# Reducing testing affine spaces to testing linearity*

Oded Goldreich†

May 20, 2016

**Abstract**

We consider the task of testing whether a Boolean function $f : \{0,1\}^\ell \to \{0,1\}$ is the indicator function of an $(\ell - k)$-dimensional affine space. An optimal tester for this property was presented by Parnas, Ron, and Samorodnitsky (*SIDMA*, 2002), by mimicking the celebrated linearity tester (of Blum, Luby and Rubinfeld, *JCSS*, 1993) and its analysis. We show that the former task (i.e., testing $(\ell-k)$-dimensional affine spaces) can be reduced to testing the linearity of a related function $g : \{0,1\}^\ell \to \{0,1\}^k$, yielding an almost optimal tester.

In addition, we show that testing monomials can be performed by using the foregoing reduction and reducing part of the analysis to the analysis of the dictatorship test.

## 1 Introduction

Revising our lecture notes on property testing[1] towards a forthcoming book, we were baffled by the tester of (monotone) $k$-monomials presented by Parnas, Ron, and Samorodnitsky [5]. This tester generalizes their tester of dictatorship, and does so by following the same strategy and using similar arguments at each step.

Specifically, the basic strategy consists of two steps. First, one tests whether the input function $f : \{0,1\}^\ell \to \{0,1\}$ defines an $(\ell - k)$-dimensional affine space, where the space defined by $f$ is $f^{-1}(1)$. In the case of dictatorship (i.e., $k = 1$), this amounts to testing whether $f$ is an affine function, but in the case of $k > 1$ a more general task arises. In the second step, one tests whether this affine space is of the right form (i.e., a translation by $1^\ell$ of a linear space spanned by axis-parallel vectors). In the case of $k = 1$ this amounts to testing whether the affine function depends on a single variable, but in the case of $k > 1$ another more general task arises.

Both these general tasks were solved by Parnas, Ron, and Samorodnitsky [5], but their solutions mimic the solutions used in the case of $k = 1$. Furthermore, in both cases, the generalization is very cumbersome. We find this state of affairs quite annoying, and believe that it is more appealing to reduce the general case to the special case.

*A preliminary version of this note was posted in April 2016 as a guest column on the *Property Testing Review*. The current version is more detailed and contains an additional result (presented in Section 5). In addition, in the current version, we first reduce the testing of affine subspaces to the testing of linear subspaces, and apply the core reduction to this special case.

†Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel.

[1] Available at `http://www.wisdom.weizmann.ac.il/~oded/pt-ln.html`

**The current revision.** We resolve a problem left open in our previous posting, of a couple of days ago. This new material, which refers to the problem of testing monomials, appears in Section 5. The prior text was adapted quite superficially.

**Organization.** In Section 2 we recall the standard definition of property testing and formally define the properties considered in this note. In Section 3 we detail the context of this note as outlined in the introduction. The reduction of testing affine spaces to testing linearity is presented in Section 4, whereas the problem of testing monomials is considered in Section 5. Both sections can be read independently of Sections 2 and 3.

## 2 Preliminaries

We assume that the reader is familiar with the basic definition of property testing, but for sake of good order we reproduce it here, while specializing it to the case of functions with domain $\{0,1\}^\ell$.

**Definition 1** (a tester for property $\Pi$): *Let $\Pi$ be a set of functions of the form $f:\{0,1\}^\ell\to\{0,1\}^*$. A* tester for $\Pi$ *is a probabilistic oracle machine, denoted $T$, that, on input parameter $\epsilon$ and oracle access to a function $f:\{0,1\}^\ell\to\{0,1\}^*$, outputs a binary verdict that satisfies the following two conditions.*

1. $T$ accepts inputs in $\Pi$: *For every $\epsilon>0$, and for every $f\in\Pi$, it holds that $\mathbf{Pr}[T^f(\epsilon)=1]\geq 2/3$.*
2. $T$ rejects inputs that are $\epsilon$-far from $\Pi$: *For every $\epsilon>0$, and for every function $f:\{0,1\}^\ell\to\{0,1\}^*$ that is $\epsilon$-far from $\Pi$ it holds that $\mathbf{Pr}[T^f(\epsilon)=0]\geq 2/3$, where $f$ is* $\epsilon$-far from $\Pi$ *if for every $g\in\Pi$ it holds that $|\{x\in\{0,1\}^\ell: f(x)\neq g(x)\}|>\epsilon\cdot 2^\ell$.*

*If the first condition holds with probability 1* (i.e., $\mathbf{Pr}[T^f(\epsilon)=1]=1$)*, then we say that $T$ has* one-sided error*; otherwise, we say that $T$ has* two-sided error*.*

We focus on the query complexity of such testers, while viewing $\ell$ as an additional parameter. We seek testers of query complexity that is independent of $\ell$, which means that the complexity will be a function of the proximity parameter $\epsilon$ and an auxiliary parameter $k$ (of the two properties that we consider). The properties we shall consider are formally defined next.

