f'(x) = R(x, r_1, ..., r_t, f(r_1), ..., f(r_t))] \\
&\quad\geq\ \mathbf{Pr}_{x\in D}[f(x) \neq f'(x)] \cdot \min_{x\in D}\left\{\mathbf{Pr}_{(r_1,...,r_t)\leftarrow Q(x)}[f'(x) = R(x, r_1, ..., r_t, f(r_1), ..., f(r_t))]\right\} \\
&\quad\geq\ \epsilon \cdot (1 - t \cdot \delta),
\end{aligned}$$

which is at least $\epsilon/2$. The theorem follows. ■

3 Testing juntas

Here we consider testing a property of Boolean functions $f : \{0,1\}^\ell \to \{0,1\}$ called k-junta, which consists of functions that depend on at most k of their variables. Indeed, the notion of a k-junta generalizes the notion of a dictatorship, which corresponds to the special case of $k = 1$. For $k \geq 2$, the set of k-juntas is a proper superset of the set of k-monomials, which of course says nothing about the relative complexity of testing these two sets.

[19] **Advanced comment:** Actually, we can use a c/t-tester, for any constant $c > 1$. The point is that, when the function is c/t-close to the property, we only need the self-corrector to yield the correct value with positive probability (rather than with probability greater than 1/2).

[20] An alternative presentation views Step 2 as repeating a $(t + 1)$-query proximity-oblivious tester of detection probability $\varrho(\delta) = 1 - t \cdot \delta$ (see Exercise 6) for $O(1/\epsilon)$ times. Indeed, we can obtain a $(q(n, 1/2t) + t + 1)$-query proximity-oblivious tester of detection probability $\varrho(\delta) = \Omega(\delta)$ for Π.

Definition 13 (k-juntas): *A function $f : \{0,1\}^\ell \to \{0,1\}$ is called a* junta of size k (or a k-junta) *if there exist k indices $i_1, ..., i_k \in [\ell]$ and a Boolean function $f' : \{0,1\}^k \to \{0,1\}$ such that $f(x) = f'(x_{i_1} \cdots x_{i_k})$ for every $x = x_1 \cdots x_\ell \in \{0,1\}^\ell$.*

In order to facilitate the exposition, let us recall some notation: For $I = \{i_1, ..., i_t\} \subseteq [\ell]$ such that $i_1 < \cdots < i_t$ and $x \in \{0,1\}$, let x_I denote the t-bit long string $x_{i_1} \cdots x_{i_t}$. Then, the condition in Definition 13 can be restated as asserting that *there exists a k-set $I \subseteq [\ell]$ and a function $f' : \{0,1\}^k \to \{0,1\}$ such for every $x \in \{0,1\}^\ell$ it holds that $f(x) = f'(x_I)$*. In other words, for every $x, y \in \{0,1\}^\ell$ that satisfy $x_I = y_I$ it holds that $f(x) = f(y)$. An alternative formulation of this condition asserts that *there exists a $(\ell - k)$-set $U \subseteq [\ell]$ that has zero influence*, where the influence of a subset S on f equals the probability that $f(x) \neq f(y)$ when x and y are selected uniformly subject to $x_{[\ell]\setminus S} = y_{[\ell]\setminus S}$ (see Definition 15.1). Indeed, the two alternatives are related via the correspondence between U and $[\ell] \setminus I$.

Note that the number of k-juntas is at most $\binom{\ell}{k} \cdot 2^{2^k}$, and so this property can be ϵ-tested by $O(2^k + k \log \ell)/\epsilon$ queries via proper learning (see first lecture). Our aim is to present an ϵ-tester of query complexity $\text{poly}(k)/\epsilon$.

The key observation is that if f is a k-junta, then any partition of $[\ell]$ will have at most k subsets that have positive influence. On the other hand, as will be shown in the proof of Theorem 15, if f is δ-far from being a k-junta, then a random partition of $[\ell]$ into $O(k^2)$ subsets is likely to result in more than k subsets that each have $\Omega(\delta/k^2)$ influence on the value of the function. To gain some intuition regarding the latter fact, suppose that f is the exclusive-or of $k+1$ variables. Then, with high constant probability, the locations of these $k+1$ variables will reside in $k+1$ different subsets of a random $O(k^2)$-way partition, and each of these subsets will have high influence. The same holds if f has $k+1$ variables that are each quite influential (but this is not necessarily the case, in general, and the proof will have to deal with that). In any case, the aforementioned dichotomy leads to the following algorithm.

Algorithm 14 (testing k-juntas): *On input parameters ℓ, k and ϵ, and oracle access to a function $f : \{0,1\}^\ell \to \{0,1\}$, the tester sets $t = O(k^2)$ and proceeds as follows.*

1. *Select a random t-way partition of $[\ell]$ by assigning to each $i \in [\ell]$ a uniformly selected $j \in [t]$, which means that i is placed in the j^{th} part.*

   Let $(R_1, ..., R_t)$ denote the resulting partition.

2. *For each $j \in [t]$, estimate the influence of R_j on f, or rather check whether R_j has positive influence on f. Specifically, for each $j \in [t]$, select uniformly $m \stackrel{\text{def}}{=} \widetilde{O}(t)/\epsilon$ random pairs (x, y) such that x and y agree on the bit positions in $\overline{R_j} = [\ell] \setminus R_j$* (i.e., $x_{\overline{R_j}} = y_{\overline{R_j}}$), *and mark j as influential if $f(x) \neq f(y)$ for any of these pairs (x, y).*

3. *Accept if and only if at most k indices were marked influential.*

The query complexity of Algorithm 14 is $t \cdot m = \tilde{(}t^2)/\epsilon = \widetilde{O}(k^4)/\epsilon$.

Theorem 15 (analysis of Algorithm 14): *Algorithm 14 is a one-sided tester for k-juntas.*

Proof: For sake of good order, we start by formally presenting the definition of the influence of a set on a Boolean function.[21]

[21]Recall that, for $S \subseteq [\ell]$ and $x \in \{0,1\}^\ell$, we let x_S denote the $|S|$-bit long string $x_{i_1} \cdots x_{i_s}$, where $S = \{i_1, ..., i_s\}$ and $i_1 < \cdots < i_s$. Also, $\overline{S} = [\ell] \setminus S$.

Definition 15.1 (influence of a set): *The* influence of a subset $S \subseteq [\ell]$ on the function $f : \{0,1\}^\ell \to \{0,1\}$, *denoted* $\mathtt{I}_S(f)$*, equals the probability that* $f(x) \neq f(y)$ *when* x *and* y *are selected uniformly subject to* $x_{\overline{S}} = y_{\overline{S}}$*; that is,*

$$\mathtt{I}_S(f) \stackrel{\rm def}{=} \mathbf{Pr}_{x,y\in\{0,1\}^\ell : x_{\overline{S}}=y_{\overline{S}}}[f(x) \neq f(y)]. \tag{5}$$

Note that the substrings x_S and y_S are uniformly and independently distributed in $\{0,1\}^{|S|}$, whereas the substring $x_{\overline{S}} = y_{\overline{S}}$ is uniformly distributed in $\{0,1\}^{|\overline{S}|}$ (independently of x_S and y_S). Hence, $\mathtt{I}_S(f)$ equals the probability that the value of f changes when the argument is "re-randomized" in the locations that correspond to S, while fixing the random value assigned to the locations in $\overline{S}$. In other words, $\mathtt{I}_S(f)$ equals the expected value of $\mathbf{Pr}_{x,y\in\Omega_{\overline{S},r}}[f(x) \neq f(y)]$, where $\Omega_{\overline{S},r} \stackrel{\rm def}{=} \{z \in \{0,1\}^\ell : z_{\overline{S}} = r\}$ and the expectation is taken uniformly over all possible choices of $r \in \{0,1\}^{|\overline{S}|}$; that is,

$$\mathtt{I}_S(f) = \mathbb{E}_{r\in\{0,1\}^{|\overline{S}|}}\left[\mathbf{Pr}_{x,y\in\{0,1\}^\ell : x_{\overline{S}}=y_{\overline{S}}=r}[f(x) \neq f(y)]\right] \tag{6}$$

The following two facts are quite intuitive, but their known proofs are quite tedious:[22]

Fact 1 (monotonicity): $\mathtt{I}_S(f) \leq \mathtt{I}_{S\cup T}(f)$.