**Definition 2** (affine spaces): *A function $f:\{0,1\}^\ell\to\{0,1\}$* describes an $(\ell-k)$-dimensional affine space *if $f^{-1}(1)=\{x\in\{0,1\}^\ell: f(x)=1\}$ is an $(\ell-k)$-dimensional affine space; that is, $f^{-1}(1)=\{yG+s: y\in\{0,1\}^{\ell-k}\}$, where $G$ is an $(\ell-k)$-by-$\ell$ full-rank Boolean matrix and $s\in\{0,1\}^\ell$. When $s=0^\ell$, the* described space is linear*.*

**Definition 3** (linear functions): *For fixed $k$, we we say that $g:\{0,1\}^\ell\to\{0,1\}^k$ is* linear *if $g(x+y)=g(x)+g(y)$ for all $x,y\in\{0,1\}^\ell$. Equivalently, $g(z)=zT$ for a $\ell$-by-$k$ matrix $T$. We say that $f$ is* affine *if $f(z)=f'(z)+s$ for a linear fumnction $f'$ and some $s\in\{0,1\}^k$.*

## 3 Perspective: Testing monomials

As stated in the introduction, this note focuses on trying to reduce testing monomials to testing dictatorships. Such a reduction is sought in contrast to the extension of the ideas that underly the tests of (monotone) dictatorship towards testing the set of functions that are (monotone) $k$-monomials, for any $k \geq 1$. In this section we review the said extension, as performed in [5].

**Definition 4** (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. We focus on the task of testing monotone $k$-monomials, while recalling that the task of testing $k$-monomials is reducible to it (see [5] or our lecture notes). We also recall that the problem is of interest only when the proximity parameter, denoted $\epsilon$, is small (in relation to $2^{-k}$). In contrast, when $\epsilon > 2^{-k+2}$, we may just estimate the density of $f^{-1}(1)$ and accept if and only if the estimate is below $\epsilon/2$.

We start by interpreting the dictatorship tester of [1, 5] 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 $\ell$-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 $1^\ell$ of a linear space that is spanned by $\ell - 1$ unit vectors (i.e., vectors of Hamming weight 1).[2]

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 in [5].

**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 that is inversely proportional to the designated density (i.e., $2^{-k}$).)[3]

This task is related to linearity testing and it was indeed solved in [5] using a tester and an analysis that resembles the standard linearity tester of [2]. 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

[2] 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 $\ell$ unit vectors (i.e., vectors of Hamming weight 1).

[3] Recall that if $\epsilon < 2^{-k+2}$, then $O(2^k) = O(1/\epsilon)$, and otherwise (i.e., for $\epsilon \geq 2^{-k+2}$) testing affinity of $H$ reduces to estimating the density of $h^{-1}(1)$.

affine subspaces can be reduced to testing linearity, providing an alternative to the presentation of [5], and doing so is the core of this note (see Section 4).

**Testing that an affine subspace is a translation by $1^\ell$ 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 [5], 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 as part of the dictatorship tester of [1, 5], and the test and the analysis in [5] (for the case of $k > 1$) resemble the corresponding parts in [1, 5] (which handle the case of $k = 1$). Building on the main idea of Section 4, in Section 5, we present a simple reduction from the general case (of any $k \geq 1$) to the special case (of $k = 1$).

# 4 The reduction (of testing affine spaces to testing linearity)

We start by restating the problem. We are given access to a Boolean function $h : \{0,1\}^\ell \to \{0,1\}$ and wish to test whether $h^{-1}(1)$ is an $(\ell - k)$-dimensional affine subspace by reducing this problem to testing linearity. We present two reductions. The first (and simpler) reduction increases the complexities by a factor of $2^k$, whereas the second reduction only incurs an overhead of $\widetilde{O}(\log(1/\epsilon))$.