Fact 2 (sub-additivity): $\mathtt{I}_{S\cup T}(f) \leq \mathtt{I}_S(f) + \mathtt{I}_T(f)$.

Now, if f is a k-junta, then there exists a k-subset $J \subseteq [\ell]$ such that $[\ell] \setminus J$ has zero influence on f, and so Algorithm 14 always accepts f (since for every partition of $[\ell]$ at most k parts intersect J).[23] On the other hand, we first show (see Claim 15.2) that if f is δ-far from a k-junta, then for every k-subset $J \subseteq [\ell]$ it holds that $[\ell] \setminus J$ has influence greater than δ on f. This, by itself, does not suffice for concluding that Algorithm 14 rejects f (w.h.p.), but it will be used towards establishing the latter claim.

Claim 15.2 (influences of large sets versus distance from small juntas): *If there exists a k-subset $J \subseteq [\ell]$ such that $[\ell] \setminus J$ has influence at most δ on f, then f is δ-close to being a k-junta.*

Proof: Fixing J as in the hypothesis, let $g(x) \stackrel{\rm def}{=} \mathtt{maj}_{u:u_J=x_J}\{f(u)\}$. Then, on the one hand, g is a k-junta, since the value of $g(x)$ depends only on x_J. On the other hand, we shall show that f is δ-close to g. Let $g' : \{0,1\}^k \to \{0,1\}$ be such that $g(x) = g'(x_J)$ for every x. Then

$$\begin{aligned}
\mathbf{Pr}_{x\in\{0,1\}^\ell}[f(x) = g(x)] &= \mathbf{Pr}_{x\in\{0,1\}^\ell}[f(x) = g'(x_J)] \\
&= \mathbb{E}_{\alpha\in\{0,1\}^k}\left[\mathbf{Pr}_{x:x_J=\alpha}[f(x) = g'(\alpha)]\right] \\
&= \mathbb{E}_{\alpha\in\{0,1\}^k}[p_\alpha]
\end{aligned}$$

where $p_\alpha \stackrel{\rm def}{=} \mathbf{Pr}_{x:x_J=\alpha}[f(x) = g'(\alpha)]$. Let Z denote the uniform distribution over $\{z \in \{0,1\}^\ell : z_J = \alpha\}$. Then, $p_\alpha = \mathbf{Pr}[f(Z) = g'(\alpha)] = \max_v\{\mathbf{Pr}[f(Z) = v]\}$, by the definition of g' (and g). It

[22]See [15, Prop. 2.4] or Exercise 11. An alternative proof appears in [6, Cor. 2.10], which relies on [6, Prop. 2.9]. Indeed, a simpler proof will be appreciated.

[23]This uses Fact 1. Specifically, for R such that $R \cap J = \emptyset$, it holds that $\mathtt{I}_R(f) \leq \mathtt{I}_{[\ell]\setminus J}(f) = 0$.

follows that the collision probability of $f(Z)$, which equals $\sum_v \mathbf{Pr}[f(Z)=v]^2 = p_\alpha^2 + (1-p_\alpha)^2$, is at most p_α. Hence,

$$\begin{aligned}
\mathbf{Pr}_{x\in\{0,1\}^\ell}[f(x)=g(x)] &= \mathbb{E}_{\alpha\in\{0,1\}^k}[p_\alpha] \\
&\geq \mathbb{E}_{\alpha\in\{0,1\}^k}\left[\sum_v \mathbf{Pr}_{z:z_J=\alpha}[f(z)=v]^2\right] \\
&= \mathbb{E}_{\alpha\in\{0,1\}^k}\left[\mathbf{Pr}_{x,y:x_J=y_J=\alpha}[f(x)=f(y)]\right] \\
&= \mathbf{Pr}_{x,y:x_J=y_J}[f(x)=f(y)] \\
&= 1-\mathtt{I}_{\overline{J}}(f).
\end{aligned}$$

Recalling that $\mathtt{I}_{\overline{J}}(f) \leq \delta$, it follows that f is δ-close to g, and recalling that g is a k-junta the claim follows. ■

Recall that our goal is to prove that any function that is ϵ-far from being a k-junta is rejected by Algorithm 14 with probability at least 2/3. Let us fix such a function f for the rest of the proof, and shorthand $\mathtt{I}_S(f)$ by $\mathtt{I}_S$. By Claim 15.2, every $(\ell-k)$-subset has influence greater than ϵ on f. This noticeable influence may be due to one of two cases:

1. There are at least $k+1$ elements in $[\ell]$ that have each a noticeable influence (i.e., each singleton that consists of one of these elements has noticeable influence).

   Fixing such a collection of $k+1$ influential elements, a random t-partition is likely to have these elements in separate parts (since, say, $t > 10\cdot(k+1)^2$), and in such a case each of these parts will have noticeable influence, which will be detected by the algorithm and cause rejection.

2. Otherwise (i.e., at most k elements have noticeable influence), the set of elements that are individually non-influential is of size at least $\ell-k$, and thus contains an $(\ell-k)$-subset, which (by Claim 15.2) must have noticeable influence. It is tempting to think that the t parts in a random t-partition will each have noticeable influence, but proving this fact is not straightforward at all. Furthermore, this fact is true only because, in this case, we have a set that has noticeable influence but consists of elements that are each of small individual influence.

Towards making the foregoing discussion precise, we fix a threshold $\tau = c\cdot\epsilon/t$, where $c>0$ is a universal constant (e.g., $c=0.01$ will do), and consider the set of elements H that are "heavy" (w.r.t individual influence); that is,

$$H \stackrel{\text{def}}{=} \{i\in[\ell] : \mathtt{I}_{\{i\}} > \tau\}. \tag{7}$$

The easy case is when $|H|>k$. In this case (and assuming $t \geq 3\cdot(k+1)^2$), with probability at least 5/6, the partition selected in Step 1 has more than k parts that intersect H (i.e., $\mathbf{Pr}_{(R_1,..,R_t)}[|\{i\in[t] : R_i\cap H \neq \emptyset\}| > k] \geq 5/6$).[24] On the other hand, for each $j\in[t]$ such that $R_j \cap H \neq \emptyset$, it holds that $\mathtt{I}_{R_j} \geq \min_{i\in H}\{\mathtt{I}_{\{i\}}\} > c\epsilon/t$, where the first inequality uses Fact 1. Hence, with probability at

[24]Let H' be an arbitrary $(k+1)$-subset of H. Then, the probability that some R_j has more than a single element of H' is upper bounded by $\binom{k+1}{2}\cdot 1/t < (k+1)^2/2t$.

least $1-(1-c\epsilon/t)^m \geq 1-(1/6t)$, Step 2 (which estimates $\mathtt{I}_{R_j}$ by $m = \widetilde{O}(t)/\epsilon$ experiments) will mark j as influential, and consequently Step 3 will reject with probability at least $(5/6)^2 > 2/3$.

We now turn to the other case, in which $|H| \leq k$. By Claim 15.2 (and possibly Fact 1)[25], we know that $\mathtt{I}_{\overline{H}} > \epsilon$. Our aim is to prove that, with probability at least 5/6 over the choice of the random t-partition $(R_1, ..., R_t)$, there are at least $k+1$ indices j such that $\mathtt{I}_{R_j} > c\epsilon/t$. In fact, we prove something stronger.[26]

Lemma 15.3 (on the influence of a random part): *Let H be as in* Eq. (7) *and $\mathtt{I}_{\overline{H}} > \epsilon$. Then, for every $j \in [t]$, it holds that $\mathbf{Pr}[\mathtt{I}_{R_j \cap \overline{H}} > \epsilon/2t] > 0.9$.*

(We comment that the same proof also establishes that $\mathbf{Pr}[\mathtt{I}_{R_j \cap \overline{H}} = \Omega(\epsilon/t \log t)] > 1-(1/6t)$, which implies that with probability at least 5/6 each R_j has influence $\Omega(\epsilon/t \log t)$.)[27] Calling j good if $\mathtt{I}_{R_j} > c\epsilon/t$, Lemma 15.3 implies that the expected number of good j's it at least $0.9t$. Hence, with probability at least 5/6, there exist $\frac{0.9t-(5/6)t}{1/6} = 0.4t > k$ good j's, and the bound on the rejection probability of the algorithm holds (as in the easy case).

Proof: Denote $R = R_j$, and recall that each $i \in [\ell]$ is placed in R with probability $1/t$, independently of all other choices.

Things would have been simple if influence was additive; that is, if it were the case that $\mathtt{I}_S = \sum_{i \in S} \mathtt{I}_{\{i\}}$. In this case, applying a multiplicative Chernoff Bound[28] would have yielded the desired bound. Specifically, defining random variables, $\zeta_1, ..., \zeta_\ell$, such that $\zeta_i \stackrel{\text{def}}{=} \zeta_i(R)$ equals $\mathtt{I}_{\{i\}}$ if $i \in (R \setminus H)$ and zero otherwise, and observing that $\sum_{i \in [\ell]} \mathbb{E}[\zeta_i] > \epsilon/t$ and $\zeta_i \in [0, \tau]$, we would have obtained

$$\mathbf{Pr}_R\left[\sum_{i \in [\ell]} \zeta_i(R) < \epsilon/2t\right] < \exp(-\Omega(\tau^{-1}\epsilon/t))$$

which is smaller than 0.1 when $c = t\tau/\epsilon > 0$ is small enough. However, $\mathtt{I}_S = \sum_{i \in S} \mathtt{I}_{\{i\}}$ does not hold in general, and for this reason the proof of the current lemma is not so straightforward. In particular, we shall use the following fact about the influence of sets.

Fact 3 (diminishing marginal gain): For every $S, T, M \subseteq [\ell]$ and every f, it holds that

$$\mathtt{I}_{S \cup T \cup M}(f) - \mathtt{I}_{S \cup T}(f) \leq \mathtt{I}_{S \cup M}(f) - \mathtt{I}_S(f).$$

(This fact may not be as intuitive as Facts 1 and 2, but it is quite appealing; see Exercise 12 or [15, Prop. 2.5].)

[25] Fact 1 is used in case $|H| < k$. In this case we consider an arbitrary k-superset $H' \supset H$ and use $\mathtt{I}_{\overline{H}} \geq \mathtt{I}_{\overline{H'}} > \epsilon$. Alternatively, one could have stated Claim 15.2 with respect to set of size at most k, while observing that its current proof would have held intact.

[26] The significant strengthening is in arguing on each individual R_j rather than on all of them. The fact that the lemma refers to $\mathtt{I}_{R_j \cap \overline{H}}$ rather than to $\mathtt{I}_{R_j}$ is less significant. While the weaker form suffices for our application, we believe that the stronger form is more intuitive (both as a statement and as a reflection of the actual proof).

[27] Using this bound would have required to use (in Step 2) a value of m that is a factor of $\log t$ larger.

[28] The standard formulation of the multiplicative Chernoff Bound refers to independent random variables, $\zeta_1, ..., \zeta_\ell$, such that $\zeta_i \in [0,1]$ and $\mu = \sum_{i \in [\ell]} \mathbb{E}[\zeta_i]$, and asserts that for $\gamma \in (0,1)$ it holds that $\mathbf{Pr}\left[\sum_{i \in [\ell]} \zeta_i < (1-\gamma) \cdot \mu\right] < \exp(-\Omega(\gamma^2 \mu))$. We shall use $\zeta_i \in [0, \tau]$, and thus have $\mathbf{Pr}\left[\sum_{i \in [\ell]} \zeta_i < (1-\gamma) \cdot \mu\right] < \exp(-\Omega(\gamma^2 \tau^{-1} \mu))$.

Now, we consider the following (less straightforward) sequence of random variables, $\zeta_1, ..., \zeta_\ell$, such that $\zeta_i \overset{\rm def}{=} \zeta_i(R)$ equals $\mathtt{I}_{[i]\setminus H} - \mathtt{I}_{[i-1]\setminus H}$ if $i \in R$ and zero otherwise.[29] Observe that

1. The ζ_i's are independent random variables, since the value of ζ_i only depends on whether or not $i \in R$.
2. Each ζ_i is assigned values in the interval $[0, \tau]$, since $0 \le \mathtt{I}_{[i]\setminus H} - \mathtt{I}_{[i-1]\setminus H} \le \tau$, where the first inequality is due to Fact 1 and the second inequality follows by combining Fact 2 and $\mathtt{I}_{\{i\}\setminus H} \le \tau$. (Indeed, if $i \in H$, then $\mathtt{I}_{\{i\}\setminus H} = \mathtt{I}_\emptyset = 0$, and otherwise $\mathtt{I}_{\{i\}\setminus H} = \mathtt{I}_{\{i\}} \le \tau$.)
3. The expected value of $\sum_{i\in[\ell]} \zeta_i$ equals $\mathtt{I}_{\overline{H}}/t$, since $\mathbb{E}[\zeta_i] = (\mathtt{I}_{[i]\setminus H} - \mathtt{I}_{[i-1]\setminus H})/t$ whereas $\sum_{i\in[\ell]}(\mathtt{I}_{[i]\setminus H} - \mathtt{I}_{[i-1]\setminus H})$ equals $\mathtt{I}_{[\ell]\setminus H} - \mathtt{I}_\emptyset = \mathtt{I}_{\overline{H}}$.
4. As shown next, for every fixed set F, it holds that
$$\sum_{i\in[\ell]} \zeta_i(F) \le \mathtt{I}_{F\cap\overline{H}}. \tag{8}$$

   Therefore, $\mathbf{Pr}_R\left[\mathtt{I}_{R\cap\overline{H}} > \epsilon/2t\right] \le \mathbf{Pr}\left[\sum_{i\in[\ell]} \zeta_i(F) > \epsilon/2t\right]$, and so upper-bounding the latter probability suffices for establishing the lemma.

   The proof of Eq. (8) uses Fact 3, and proceeds as follows:[30]
$$\begin{aligned}\sum_{i\in[\ell]} \zeta_i(F) &= \sum_{i\in F}(\mathtt{I}_{[i]\setminus H} - \mathtt{I}_{[i-1]\setminus H}) \\ &= \sum_{i\in F\setminus H}(\mathtt{I}_{([i-1]\setminus H)\cup\{i\}} - \mathtt{I}_{[i-1]\setminus H}) \\ &\le \sum_{i\in F\setminus H}(\mathtt{I}_{(([i-1]\setminus H)\cap F)\cup\{i\}} - \mathtt{I}_{([i-1]\setminus H)\cap F}) \\ &= \sum_{i\in[\ell]}(\mathtt{I}_{([i]\setminus H)\cap F} - \mathtt{I}_{([i-1]\setminus H)\cap F}) \\ &= \mathtt{I}_{([\ell]\setminus H)\cap F} \\ &= \mathtt{I}_{F\cap\overline{H}}\end{aligned}$$

   where the inequality uses Fact 3 (with $S = ([i-1]\setminus H)\cap F$, $T = ([i-1]\setminus H)\cap\overline{F}$ and $M = \{i\}$, and so $S \cup T = [i-1]\setminus H$).

Hence, $\zeta = \sum_{i\in[\ell]} \zeta_i$ is the sum of ℓ independent random variables, each ranging in $[0, c\epsilon/t]$, such that $\mathbb{E}[\zeta] > \epsilon/t$. Applying a multiplicative Chernoff Bound implies that under these conditions it holds that $\mathbf{Pr}[\zeta \le \epsilon/2t] < \exp(-\Omega(1/c)) < 0.1$, and the lemma follows (by Eq. (8), when using $F = R$ and recalling that $\zeta_i = \zeta_i(R)$). $\blacksquare$

Let us recap. Assuming that f is ϵ-far from being a k-junta, we have defined the set H (in Eq. (7)) and showed that if $|H| > k$ then the algorithm rejects with probability at least 5/6.

[29] Indeed, we define $[0] = \emptyset$, which implies $\mathtt{I}_{[0]\setminus H} = \mathtt{I}_\emptyset = 0$.