## 4.1 Simplifying assumptions

Recall that we may assume that $\epsilon = O(2^{-k})$, which means that $2^k = O(1/\epsilon)$, since the case of $\epsilon > 2^{-k+2}$ can be handled by estimating the density of $h^{-1}(1)$ (and accepting iff the estimate is below $\epsilon/2$).[4]

Another simplifying assumption is that we are dealing with a linear subspace. Actually, we present a reduction of the general case to this special case.[5]

**Claim 5** (reducing to the linear case): *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 a space) *can be reduced to testing*

[4] If $h$ is $0.75\epsilon$-close to the all-zero function, then it is $\epsilon$-close to describing a $(\ell - k)$-dimensional subspace, since $2^{-k} < \epsilon/4$. On the other hand, if $h$ is $0.25\epsilon$-far from the all-zero function, then it cannot describe a $(\ell-k)$-dimensional subspace, and it is OK to reject.

[5] This reduction somehat simplifies the presentation in Section 4.2, and more significantly so in Section 4.3.

*whether a Boolean function $h' : \{0,1\}^\ell \to \{0,1\}$ describes a $(\ell-k)$-dimensional linear subspace* (i.e., whether $\{x : h'(x)=1\}$ is such a space)*, where the reduction introduces an additive overhead of $O(2^k)$ queries.*

**Proof:** On input parameter $\epsilon > 0$ and oracle access to $h$, we proceed as follows.

1. Select uniformly a sample of $O(2^k)$ points in $\{0,1\}^\ell$. If $h$ evaluates to 0 on all these points, then reject. Otherwise, let $u$ be a point in this sample such that $h(u) = 1$.

2. Invoke the tester for linear subspaces on input parameter $\epsilon$ and oracle access to $h'$ defined by $h'(x) \stackrel{\text{def}}{=} h(x+u)$, and output its verdict. That is, each query $x$ to $h'$ is emulated by making the query $x+u$ to $h$.

If $h$ describes an $(\ell-k)$-dimensional affine subspace, then with high probability Step 1 finds $u \in h^{-1}(1)$, since $h^{-1}(1)$ has density $2^{-k}$, and we proceed Step 2. But in this case it holds that $h'(x) = 1$ if and only if $x+u \in h^{-1}(1)$, which means that $h'$ describes the $(\ell-k)$-dimensional linear space $h^{-1}(1) - u$.

On the other hand, if $h$ is $\epsilon$-far from being an $(\ell-k)$-dimensional affine subspace, then either Step 1 rejects or else $u \in h^{-1}(1)$. But in this case $h'$ defined by $h'(x) \stackrel{\text{def}}{=} h(x+u)$ must be $\epsilon$-far from describing an $(\ell-k)$-dimensional linear subspace (since if $h'$ is $\epsilon$-close to $g'$ that describes such a linear space, then $g(x) = g'(x-u)$ describes an affine space whereas $h$ is $\epsilon$-close to $g$). So, in this case, Step 2 rejects with high probability. ■

## 4.2 The first reduction

We define a function $g : \{0,1\}^\ell \to \{0,1\}^k \cup \{\perp\}$ such that if $H \stackrel{\text{def}}{=} h^{-1}(1)$ is an $(\ell-k)$-dimensional linear space, 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 linear 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. Using matrix notation, we restate this definition next (where the $v^{(i)}$'s are the rows of the matrix $V$).

**Definition 6** (the function $g = g_{H,V}$): *Let $V$ be a $k$-by-$\ell$ full rank matrix such that $cV \in H$ implies $c = 0^k$. Then, $g_{H,V} : \{0,1\}^\ell \to \{0,1\}^k \cup \{\perp\}$ is defined such that $g_{H,V}(x) = c$ if $c \in \{0,1\}^k$ is the unique vector that satisfies $x + cV \in H$, and $g_{H,V}(x) = \perp$ if the number of such vectors is not one.*

(Whenever we say that $g$ is linear, we mean, in particular, that it never assumes the value $\perp$.)[6]

**Claim 7** ($H$ versus $g_{V,H}$): *Let $V$ be as in Definition 6. Then, $H$ is an $(\ell-k)$-dimensional linear space if and only if $g = g_{H,V}$ is a surjective linear function.*

It follows that if $g$ is $\epsilon$-close to being a surjective linear function, then $g^{-1}(0^k)$ is $\epsilon$-close to being an ($(\ell-k)$-dimensional) linear space (i.e., the indicator functions of these sets are $\epsilon$-close).[7]

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

[7] Considering a linear surjective $g'$ that is $\epsilon'$-close to $g$, note that $H' = \{x : g'(x) = 0^k\}$ is $\epsilon$-close to $g^{-1}(0^k)$.

**Proof:** First note that $g^{-1}(0^k) \subseteq H$ always holds (since $g(x) = c$ implies $x + cV \in H$), and that equality holds when $g$ never assumes the value $\perp$ (since in this case $x + cV \in H$ implies that $g(x) = c$).