[30] The second equality (i.e., moving from summation over F to summation over $F \setminus H$) uses the fact that for every $i \in H$ it holds that $[i] \setminus H = [i-1] \setminus H$. Likewise, the third equality (i.e., moving from summation over $F \setminus H$ to summation over $[\ell]$) uses the fact that for every $i \in H \cup \overline{F}$ it holds that $([i] \setminus H) \cap F = ([i-1] \setminus H) \cap F$.

On the other hand, if $|H| \leq k$, then the hypothesis of Lemma 15.3 holds, and iit follows that $\mathbf{Pr}[\mathtt{I}_{R_j} > \epsilon/2t] > 0.9$ for each $j] \in [t]$. As shown above, this implies that the algorithm rejects with probability at least $5/6$ also in this case. The theorem follows. ■

Digest. The main difficulty is establishing Theorem 15 is captured by the proof of Lemma 15.3, which can be abstracted as follows. For a function $\nu : 2^{[\ell]} \to [0,1]$ assigning values to subsets of $[\ell]$ such that $\nu([\ell]) \geq \epsilon$ and $\max_{i\in[\ell]}\{\nu(\{i\})\} \leq \tau \ll \epsilon/t$, we wish to show that $\mathbf{Pr}_R[\nu(R) < \epsilon/2t]$ is small, when R is selected at random by picking each element with probability $1/t$ independently of all other elements.[31] Of course, this is not true in general, and some conditions must be made on ν such that $\mathbf{Pr}_R[\nu(R) < \epsilon/2t]$ is small. The conditions we have used are (1) monotonicity, (2) sub-additivity, and (3) diminishing marginal gain. These conditions correspond to Facts 1-3, respectively, which were established for $\nu(S) \stackrel{\text{def}}{=} \mathtt{I}_S(f)$. Hence, we actually established the following (which is meaningful only for $\tau \ll \epsilon/t$).

Lemma 16 (Lemma 15.3, generalized): *Let $\nu : 2^{[\ell]} \to [0,1]$ be monotone, sub-additive, and having diminishing marginal gain such that $\nu([\ell]) \geq \epsilon$ and $\max_{i\in[\ell]}\{\nu(\{i\})\} \leq \tau$. Suppose that R is selected at random by picking each element with probability $1/t$ independently of all other elements. Then, $\mathbf{Pr}_R[\nu(R) < \epsilon/2t] < \exp(-\Omega(\tau^{-1}\epsilon/t))$.*

Recall that ν is monotone if $\nu(S\cup T) \geq \nu(S)$ for all $S, T \subseteq [\ell]$, it is sub-additive if $\nu(S\cup T) \leq \nu(S) + \nu(T)$ for all $S, T \subseteq [\ell]$, and it has diminishing marginal gain if $\nu(S\cup T\cap M) - \nu(S\cup T) \leq \nu(S\cup M) - \nu(S)$ for all $S, T, M \subseteq [\ell]$,

4 Historical notes, suggested reading, a comment, and exercises

The presentation of testing dictatorship via self-correction (Section 2.1) is based on the work of Parnas, Ron, and Samorodnitsky [29]. As noted in Section 2.1, the Long Code (presented by Bellare, Goldreich, and Sudan [4]) yields a family of Boolean functions that coincide with dictatorship functions when ℓ is a power of two. Interestingly, the monotone dictatorship tester of [29] is almost identical to the tester of the Long Code presented in [4]. This tester has played a pivotal role in the PCP constructions of [4] as well as in numerous subsequent PCP constructions including those in [4, 22, 23, 31, 14, 24, 25, 28].

The problem of testing monomials was also studied by Parnas, Ron, and Samorodnitsky [29], and the overview provided in Section 2.2 is based on their work. An alternative procedure for testing monomials is presented in the next lecture (on testing via implicit sampling).

The problem of testing juntas was first studied by Fischer, Kindler, Ron, Safra, and Samorodnitsky [15], and the presentation provided in Section 3 is based on their work. Recall that the query complexity of the k-junta tester presented as Algorithm 14 is $\widetilde{O}(k^4)/\epsilon$. Several alternative testers are known, culminating in a tester of Blais [6] that has query complexity $O(k/\epsilon) + \widetilde{O}(k)$, which is almost optimal. For further discussion, see the survey [7].

Self-correction. The self-correction paradigm, as an algorithmic tool towards the construction of property testers, was introduced by Blum, Luby, and Rubinfeld [8]. The self-correction paradigm

[31] Indeed, this formulation refers to the case of $H = \emptyset$, and captures the essence of Lemma 15.3.

has been used extensively in constructions of PCP schemes, starting with [9]. As explained in Section 2.3, this paradigm is based on random self-reducibility, which seems to have first appeared in the "index calculus" algorithms [1, 26, 30] for the Discrete Logarithm Problem. Random self-reducibility was extensively used in (the complexity theoretic foundations of) cryptography, starting with the work of Goldwasser and Micali [21]. (These applications, which have a hardness amplification flavor, may be viewed as applying self-correction to a hypothetical adversary, yielding an algorithm that is postulated not to exist, and thus establishing specific limitations on the success probability of efficient adversaries.)

Comment: Invariances. We note that all properties considered in this lecture are invariant under a permutation of the variables; that is, for each of these properties Π, the function $f : \{0,1\}^\ell \to \{0,1\}$ is in Π if and only if for every permutation $\pi : [\ell] \to [\ell]$ the function $f_\pi(x_1, ..., x_\ell) = f(x_{\pi(1)}, ..., x_{\pi(\ell)})$ is in Π. (Note that such permutations of the ℓ variables induce a permutation of the domain $\{0,1\}^\ell$; that is, the permutation $\pi : [\ell] \to [\ell]$ induces a permutation $T_\pi : \{0,1\}^\ell \to \{0,1\}^\ell$ such that $T_\pi(x) = (x_{\pi(1)}, ..., x_{\pi(\ell)})$.)

Basic exercises

Exercise 1 states a well-known and widely used classic.

Exercise 1 (The Schwartz–Zippel Lemma [32, 33, 10]): *Prove the following two claims.*

Large field version: *Let $p : \mathcal{F}^m \to \mathcal{F}$ be a* non-zero *m-variate polynomial of total degree d over a finite field $\mathcal{F}$. Then, $\mathbf{Pr}_{x\in\mathcal{F}^m}[p(x)=0] \leq d/|\mathcal{F}|$.*

Small field version: *Let $p : \mathcal{F}^m \to \mathcal{F}$ be a* non-zero *m-variate polynomial of total degree d over a finite field $\mathcal{F}$. Then, $\mathbf{Pr}_{x\in\mathcal{F}^m}[p(x)=0] \leq 1 - |\mathcal{F}|^{-d/(|\mathcal{F}|-1)}$.*

Note that the individual degree of p is at most $|\mathcal{F}| - 1$, and so $d \leq m \cdot (|\mathcal{F}| - 1)$. The large field version, which is meaningful only for $|\mathcal{F}| > d$, is called the Schwartz–Zippel Lemma.[32] *When establishing* Eq. (4)*, we used the small field version with $|\mathcal{F}| = 2$.*

Guideline: Both versions are proved by induction on the number of variables, m. The base case of $m = 1$ follows by the fact that $p \not\equiv 0$ has at most d roots, whereas in the small field version we use the fact that $\frac{d}{|\mathcal{F}|} \leq 1 - |\mathcal{F}|^{-d/(|\mathcal{F}|-1)}$ for every $d \in [0, |\mathcal{F}| - 1]$ (which is trivial for $|\mathcal{F}| = 2$).[33] In the induction step, assuming that p depends on all its variables, write $p(x) = \sum_{i=0}^{d} p_i(x_1, ..., x_{m-1}) \cdot x_m^i$, where p_i is an $(m-1)$-variate polynomial of degree at most $d - i$, and let i be the largest integer such that p_i is non-zero. Then, using $x' = (x_1, ..., x_{m-1})$, observe that

$$\mathbf{Pr}_{x\in\mathcal{F}^m}[p(x) = 0] \leq \mathbf{Pr}_{x'\in\mathcal{F}^{m-1}}[p_i(x') = 0] + \mathbf{Pr}_{x'\in\mathcal{F}^{m-1}}[p_i(x') \neq 0] \cdot \mathbf{Pr}_{x\in\mathcal{F}^m}[p(x) = 0|p_i(x') \neq 0].$$

Using the induction hypothesis, prove the induction claim (in both versions).[34]

[32]There is also a version for infinite fields. It asserts that for every finite set S such that $S \subseteq \mathcal{F}$, where $\mathcal{F}$ is an arbitrary field (which is possibly infinite), and for any *non-zero* m-variate polynomial $p : \mathcal{F}^m \to \mathcal{F}$ of total degree d, it holds that $\mathbf{Pr}_{x\in S^m}[p(x)=0] \leq d/|S|$.