Now, on the one hand, if $g$ is a surjective linear function (i.e., $g(x) = xT$ for some full-rank $\ell$-by-$k$ matrix $T$), then $H = g^{-1}(0^k)$ (i.e., $H = \{x : xT = 0^k\}$), which implies that $H$ is an $(\ell - k)$-dimensional linear subspace.

On the other hand, if $H$ is an $(\ell - k)$-dimensional linear space, then, for some full-rank $(\ell - k)$-by-$\ell$ matrix $G$, it holds that $H = \{yG : y \in \{0,1\}^{\ell-k}\}$. In this case, for every $x$ there exists a *unique* representation of $x$ as $yG - cV$, since $V$ is a basis for a $k$-dimensional linear space that complements the $(\ell - k)$-dimensional linear space $H$, which implies $x + cV = yG \in H$, and so $g(x) = c$. It follows that $g$ is surjective (since $g(cV) = c$ for every $c$) and linear (since $g(yG + cV) + g(y'G + c'V) = c + c' = g((y + y')G + (c + c')V)$ for every $y, y'$ and $c, c'$). ■

**Claim 8** (finding $V$): *If $H$ is an $(\ell - k)$-dimensional linear space, then a matrix $V$ as underlying the definition of $g$ can be found* (w.h.p.) *by making $O(2^k)$ queries to $h$.*

**Proof:** The matrix $V$ can be found in $k$ iterations as follows. In the $i^{\text{th}}$ iteration we try to find a vector $v^{(i)}$ such that $\sum_j c_j v^{(j)} \notin H$ holds for every $(c_1, ..., c_i) \in \{0,1\}^i \setminus \{0^i\}$. In each trial, we pick $v^{(i)}$ at random, noting that the probability of success is $1 - 2^{i-1} \cdot 2^{-k}$, whereas the condition can be checked by making $2^{i-1}$ queries to $h$, since in the $i^{\text{th}}$ iteration it suffices to check the cases in which $c_i = 1$. ■

Combining the above two claims, the desired reduction follows (as detailed next). 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 $\epsilon < 2^{-k+2}$).[8] Recall that we have already justified the assumption $\epsilon = O(2^{-k})$. In fact, we are going to assume that $\epsilon \leq 0.1/t$ for some $t = O(2^k)$, by possibly resetting $\epsilon \leftarrow \min(\epsilon, 0.1/t)$.

**Algorithm 9** (testing whether $H$ is an $(\ell - k)$-dimensional linear space): *On input a proximity parameter $\epsilon \in (0,1)$ and oracle access to $h : \{0,1\}^\ell \to \{0,1\}$, specifying $H = h^{-1}(1)$, proceed as follows.*

1. Find an adequate matrix $V$: *Using $O(2^k)$ queries to $h$, try to find a $k$-by-$\ell$ full-rank matrix $V$ such that for any non-zero $c \in \{0,1\}^k$ it holds that $cV \notin H$. If such a matrix $V$ is found, then proceed to the next step. Otherwise, reject.*

2. Test whether the function $g = g_{H,V}$ is linear: *Invoke a linearity test with proximity parameter $\epsilon$, while providing it oracle access to the function $g = g_{H,V}$. 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.*

   *If the linearity tester accepts, then proceed to the next step. Otherwise, reject.*

3. Check whether $g$ is surjective: *Assuming that $g$ is $\epsilon$-close to linear, check whether it is surjective as follows.*

[8] Needless to say, we would welcome a one-sided error reduction. Recall that the case $\epsilon \geq 2^{-k+2}$ can be handled by density estimation. A complexity improvement for the main case (of $\epsilon < 2^{-k+2}$) appears in Section 4.3.

(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., $g(x) \in \{c, \perp\}$, which is likely to indicate $g(x) = c$ when $\mathbf{Pr}_x[g(x) = \perp] \leq \epsilon$).

*We stress that we do not compute* $g$ *at* $x$*, which would have required* $2^k$ *queries to* $h$*, but rather check whether* $g(x) \in \{c, \perp\}$ *by making a single query to* $h$ (i.e., we query $h$ at $x + cV$).

(Recall that if $g$ is linear and surjective, then each $c \in \{0,1\}^k$ has $2^{\ell-k}$ pre-images under $g$, and (w.h.p.) $S$ contains such a pre-image.)

*Recall that if* $g$ *is linear but not surjective, then its image has size at most* $2^{k-1}$.

Recalling that linearity testing has complexity $O(1/\epsilon)$, the complexity of the foregoing algorithm is $O(2^k) + 2^k \cdot O(1/\epsilon)$.

**Analysis of Algorithm 9 (assuming $\epsilon \leq 0.1/t$).** Suppose that $H$ is an $(\ell-k)$-dimensional linear space. Then, by Claim 8, with high probability (i.e., with probability $1 - (1 - 2^{-k})^{O(2^k)} > 0.99$), a suitable matrix $V$ will be found (in Step 1) and Step 2 will accept, since (by Claim 7) the function $g_{H,V}$ is surjective and linear. Likewise, Step 3 will accept with high probability (i.e., with probability $1 - (1 - 2^{-k})^{O(2^k)} > 0.99$). On the other hand, if $h$ is $\epsilon$-far from describing an $(\ell-k)$-dimensional linear space, then either no suitable matrix $V$ is found in Step 1 or $g_{H,V}$ is $\epsilon$-far from being surjective and linear (see discussion following Claim 7). Now, if $g_{H,V}$ is $\epsilon$-far from being linear, then with high probability Step 2 will reject. Otherwise, $g_{H,V}$ is $\epsilon$-close to a non-surjective linear function, and in this case Step 3 will reject with probability at least $0.5 - t \cdot \epsilon \geq 0.4$, since $\epsilon \leq 0.1/t$.

**Remark 10** (analysis of Algorithm 9 when $\epsilon = \Omega(1)$): *The foregoing analysis of Algorithm 9 only uses the hypothesis* $\epsilon \leq 0.1/t$ *in the analysis of Step 3 (for the case of "far" functions). Recall that* $t = \Omega(2^k)$ *must hold in order to guarantee acceptance of functions having the property (in Step 3). Here we show that for a sufficiently small constant* $\epsilon_0 > 0$ (e.g., $\epsilon_0 = 2^k/10t = \Omega(1)$)*, we can use a more refined analysis that holds also for any* $\epsilon \leq \epsilon_0$*. Actually, we first modify Step 3 such that if* $S$ *contains some* $x$ *such that* $h(x + cV) = 1$ *holds, then we accept if and only if* $c$ *is the only vector* $c'$ *that satisfies* $h(x + c'V) = 1$. (Indeed, this requires $2^k$ additional queries.)[9] *We focus on the analysis of Step 3. For each* $c \in \{0,1\}^\ell$*, let* $W_c$ *denote the set of* $x$*'s that satisfy the foregoing condition* (i.e., $c$ is the unique $k$-bit long vector that satisfies $h(x + cV) = 1$). *Now, if* $g = g_{H,V}$ *is* $\epsilon_0$*-close to a non-surjective linear function* $g'$*, then there exists a set* $B \subseteq \{0,1\}^k$ *of size at least* $2^{k-1}$ (i.e., the vectors that are not in the image of $g'$) *such that* $\sum_{c \in B} |W_c| \leq \epsilon_0 \cdot 2^\ell$ *(since* $\cup_{c \in B} W_c$ *must be conatined in* $\{x \in \{0,1\}^\ell : g(x) \neq g'(x)\}$*, whereas the* $W_c$*'s are disjoint). It follows that, in this case, Step 3 rejects with probability at least*

$$\begin{aligned} \mathbf{Pr}_c[c \in B] \cdot \mathbf{Pr}_{c,S}[S \cap W_c = \emptyset | c \in B] &= \frac{|B|}{2^k} \cdot \frac{1}{|B|} \sum_{c \in B} \left(1 - \mathbf{Pr}_{x \in \{0,1\}^\ell}[x \in W_c]\right)^t \\ &\geq 0.5 - \frac{1}{2^k} \sum_{c \in B} \frac{t \cdot |W_c|}{2^\ell} \end{aligned}$$

[9] We stress that we perform the check only for one $x \in S$ that satisfies $h(x + cV) = 1$, and act accordingly.

$$\geq \quad 0.5 - \frac{t \cdot \epsilon_0}{2^k}$$

*which is at least* 0.4*, provided* $\epsilon_0 \leq 2^k/10t$*.*

**Remark 11** (extension to affine spaces): *In light of Claim 5 there is no real need to extend Algorithm 9 to the affine case, but let us outline such an extension nevertheless. The definition of* $g$ *will be as in the linear case, except that it will be based on a fixed sequence of linearly independent vectors* $v^{(1)}, ..., v^{(k)} \in \{0,1\}^\ell$ *such that for some* $u \in H$ *and every non-empty* $I \subseteq [k]$ *it holds that* $u + \sum_{i \in I} v^{(i)} \notin H$*.* (Indeed, finding such a $u \in H$ is moved from Claim 5 to the revised algorithm.) *Next, Claim 7 can be extended to show that* $H$ *is an* $(\ell - k)$*-dimensional affine space if and only if* $g$ *is a surjective affine function. Lastly, testing the affinity of* $g$ *is reduced to testing the linearity of the mapping* $x \mapsto g(x) - g(0^\ell)$*.*