[33]In general, for $a = 1/|\mathcal{F}|$ and $b = (\ln|\mathcal{F}|)/(|\mathcal{F}|-1)$, we need to show that $f(x) \stackrel{\rm def}{=} 1 - ax - e^{-bx}$ is non-negative for every integer $x \in [0, |\mathcal{F}|-1]$. This can be shown by observing that f decreases at x if and only if $x > \tau \stackrel{\rm def}{=} (1/b)\ln(b/a)$. Since $\tau > 0$, this means that $\min_{x\in[0,|\mathcal{F}|-1]}\{f(x)\}$ equals $\min(f(0), f(|\mathcal{F}| - 1)) = 0$.

[34]A rather careless approach suffices for the large field case (i.e., we can use $\mathbf{Pr}_x[p(x) = 0] \leq \mathbf{Pr}_{x'}[p_i(x') = 0] + \mathbf{Pr}_x[p(x) = 0|p_i(x') \neq 0]$), but not for the small field case (where one better keep track of the effect of $\mathbf{Pr}_{x'}[p_i(x') \neq 0]$).

Exercise 2 (a variant of Algorithm 3): *Consider a variant of Algorithm 3 in which one selects uniformly $i \in [3]$ and performs only Step i of Algorithm 3, while accepting if and only if this step did not reject. Show that this four-query algorithm is a one-sided error proximity oblivious tester for monotone dictatorship with rejection probability $\varrho(\delta) = \Omega(\delta)$.*

Guideline: Reduce the analysis to the statement of Theorem 4.

Exercise 3 (random self-reducibility mandates distance): *Prove that if Π is random self-reducible by q queries, then for every distinct $g, h \in \Pi$ it holds that $\delta(g,h) \geq 1/q$.*

Guideline: This can be proved by invoking Theorem 10 twice (with z on which g and h disagree): In the first invocation we use $f = f' = g$ and $\epsilon = 0$, and in the second invocation $(f, f') = (g, h)$ and $\epsilon = \delta(g,h) < 1/q$.

Exercise 4 (testing intersection with a self-correctable property): *Let Π' and Π'' be sets of functions. Suppose that functions in Π' are randomly self-reducible by q queries, and that Π' and Π'' are ϵ-testable using $q'(\epsilon)$ and $q''(\epsilon)$ queries, respectively. Then, for every $\epsilon_0 < 1/q$, the property $\Pi' \cap \Pi''$ is ϵ-testable using $O(q'(\min(\epsilon, 1/3q))) + q \cdot \widetilde{O}(q''(\epsilon_0))$ queries.*

Guideline: Using the fact that Π' has distance at least $1/q$ (see Exercise 3), show that the tester for Π'' implies that the promise problem (Π', Π'') can be solved in query complexity $q''(\epsilon_0)$. Finally, invoke Theorem 11.

Exercise 5 (generalizing Theorem 10): *Consider a relaxation of Definition 9 in which R is allowed to be randomized as long as it recovers the correct value with probability at least $2/3$. Suppose that functions in Π are randomly self-reducible by q queries, in this relaxed sense. Prove that for every f that is ϵ-close to some $f' \in \Pi$ and for every $z \in D$, self-correction succeeds with probability at least $\frac{2}{3} \cdot (1 - q \cdot \epsilon)$; that is,*

$$\mathbf{Pr}_{(r_1,...,r_q)\leftarrow Q(z)}[R(z, r_1, ..., r_q, f(r_1), ..., f(r_q)) = f'(z)] \geq \frac{2}{3} \cdot (1 - q \cdot \epsilon).$$

Exercise 6 (POTs for self-correctable properties – take 1):[35] *Show that if the functions in Π are randomly self-reducible by t queries, then Π has a $(t+1)$-query proximity-oblivious tester of detection probability $\varrho(\delta) = (1 - t \cdot \delta) \cdot \delta$.* (Indeed, this is meaningful only to functions that are $1/t$-close to Π, and Π may be hard to test in general (i.e., for $\delta \geq 1/t$).)[36]

Guideline: The POT selects uniformly a random element in the function's domain and compares the value of the function on it to the value obtained via the self-correction procedure.

Exercise 7 (POTs for self-correctable properties – take 2): *In continuation to Exercise 6, suppose that the functions in Π are randomly self-reducible by t queries. Furthermore, suppose that the recovery algorithm is extremely sensitive to the function values in the sence that for every sequence $(z, r_1, ..., r_1, v_1, ..., v_1)$ and for every $i \in [t]$, the mapping*

$$z \mapsto R(z, r_1, ..., r_t, v_1, ..., v_{i-1}, z, v_{i+1}, ..., v_t)$$

[35]See proof of Theorem 12.

[36]See results regarding k-linearity in the lecture notes on lower bound techniques.

is a bijection. Show that Π *has a* $(t+1)$*-query proximity-oblivious tester of detection probability* $\varrho(\delta) = (t+1) \cdot (1 - t \cdot \delta) \cdot \delta$*.* (Indeed, the partial analysis of the linearity tester follows as a special case.)

Guideline: The tester is the same as in Exercise 6, but detection is due not only to the case that the value of the function at the selected point is wrong but also to the cases that correspond the corruption of a single value at one of the queries made by the self-correction procedure.

Additional exercises

The following exercises actually present proof sketches of various results and facts. Exercises 8–10 outline reductions of the task of testing affine subspaces to the task of testing linearity. They are highly recommended. Exercises 11 and 12 call for proving Facts 1-3 that were used in the proof of Theorem 15. These facts are of independent interest, and proofs that are simpler and/or more intuitive than those presented in our guidelines will be greatly appreciated.

Exercise 8 (testing affine subspaces – a reduction to the linear case):[37] *Show that testing whether a Boolean function* $h : \{0,1\}^\ell \to \{0,1\}$ *describes a* $(\ell - k)$*-dimensional affine subspace* (i.e., $h^{-1}(1)$ is such an affine space) *can be reduced to testing whether a Boolean function* $h' : \{0,1\}^\ell \to \{0,1\}$ *describes a* $(\ell - k)$*-dimensional linear subspace* (i.e., $\{x : h'(x) = 1\}$ is such a linear space)*, where the reduction introduces an additive overhead of* $O(2^k)$ *queries.*

Guideline: Note that if h describes a $(\ell - k)$-dimensional affine subspace, then (w.h.p.) a sample of $O(2^k)$ random points in $\{0,1\}^\ell$ contains a point on which h evaluates to 1. On the other hand, for any u such that $h(u) = 1$, consider the function $h'(x) \stackrel{\text{def}}{=} h(x + u)$.

Exercise 9 (reducing testing linear subspaces to testing linearity):[38] *Let* $h : \{0,1\}^\ell \to \{0,1\}$ *be a Boolean function. Show that testing whether* $h^{-1}(1)$ *is an* $(\ell - k)$*-dimensional linear subspace is reducible to testing linearity, while increasing the complexities by a factor of* 2^k*. Specifically, define a function* $g : \{0,1\}^\ell \to \{0,1\}^k \cup \{\perp\}$ *such that if* $H \stackrel{\text{def}}{=} h^{-1}(1)$ *is linear then* g (ranges over $\{0,1\}^k$ and) *is linear and* $g^{-1}(0^k) = H$*. The definition of* g *is based on any fixed sequence of linearly independent vectors* $v^{(1)}, ..., v^{(k)} \in \{0,1\}^\ell$ *such that for every non-empty* $I \subseteq [k]$ *it holds that* $\sum_{i \in I} v^{(i)} \notin H$*.* (If H is an $(\ell - k)$-dimensional affine space, then these $v^{(i)}$'s form a basis for a k-dimensional space that complements H.) *Fixing such a sequence, define* $g : \{0,1\}^\ell \to \{0,1\}^k \cup \{\perp\}$ *such that* $g(x) = (c_1, ..., c_k)$ *if* $(c_1, ..., c_k) \in \{0,1\}^k$ *is the unique sequence that satisfies* $x + \sum_{i \in [k]} c_i v^{(i)} \in H$ *and let* $g(x) = \perp$ *otherwise.* (Whenever we say that g is affine, we mean, in particular, that it never assumes the value $\perp$.)[39]

- *Show that* H *is an* $(\ell - k)$*-dimensional linear space if and only if* g (as defined above) *is a surjective linear function.*
- *Show that if* H *is an* $(\ell - k)$*-dimensional linear space, then a sequence as underlying the definition of* g *can be found* (w.h.p.) *by making* $O(2^k)$ *queries to* h*.*

[37]Based on [20, Sec. 4.1].

[38]Based on [20, Sec. 4.2]. The argument can be generalized to the case of affine subspaces, while also using a reduction of testing affinity to testing linearity (of functions); but, in light of Exercise 8, such a generalization is not needed.

[39]Indeed, when emulating g for the linearity tester, we shall reject if we ever encounter the value $\perp$.

- *Assuming that g is linear, show that testing whether it is surjective can be done by making $O(2^k)$ queries to h.* (It is indeed easier to perform such a check by using $O(2^k)$ queries to g.)

Combining the above ideas, present the claimed reduction. Note that this reduction has two-sided error, and that the resulting tester has query complexity $O(2^k/\epsilon)$ (rather than $O(1/\epsilon)$, all in case $\epsilon < 2^{-k+2}$).[40]

Guideline: Let V be a k-by-ℓ full rank matrix such that $cV \in H$ implies $c = 0^k$ (i.e., the rows of V are the $v^{(i)}$'s of the hypothesis). Recall that $g : \{0,1\}^\ell \to \{0,1\}^k$ is defined such that $g(x) = c$ if $c \in \{0,1\}^k$ is the unique vector that satisfies $x + cV \in H$ (and $g(x) = \perp$ if the number of such vectors is not one). Note that $g^{-1}(0^k) \subseteq H$ always holds (since $g(x) = c$ implies $x + cV \in H$), and that when g never assumes the value $\perp$ equality holds (since in this case $x + cV \in H$ implies that $g(x) = c$). Now, on the one hand, if H is an $(\ell - k)$-dimensional linear space, then, for some full-rank $(\ell-k)$-by-ℓ matrix G, it holds that $H = \{yG : y \in \{0,1\}^{\ell-k}\}$. In this case, g is a surjective linear function (since for every x there exists a *unique* representation of x as $yG+cV$, which implies $x + cV = yG \in H$, and so $g(x) = c$). On the other hand, if g is a surjective linear function (i.e., $g(x) = xT$ for some full-rank ℓ-by-k matrix T), then $H = \{x : g(x) = 0^k\}$, which implies that H is an $(\ell - k)$-dimensional linear subspace. It follows that if g is ϵ-close to being a surjective linear function, then $g^{-1}(0^k)$ is ϵ-close to being an $((\ell - k)$-dimensional) linear space (i.e., the indicator functions of these sets are ϵ-close). In light of the above, consider the following algorithm.

1. Using $O(2^k)$ queries to h, try to find a k-by-ℓ matrix V such that for any non-zero $c \in \{0,1\}^k$ it holds that $cV \notin H$. (The matrix V can be found in k iterations such that in the i^{th} iteration we try to find a vector $v^{(i)}$ such that $\sum_j c_j v^{(j)} \notin H$ holds for every $(c_1, ..., c_i) \in \{0,1\}^i \setminus \{0^i\}$.) If such a matrix V is found, then proceed to the next step. Otherwise, reject.
2. Test whether the function $g : \{0,1\}^\ell \to \{0,1\}^k$ (defined based on this V) is linear, and reject if the linearity tester rejects. When the tester queries g at x, query h on $x + cV$ for all $c \in \{0,1\}^k$, and answer accordingly; that is, the answer is c if c is the unique vector satisfying $h(x+cV) = 1$, otherwise (i.e., $g(x) = \perp$) the execution is suspended and the algorithm rejects.
3. Test whether g is surjective. Assuming that g is linear, the task can be performed as follows.
   (a) Select uniformly at random a target image $c \in \{0,1\}^k$.
   (b) Select uniformly at random a sample S of $t = O(2^k)$ elements in $\{0,1\}^\ell$, and accept if and only if there exists $x \in S$ such that $x + cV \in H$ (i.e., $g(x) = c$).
   We stress that we do not compute g at x, which would have required 2^k queries to h, but rather check whether $g(x) = c$ by making a single query to h (i.e., we query h at $x + cV$).

Exercise 10 (improving the efficiency of the reduction of Exercise 9):[41] *Let $h : \{0,1\}^\ell \to \{0,1\}$ be a Boolean function. In Exercise 9, we reduced ϵ-testing whether $h^{-1}(1)$ is an $(\ell - k)$-dimensional linear subspace to ϵ-testing the linearity of a function g, where the value of g at any point can be*

[40]Needless to say, we would welcome a one-sided error reduction. Note that the case $\epsilon \geq 2^{-k+2}$ can be handled as in Algorithm 6. A complexity improvement for the main case (of $\epsilon < 2^{-k+2}$) appears in Exercise 10.

[41]Based on [20, Sec. 4.3], while being inspired by [18] (as presented in [16, Sec. 7.1.3]). Again, the argument can be generalized to the case of affine subspaces.

computed by making 2^k queries to h. (Indeed, that reduction made $O(2^k)$ additional queries to h.) *This yields an ϵ-tester of time complexity $O(2^k/\epsilon)$ for testing linear subspaces. Recall that, for every $\epsilon_0 < 1/4$, if g is ϵ_0-close to being a linear function, then it is ϵ_0-close to a unique linear function g', which can be computed by self-correction of g* (where each invocation of the self-corrector makes two queries to g and is correct with probability at least $1-2\epsilon_0$). *This suggests the following algorithm.*

1. *Invoke the algorithm of Exercise 9 with proximity parameter set to a suyfficiently small constant $\epsilon_0 > 0$. If the said invocation rejects, then reject. Otherwise, let V be the matrix found in Step 1 of that invocation, and let g be the corresponding function. Let g' denote the linear function closest to g.*

2. *Test that h is ϵ-close to $h' : \{0,1\}^\ell \to \{0,1\}$, where $h'(x) = 1$ if and only if $g'(x) = 0^k$.*

   We implement this step in complexity $\widetilde{O}(1/\epsilon)$ by taking a sample of $m = O(1/\epsilon)$ pairwise independent points in $\{0,1\}^\ell$ such that evaluating g' on these m points only requires time $O(m + 2^k \cdot \widetilde{O}(\log m))$. Specifically, for $t = \lceil \log_2(m+1) \rceil$, select uniformly $s^{(1)}, ..., s^{(t)} \in \{0,1\}^\ell$, compute each $g'(s^{(j)})$ via self-correcting g, with error probability $0.01/t$, and use the sample points $r^{(J)} = \sum_{j\in J} s^{(j)}$ for all non-empty subsets $J \subseteq [t]$.

Assuming that g' is surjective, show that the foregoing algorithm constitutes an ϵ-tester of time complexity $O(\epsilon^{-1} + 2^k \cdot \widetilde{O}(\log(1/\epsilon)))$ for $(\ell - k)$-dimensional linear subspaces. The assumption can be removed by slightly augmenting the algorithm.

Guideline: Note that $g'(\sum_{j\in J} s^{(j)}) = \sum_{j\in J} g'(s^{(j)})$, and show that the $r^{(J)}$'s are pairwise independent.