### 4.3 The second reduction

Let $h : \{0,1\}^\ell \to \{0,1\}$ be a Boolean function. In Section 4.2, we reduced $\epsilon$-testing whether $h^{-1}(1)$ is an $(\ell - k)$-dimensional linear subspace to $\epsilon$-testing the linearity of a function $g$, where the value of $g$ at any point can be computed by making $2^k$ queries to $h$. (Indeed, that reduction made $O(2^k)$ additional queries to $h$.) This yields an $\epsilon$-tester of time complexity $O(2^k/\epsilon)$ for testing $(\ell - k)$-dimensional linear subspaces, when considering basic operations on $\ell$-bit strings at unit cost. Recall that, for every $\epsilon_0 < 1/4$, if $g$ is $\epsilon_0$-close to being a linear function, then it is $\epsilon_0$-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 (where we assume again that $\epsilon \leq 2^{-k-O(1)}$).

**Step I:** Invoke Algorithm 9 with proximity parameter set to $\epsilon_0$, where $\epsilon_0 > 0$ is a constant as in Remark 10 and Step 3 is modified as in Remark 10. If the said invocation rejects, then reject. Otherwise, let $V$ be the matrix found in Step 1 of that invocation, and let $g = g_{H,V}$ be the corresponding function. Let $g'$ denote the linear function closest to $g$.

This step can be implemented in time $O(2^k)$.

**Step II:** Test whether $h$ equals $h'$, where $h'(x) = 1$ if and only if $g'(x) = 0^k$. Testing is done based on a sample of $O(1/\epsilon)$ points, as detailed next. Access to $h'$ is implemented by computing $g'$, which in turn is computed via self-correction such that the value of $g'$ at each desired point is implemented by making two random queries to $g$.

Note that if $h$ describes an $(\ell - k)$-dimensional linear subspace, then $g = g'$. On the other hand, if $h$ is $\epsilon$-far from this property and we reached the current step, then (as detailed below) $h$ is $\epsilon$-far from $h'$, and a sample of $O(1/\epsilon)$ random points will contain a point of disagreement. (This holds also if the sample points are only pairwise independent.)

The problem is that each query to $g$ is implemented by $2^k$ queries to $h$. Hence a straightforward implementation of Step II will result in making $O(2^k/\epsilon)$ queries to $h$, which is no better than Algorithm 9.

The key observation here is that Step II can be implemented 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 $O(m + 2^k \cdot \widetilde{O}(\log m))$ rather than $O(2^k \cdot m)$. This is done as follows.[10]

[10] Inspired by [4] (as presented in [3, Sec. 7.1.3]).

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 $J \subseteq [t]$. The key observations are that (1) the $r^{(J)}$'s are pairwise independent, and (2) the values of $g'$ at all $r^{(J)}$'s can be determined based on the values of $g'$ on the $s^{(j)}$'s. This determination is based on the fact that $g'(r^{(J)}) = \sum_{j \in J} g'(s^{(j)})$, by linearity of $g'$. Hence, the values of $g'$ on $t$ random points (i.e., the $s^{(j)}$'s) determines the value of $g'$ on $m = 2^t - 1$ pairwise independent points (i.e., the $r^{(J)}$'s).

**Theorem 12** (analysis of the foregoing algorithm): *The foregoing algorithm constitutes a tester for $(\ell - k)$-dimensional linear subspaces of time complexity $O(1/\epsilon) + \widetilde{O}(\log(1/\epsilon)) \cdot 2^k$.*

Recall that for $\epsilon > 2^{-k+2}$ there is an almost trivial tester of complexity $O(1/\epsilon)$, and note that for $\epsilon < 2^{-k-O(\log k)}$ it holds that $\widetilde{O}(\log(1/\epsilon)) \cdot 2^k = O(1/\epsilon)$.

**Proof:** Let us first take a closer look at the (time and query) complexity of the foregoing algorithm. As stated in Step I, this step can be implemented in time $O(2^k)$. In implementing Step II we need to query $h$ on $m = O(1/\epsilon)$ points (i.e., the $r^{(J)}$'s) and compare these values to the values of $g'$ on these points. The latter values are determined by the values of $g'$ at $t = \lceil \log_2(m+1) \rceil$ points, which in turn are obtained via self-correction. Recalling that we sought error probability $0.01/t$ for each of these values, each such self-correction uses $O(\log t)$ queries to $g$, where each query to $g$ is served by $2^k$ queries to $h$. So the total time is $O(1/\epsilon) + t \cdot O(\log t) \cdot 2^k$, which is $O(1/\epsilon) + \widetilde{O}(\log(1/\epsilon)) \cdot 2^k$.