Exercise 11 (the influence of sets is monotone and sub-additive): *Prove that for every $S, T \subseteq [\ell]$ and every $f : \{0,1\}^\ell \to \{0,1\}$ it holds that*

1. $\mathtt{I}_S(f) \leq \mathtt{I}_{S\cup T}(f)$.

2. $\mathtt{I}_{S\cup T}(f) \leq \mathtt{I}_S(f) + \mathtt{I}_T(f)$.

Guideline: The key observation is that $\mathtt{I}_S(f)$ equals twice the expectation over r of $\mathbb{V}_{z\in\{0,1\}^\ell : z_{\overline{S}} = r_{\overline{S}}}[f(z)]$, where r is distributed uniformly in $\{0,1\}^\ell$; that is,

$$0.5 \cdot \mathtt{I}_S(f) \;=\; \mathbb{E}_{r\in\{0,1\}^\ell}\left[\mathbb{V}_{z\in\{0,1\}^\ell : z_{\overline{S}} = r_{\overline{S}}}[f(z)]\right]. \qquad (9)$$

This is the case since

$$\begin{aligned} \mathtt{I}_S(f) &= \mathbb{E}_{r\in\{0,1\}^\ell}\left[\mathbf{Pr}_{x,y\in\{0,1\}^\ell : x_{\overline{S}} = y_{\overline{S}} = r_{\overline{S}}}[f(x) \neq f(y)]\right] \\ &= \mathbb{E}_{r\in\{0,1\}^\ell}\left[2 \cdot p_r \cdot (1-p_r)\right] \end{aligned}$$

where $p_r \stackrel{\rm def}{=} \mathbf{Pr}_{z\in\{0,1\}^\ell : z_{\overline{S}} = r_{\overline{S}}}[f(z) = 1]$, while observing that $\mathbb{V}_{z\in\{0,1\}^\ell : z_{\overline{S}} = r_{\overline{S}}}[f(z)] = p_r \cdot (1-p_r)$. The two parts of the exercise are proven by manipulation of the relevant quantities when expressed as expectation of variances (i.e., as in Eq. (9)).

Part 1 is proved by considering, without loss of generality, the case that S and T are disjoint (since otherwise we can use S and $T \setminus S$). When proving it, use the "law of total variance", which considers a random variable Z that is generated by first picking $x \leftarrow X$ and outputting Z_x, where X and the Z_x's are independent random variables. The said law asserts that the variance of Z (i.e., $\mathbb{V}[Z]$) equals $\mathbb{E}_{x\leftarrow X}[\mathbb{V}[Z_x]] + \mathbb{V}_{x\leftarrow X}[\mathbb{E}[Z_x]]$ (and its proof is via striaghtforward manipulations, which only use the definition of variance).[42] Now, assume, for simplicity of notation, that $S = [1, a]$ and $T = [a + 1, b]$, and consider selecting uniformly $w \in \{0,1\}^{\ell-b}$ and $(u, v) \in \{0,1\}^a \times \{0,1\}^{b-a}$. Then, we have

$$\begin{aligned}
0.5 \cdot \mathtt{I}_{S\cup T}(f) &= \mathbb{E}_{w\in\{0,1\}^{\ell-b}}[\mathbb{V}_{uv\in\{0,1\}^b}[f(uvw)]] \\
&= \mathbb{E}_{w\in\{0,1\}^{\ell-b}}\left[\mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]] + \mathbb{V}_{v\in\{0,1\}^{b-a}}[\mathbb{E}_{u\in\{0,1\}^a}[f(uvw)]]\right] \\
&\geq \mathbb{E}_{w\in\{0,1\}^{\ell-b}}\left[\mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]]\right] \\
&= \mathbb{E}_{vw\in\{0,1\}^{\ell-a}}\left[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]\right] \\
&= 0.5 \cdot \mathtt{I}_S(f)
\end{aligned}$$

In Part 2 we again assume that S and T are disjoint, but now the justification is by Part 1 (which implies $\mathtt{I}_{T\setminus S}(f) \leq \mathtt{I}_T(f)$). In the proof itself, using the same notations as in the proof of Part 1, we have

$$\begin{aligned}
0.5 \cdot \mathtt{I}_{S\cup T}(f) &= \mathbb{E}_{w\in\{0,1\}^{\ell-b}}\left[\mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]] + \mathbb{V}_{v\in\{0,1\}^{b-a}}[\mathbb{E}_{u\in\{0,1\}^a}[f(uvw)]]\right] \\
&\leq \mathbb{E}_{w\in\{0,1\}^{\ell-b}}\left[\mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]] + \mathbb{E}_{u\in\{0,1\}^a}[\mathbb{V}_{v\in\{0,1\}^{b-a}}[f(uvw)]]\right] \\
&= \mathbb{E}_{vw\in\{0,1\}^{\ell-a}}\left[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]\right] + \mathbb{E}_{uw\in\{0,1\}^{a+\ell-b}}\left[\mathbb{V}_{v\in\{0,1\}^{b-a}}[f(uvw)]\right] \\
&= 0.5 \cdot \mathtt{I}_S(f) + 0.5 \cdot \mathtt{I}_T(f)
\end{aligned}$$

where the inequality is proved by using the definition of variance.[43] Indeed, we could have assumed, w.l.o.g., that $b = \ell$ (and avoided taking the expectation over w), since for every $A \subseteq S \cup T$ it holds

[42]The proof is as follows

$$\begin{aligned}
\mathbb{V}[Z] &= \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 \\
&= \mathbb{E}_{x\leftarrow X}[\mathbb{E}[Z_x^2]] - \mathbb{E}_{x\leftarrow X}[\mathbb{E}[Z_x]]^2 \\
&= \mathbb{E}_{x\leftarrow X}[\mathbb{V}[Z_x] + \mathbb{E}[Z_x]^2] - \mathbb{E}_{x\leftarrow X}[\mathbb{E}[Z_x]]^2 \\
&= \mathbb{E}_{x\leftarrow X}[\mathbb{V}[Z_x]] + \mathbb{E}_{x\leftarrow X}[\mathbb{E}[Z_x]^2] - \mathbb{E}_{x\leftarrow X}[\mathbb{E}[Z_x]]^2 \\
&= \mathbb{E}_{x\leftarrow X}[\mathbb{V}[Z_x]] + \mathbb{V}_{x\leftarrow X}[\mathbb{E}[Z_x]]
\end{aligned}$$

where the last equality refers to a random variables that is assigned the value $\mathbb{E}[Z_x]$ with probability $\mathbf{Pr}[X = x]$.

[43]For any $w \in \{0,1\}^{\ell-b}$, letting $f_w(uv) = f(uvw)$, prove that

$$\mathbb{V}_{v\in\{0,1\}^{b-a}}[\mathbb{E}_{u\in\{0,1\}^a}[f_w(uv)]] \leq \mathbb{E}_{u\in\{0,1\}^a}[\mathbb{V}_{v\in\{0,1\}^{b-a}}[f_w(uv)]],$$

using $\mathbb{V}[Z] = \mathbb{E}[(Z - \mathbb{E}[Z])^2] = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2$ and $\mathbb{E}[Z]^2 \leq \mathbb{E}[Z^2]$ (which is implied by it). Specifically, we have

$$\begin{aligned}
\mathbb{V}_{v\in\{0,1\}^{b-a}}[\mathbb{E}_{u\in\{0,1\}^a}[f_w(uv)]] &= \mathbb{E}_v[(\mathbb{E}_u[f_w(uv)] - \mathbb{E}_{v'}[\mathbb{E}_u[f_w(uv')]])^2] \\
&= \mathbb{E}_v[(\mathbb{E}_u[f_w(uv)] - \mathbb{E}_u[\mathbb{E}_{v'}[f_w(uv')]])^2] \\
&= \mathbb{E}_v[\mathbb{E}_u[f_w(uv) - \mathbb{E}_{v'}[f_w(uv')]]^2] \\
&\leq \mathbb{E}_v[\mathbb{E}_u[(f_w(uv) - \mathbb{E}_{v'}[f_w(uv')])^2]] \\
&= \mathbb{E}_u[\mathbb{E}_v[(f_w(uv) - \mathbb{E}_{v'}[f_w(uv')])^2]] \\
&= \mathbb{E}_u[\mathbb{V}_v[f_w(uv)]]
\end{aligned}$$

that $\mathtt{I}_A(f) = \mathbb{E}_w[\mathtt{I}_A(f_w)]$, where $f_w(uv) = f(uvw)$.