If $h$ describes an $(\ell-k)$-dimensional linear subspace, then (w.h.p.) the execution reaches Step II, which always accepts. On the other hand, if $h$ is $\epsilon$-far from describing an $(\ell-k)$-dimensional linear subspace, then we consider several cases.

1. If $h$ is $\epsilon_0$-far from describing an $(\ell-k)$-dimensional linear subspace, then Step I rejects (w.h.p).

2. Otherwise, assuming that Step II is reached, we consider the corresponding functions $g$ and $g'$. By Remark 10, we may assume that $g'$ is surjective, since otherwise Step I rejects with probability at least 0.4. Hence, $g$ must be $\epsilon$-far from $g'$ (or else $h$ is $\epsilon$-close to $h'$, which contradicts the hypothesis, since in this case $h'$ describes an $(\ell - k)$-dimensional linear subspace).

3. We are left with the case that $g$ is $\epsilon$-far from $g'$. In this case (assuming Step II is reached), with probability at least 0.99, the tester obtains the correct values of $g'$ at all $s^{(j)}$'s and hence determined correctly the values of $g'$ at all the $r^{(j)}$'s. Since these $r^{(j)}$ are uniformly distributed in $\{0,1\}^\ell$ in a pairwise independent manner, with probability at least $1 - \frac{m\epsilon}{(m\epsilon)^2} > 0.9$, they contain a point on which $g$ and $g'$ disagree.

In conclusion, if $h$ is $\epsilon$-far from the tested property, then the foregoing algorithm rejects with probability at least 0.4. Using straightforward error reduction, the theorem follows. ■

**Remark 13** (extension to affine spaces): *Again, there is no real need to extend the foregoing to the affine case, but we outline such an extension nevertheless. We first note that self-correction of an affine $g$ requires querying it at three random locations rather than at two; specifically, to obtain $g'(s)$, we select uniformly $r, r' \in \{0,1\}^\ell$ and query $g$ at $r, r', r + r' + s$, while relying on $g(s) = g(r) + g(r') + g(r + r' + s)$. Likewise, the equality $g'(r^{(J)}) = \sum_{j \in J} g'(s^{(j)})$ holds only for $J$'s of odd size, and so we use only such $J$'s, which amounts to using $t = \lceil \log_2(2m) \rceil$ rather than $t = \lceil \log_2(m+1) \rceil$.*

# 5 Testing monotone monomials

As outlined in Section 3, the function $f : \{0,1\}^\ell \to \{0,1\}$ is a monotone $k$-monomial if and only if $f$ describes an $(\ell-k)$-dimensional affine space that is a translation by $1^\ell$ of an $(\ell-k)$-dimensional axis-parallel linear space; that is, if $f^{-1}(1)$ has the form $\{1^\ell + yG : y \in \{0,1\}^{\ell-k}\}$, where $G$ is a full-rank $(\ell-k)$-by-$\ell$ Boolean matrix that contains $k$ all-zero columns. Hence, we may focus on testing that the function $h : \{0,1\}^\ell \to \{0,1\}$ defined by $h(x) \stackrel{\text{def}}{=} f(x+1^\ell)$ describes an $(\ell-k)$-dimensional axis-parallel linear space. (Indeed, the reduction of Section 4.1 is instantiated here by mandating $u = 1^\ell$.)

Following [5], we first test that the Boolean function $h$ describes an $(\ell-k)$-dimensional linear space, and next test that this linear space has the right form. Again, we assume that the proximity parameter $\epsilon$ is upper-bounded by $\epsilon_0 \cdot 2^{-k}$, for some constant $\epsilon_0 > 0$. Hence, if $h$ passes the first test, then we may assume that $h$ is $\epsilon$-close to Boolean function $h'$ that describes an $(\ell-k)$-dimensional linear space. Defining corresponding functions $g : \{0,1\}^\ell \to \{0,1\}^k$ and $g' : \{0,1\}^\ell \to \{0,1\}^k$ as in Section 4.2, we infer that $g$ is $2^k \cdot \epsilon$-close to the surjective linear function $g'$, while noting that $2^k \cdot \epsilon \leq \epsilon_0$. (Indeed, we may use the function $g = g_{h^{-1}(1),V}$ defined in Step 1 of Algorithm 9, or just run this step anew.)