Exercise 12 (the influence of sets has diminishing marginal gain): *Prove that for every $S, T, M \subseteq [\ell]$ and every f, it holds that*

$$\mathtt{I}_{S\cup T\cup M}(f) - \mathtt{I}_{S\cup T}(f) \leq \mathtt{I}_{S\cup M}(f) - \mathtt{I}_S(f).$$

Guideline: As shown next, we may focus on the case that S, T and M are disjoint. Considering only T and M that are disjoint of S is without loss of generality, since we may consider $T \setminus S$ and $M \setminus S$, respectively. Focusing on disjoint M and T is justified by monotonicity (i.e., Part 1 of Exercise 11). Furthermore, we can assume, w.l.o.g., that $S \cup T \cup M = [\ell]$ (see comment at the end of the guideline for Exercise 11).

Now, assume, for simplicity of notation, that $S = [1, a]$, $T = [a + 1, b]$, and $M = [b + 1, \ell]$, and consider selecting uniformly $(u, v, w) \in \{0,1\}^a \times \{0,1\}^{b-a} \times \{0,1\}^{\ell-b}$. Then, using Eq. (9), we have

$$\begin{aligned}
&0.5 \cdot \mathtt{I}_{S\cup M}(f) - 0.5 \cdot \mathtt{I}_S(f) \\
&\quad= \mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{uw\in\{0,1\}^{a+\ell-b}}[f(uvw)]] - \mathbb{E}_{vw\in\{0,1\}^{\ell-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]] \\
&\quad= \mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{w\in\{0,1\}^{\ell-b}}[\mathbb{E}_{u\in\{0,1\}^a}[f(uvw)]] + \mathbb{E}_{w\in\{0,1\}^{\ell-b}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]]] \\
&\qquad - \mathbb{E}_{vw\in\{0,1\}^{\ell-a}}[\mathbb{V}_{u\in\{0,1\}^a}[f(uvw)]] \\
&\quad= \mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{w\in\{0,1\}^{\ell-b}}[\mathbb{E}_{u\in\{0,1\}^a}[f(uvw)]]]
\end{aligned}$$

where the second equality uses the "law of total variance" (see guideline to Exercise 11). Similarly,

$$0.5 \cdot \mathtt{I}_{S\cup T\cup M}(f) - 0.5 \cdot \mathtt{I}_{S\cup T}(f) = \mathbb{V}_{w\in\{0,1\}^{\ell-b}}[\mathbb{E}_{uv\in\{0,1\}^b}[f(uvw)]].$$

Letting $g(vw) = \mathbb{E}_{u\in\{0,1\}^a}[f(uvw)]$, we have

$$\begin{aligned}
&0.5 \cdot \mathtt{I}_{S\cup M}(f) - 0.5 \cdot \mathtt{I}_S(f) \\
&\quad= \mathbb{E}_{v\in\{0,1\}^{b-a}}[\mathbb{V}_{w\in\{0,1\}^{\ell-b}}[g(vw)]] \\
&\quad\geq \mathbb{V}_{w\in\{0,1\}^{\ell-b}}[\mathbb{E}_{v\in\{0,1\}^{b-a}}[g(vw)]] \\
&\quad= 0.5 \cdot \mathtt{I}_{S\cup T\cup M}(f) - 0.5 \cdot \mathtt{I}_{S\cup T}(f)
\end{aligned}$$

where the inequality is proved by using the definition of variance (as in Footnote 43).

References

[1] L.M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography, In *20th FOCS*, pages 55–60, 1979.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Journal of the ACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *Journal of the ACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[4] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 804–915, 1998. Extended abstract in *36th FOCS*, 1995.

[5] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006. Extended abstract in *36th STOC*, 2004.

[6] E. Blais. Testing juntas almost optimally. In *41st ACM Symposium on the Theory of Computing*, pages 151–158, 2009.

[7] E. Blais. Testing juntas: a brief survey. In [17].

[8] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47, No. 3, pages 549–595, 1993. Extended abstract in *22nd STOC*, 1990.

[9] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1, No. 1, pages 3–40, 1991. Preliminary version in *31st FOCS*, 1990.

[10] R.A. DeMillo and R.J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, Vol. 7 (4), pages 193–195, June 1978.

[11] I. Dinur. The PCP Theorem by Gap Amplification. *Journal of the ACM*, Vol. 54 (3), Art. 12, 2007. Extended abstract in *38th STOC*, 2006.

[12] I. Dinur and P. Harsha. Composition of Low-Error 2-Query PCPs Using Decodable PCPs. *SIAM Journal on Computing*, Vol. 42 (6), pages 2452–2486, 2013. Extended abstract in *50th FOCS*, 2009.

[13] I. Dinur and O. Reingold. Assignment-testers: Towards a combinatorial proof of the PCP-Theorem. *SIAM Journal on Computing*, Vol. 36 (4), pages 975–1024, 2006. Extended abstract in *45th FOCS*, 2004.

[14] I. Dinur and S. Safra. The importance of being biased. In *34th ACM Symposium on the Theory of Computing*, pages 33–42, 2002.

[15] E. Fischer, G. Kindler, D. Ron, S. Safra and A. Samorodnitsky. Testing juntas. *Journal of Computer and System Science*, Vol. 68 (4), pages 753–787, 2004. Extended abstract in *44th FOCS*, 2002.

[16] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[17] O. Goldreich (ed.). *Property Testing: Current Research and Surveys.* Springer, LNCS, Vol. 6390, 2010.

[18] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In the proceedings of *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[19] O. Goldreich and M. Sudan. Locally testable codes and PCPs of almost-linear length. *Journal of the ACM*, Vol. 53 (4), pages 558–655, 2006. Extended abstract in *43rd FOCS*, 2002.

[20] O. Goldreich. Reducing testing affine subspaces to testing linearity. *ECCC*, TR16-080, May 2016.

[21] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th ACM Symposium on the Theory of Computing*, 1982.

[22] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, Vol. 182, pages 105–142, 1999. Preliminary versions in *28th STOC* (1996) and *37th FOCS* (1996).

[23] J. Hastad. Getting optimal in-approximability results. *Journal of the ACM*, Vol. 48, pages 798–859, 2001. Extended abstract in *29th STOC*, 1997.

[24] S. Khot. On the power of unique 2-prover 1-round games. In *34th ACM Symposium on the Theory of Computing*, pages 767–775, 2002.

[25] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM Journal on Computing*, Vol. 37 (1), pages 319–357, 2007. Extended abstract in *44th FOCS*, 2004.

[26] R. Merkle. *Secrecy, authentication, and public key systems.* Ph.D. dissertation, Department of Electrical Engineering, Stanford University 1979.

[27] D. Moshkovitz and R. Raz. Two-query PCP with subconstant error. *Journal of the ACM*, Vol. 57 (5), 2010. Extended abstract in *49th FOCS*, 2008.

[28] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *46th IEEE Symposium on Foundations of Computer Science*, pages 21–30, 2005.

[29] M. Parnas, D. Ron, and A. Samorodnitsky. Testing Basic Boolean Formulae. *SIAM Journal on Disc. Math. and Alg.*, Vol. 16 (1), pages 20–46, 2002.

[30] J. Pollard. Monte Carlo methods for index computations (mod p). *Math. Comp.*, Vol 32, pages 918–924, 1978.

[31] A. Samorodnitsky and L. Trevisan. A PCP Characterization of NP with Optimal Amortized Query Complexity. In *32nd ACM Symposium on the Theory of Computing*, pages 191–199, 2000.

[32] J.T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, Vol. 27 (4), pages 701–717, October 1980.

[33] R.E. Zippel. Probabilistic algorithms for sparse polynomials. In the *Proceedings of EUROSAM '79: International Symposium on Symbolic and Algebraic Manipulation*, E. Ng (Ed.), Lecture Notes in Computer Science (Vol. 72), pages 216–226, Springer, 1979.