The key observation is that $h'$ describes an axis-parallel linear space (i.e., the set $\{x \in \{0,1\}^\ell : h'(x) = 1\}$ equals the linear space $\{yG : y \in \{0,1\}^{\ell-k}\}$ for a full-rank matrix $G$ with $k$ all-zero columns) if and only if $g'$ is a projection function (i.e., $g'(x) = x_I$ for some $k$-subset $I$). At this point, we generalize the observation that underlies the conjunction test that is part of the dictatorship test of [1, 5]. Specifically, given that $g'$ is a surjective linear function, we test that $g'(x) = x_I$ (for some $k$-subset $I$) by selecting uniformly $r, s \in \{0,1\}^\ell$ and checking whether $g'(r)g'(s) = g'(rs)$, where $uv$ denotes the bit-by-bit produce of $u$ and $v$. The point is that the analysis of this test can be reduced to the analysis of the conjunction test by considering the $k$ bits in the output of $g'$. Details follow.

Let $g'_i(x)$ denote the $i^{\text{th}}$ bit of $g'(x)$. On the one hand, if $g'(x) = x_I$ for some $k$-subset $I$, then for each $i \in [k]$ the function $g'_i$ is a dictatorship (i.e., $g'_i(x) = x_{j_i}$ for some $j_i \in [\ell]$), and so $g'_i(rs) = (rs)_{j_i} = g'_i(r)g'_i(s)$ for all $r, s \in \{0,1\}^\ell$. Hence, in this case, $g'(rs) = g'(r)g'(s)$ holds for all $r, s$. On the one hand, if $g'$ is not a projection function, then (using the fact that $g'$ is linear and surjective) there exists $i \in [k]$ such that the function $g'_i$ is a linear combination of at least two bits. The known analysis (cf. [1, 5]) implies that in this case $\mathbf{Pr}_{r,s}[g'_i(rs) = g'_i(r)g'_i(s)] \leq 3/4$, which implies $\mathbf{Pr}_{r,s}[g'(rs) = g'(r)g'(s)] \leq 3/4$.

However, as in Section 4, we do not have access to $g'$, but rather obtain its values at desired points by applying self-correction to $g$, which is $\epsilon_0$-close to $g'$. We can afford to compute $g$ at any desired point, since we intend to do so only a constant number of times. Specifically, it suffices to perform the foregoing "conjunction" test once, since such an execution rejects an improper $h$ (i.e., one that is close to a function $h'$ that describes a linear space that is not axis-parallel) with probability at least $0.25 - 4\epsilon_0 > 0.2$ (assumning $\epsilon_0 > 0$ is sufficiently small). To wrap-up, we obtain the following tester (where we assume again that $\epsilon \leq \epsilon_0/2^k$, for some adequate constant $\epsilon_0 > 0$).

**Algorithm 14** (testing whether $f$ is a monotone $k$-monomial): *On input a proximity parameter $\epsilon \in (0, \epsilon_0 \cdot 2^{-k}]$ and oracle access to $f : \{0,1\}^\ell \to \{0,1\}$, the algorithm proceeds as follows.*

1. *Apply the tester of Section 4.3 to test whether $h : \{0,1\}^\ell \to \{0,1\}$, defined by $h(x) \stackrel{\text{def}}{=} f(x+1^\ell)$, describes an $(\ell-k)$-dimensional linear space. The said algorithm is invoked with proximity*

*parameter $\epsilon$, and each query $x$ is answered by the value $f(x+1^\ell)$. If the foregoing algorithm rejects, then the current algorithm reject.*

2. *Find a matrix $V$ as in Step 1 of Algorithm 9, and let $g = g_{h^{-1}(1),V} : \{0,1\}^\ell \to \{0,1\}^k$ denote the corresponding function. Select uniformly $r, s, w \in \{0,1\}^\ell$ and accept if and only if $g(r)g(s) = g(rs+w) - g(w)$, where $g(rs+w) - g(w)$ represents self-correcting the value of $g$ at $rs$.* (Recall that $uv$ denotes the bit-by-bit produce of $u$ and $v$, and that the value of $g$ at $x$ is computed by making $2^k$ queries to $h$.)

Note that, with probability at least $1-4\epsilon_0$ (over the choice of $r, s$ and $w$), it holds that $g(r) = g'(r)$, $g(s) = g'(s)$ and $g(rs+w) - g(w) = g'(rs+w) - g'(w) = g'(rs)$.

## Acknowledgements

I am grateful to Roei Tell for reading prior versions of this text and pointing out numerous inaccuracies and gaps. I also wish to thank Clement Canonne and Tom Gur for their comments. This research was partially supported by the Israel Science Foundation (grant No. 671/13).

## References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.