To appear: SIAM J. on Computing, manuscript no. 0255151.

Robust Characterizations of Polynomials with Applications to Program Testing*

Ronitt Rubinfeld †Madhu Sudan ‡

Abstract

The study of self-testing and self-correcting programs leads to the search for robust characterizations of functions. Here we make this notion precise and show such a characterization for polynomials. From this characterization, we get the following applications. We construct simple and efficient self-testers for polynomial functions. Our characterizations provide results in the area of coding theory, by giving extremely fast and efficient error-detecting schemes for some well known codes. This error-detection scheme plays a crucial role in subsequent results on the hardness of approximating some NP-optimization problems.

1 Introduction

The study of program checkers [Blu88][BK89], self-testing programs [BLR90] and self-correcting programs [BLR90][Lip91] was introduced in order to allow one to use a program P to compute a function without trusting that P works correctly. A *program checker* checks that the program gives the correct answer on a particular input, a *self-testing program* for f tests that program P is correct on most inputs, and a *self-correcting program* for f takes a program P that is correct on most inputs and uses it to compute f correctly on every input with high probability. The program checker, self-tester and self-corrector may call the program

*This paper unifies and extends part of the results contained in Gemmell et al. [GLRSW91] and Rubinfeld and Sudan [RS92].

†Cornell University. email: ronitt@cs.cornell.edu. This work is supported by ONR Young Investigator grant N00014-93-1-0590 and the United States–Israel Binational Science Foundation grant 92-00226. Part of this work was done while the author was at Princeton University, supported by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), NSF-STC88-09648.

‡I.B.M. Thomas J. Watson Research Center. email: madhu@watson.ibm.com. Part of this work was done when the author was a student at the University of California at Berkeley under the support of NSF PYI Grant CCR 8896202.

as a black box, are required to do something other than to actually compute the function, and should be much simpler and at least different from any program for the function f in the precise sense defined by [BK89]. It is straightforward to show that checkers, self-testers and self-correctors for functions are related in the following way: If f has a self-tester and a self-corrector, then it can be shown that f has a program result checker. Conversely, if f has a checker, then it has a self-tester (though not necessarily a self-corrector). It is argued in [BK89] and [BLR90] that this provides an attractive alternative method for attacking the problem of program correctness.

One of the main goals of the research in the area of self-testing/correcting programs and program checking is to find general techniques for finding very simple and efficient self-testers, self-correctors and checkers for large classes of problems. In fact, some success towards this goal has been achieved. For example, in [BK89], it is shown how to use techniques from the area of interactive proof systems in order to write checkers. Using these and other techniques, checkers (and hence self-testers) have been found for a variety of problems [AHK, BK89, Rub90, Kan90, BFLS91, BF91]. If a function is random self-reducible, i.e., the value of the function at any input can be inferred from its value at randomly chosen inputs, then it has a self-corrector [BLR90][Lip91]. This provides self-correctors for a surprising range of functions, including the class of linear functions (homomorphisms between groups) and polynomials.

In the direction of characterizing functions that have self-testers, some success has been achieved in [BLR90]. They give a number of methods of constructing self-testers for functions, some of which we mention here: They observe that any checker for a function can be used to construct a self-tester for the function. They present a particular method of constructing self-testers for a variety of functions based on a method of bootstrapping from tests over smaller domains. They also show another method of constructing self-testers for all linear functions, i.e., functions that act as homomorphisms between groups, in other words satisfy $f(x) + f(y) = f(x + y)$ for a group operation $+$.

The main focus of this paper is to study and understand the functions which have self-testers, and to broaden the class of functions that are known to have self-testers. The linearity tester of [BLR90] is the starting point for this paper. A particularly interesting feature of this linearity tester is that it breaks the task of self-testing a function into the two tasks of (1) testing it for certain "structural properties" and (2) using the structural property to then identify the function precisely. In this paper we introduce a new notion – a function family tester – which helps delineate these two tasks more clearly. We first introduce some terminology:

We work with functions defined over some finite domain $\mathcal{D}$. The *distance* between two functions f and g over the domain $\mathcal{D}$ is the fraction of points $x \in \mathcal{D}$ where the two functions disagree:

$$d(f, g) \equiv \frac{|\{x \in \mathcal{D} | f(x) \neq g(x)\}|}{|\mathcal{D}|}$$

We say that two functions are ϵ-*close* if $d(f, g) \leq \epsilon$. In some of the informal discussions that follow, we drop the ϵ and just describe two functions as being close. In such cases, it is

implied that we are talking of some small enough ϵ. In terms of this notion a self-tester for a function f may be defined as follows:

A ϵ-self-tester T for a function f over a domain $\mathcal{D}$, is a (randomized) oracle program that takes as input a program P and behaves as follows:

- Accepts P if $d(P, f) = 0$.
- Rejects P (with high probability) if P and f are not ϵ-close.
- Behaves arbitrarily otherwise.

Testers for function families using robust characterizations Let $\mathcal{F}$ be a family of functions. An *ϵ-function family tester* T for the family $\mathcal{F}$, takes as input a program P and tests if there exists a function $f \in \mathcal{F}$ such that P is ϵ-close to f.

The notion of a function family tester captures the notion of verifying properties of a function as follows: Let $\mathcal{P}$ be a property we wish to test for. Let $\mathcal{F}$ be the family of all functions that have the property $\mathcal{P}$. Then a function family tester for $\mathcal{F}$ can be used to test if a program P "essentially" has the property $\mathcal{P}$ (i.e., there exists a function with property $\mathcal{P}$ that is close to P). To make some of these abstract definitions concrete, let us work with the simple example of the property of linearity among functions from $\mathcal{Z}_p$ to $\mathcal{Z}_p$. For this example, the family of functions we work with is $\mathcal{F}_{\text{linear}} \equiv \{f_a | a \in \mathcal{Z}_p, f_a(x) = a \cdot x\}$. Thus a tester for the family of linear functions verifies that the computation of a program P is essentially linear.

The existence of a function family tester for any class of functions implies a powerful characterization of the family. In particular, consider any program that is rejected by the tester. In order to reject the program, the tester will have found some evidence in the small set of sampled points which "proves" that P can not be a member of $\mathcal{F}$. In other words, all members of $\mathcal{F}$ must satisfy some property on the set of inputs that are examined by the family tester. Thus all members of $\mathcal{F}$ satisfy a "local" property (by local we mean a property on a set of small size – we define this notion more formally in Section 2). Moreover, if all such local properties are satisfied, then the tester accepts the function, implying that these local constraints form a characterization of the family. Thus in order for a function family to have a tester, it needs to have a local characterization. In our example, such a local characterization of linear functions is the property that $\forall x, y \in \mathcal{Z}_p$, $f(x) + f(y) = f(x + y)$. If a function is not linear then there exists a counterexample of size three that proves that it is not linear.

However, local characterizations do not form a sufficient condition for the construction of testers. Typically an exact local characterization of a family of functions involves a universal quantification, which is not feasible to verify. In our example, the characterization of linear functions by the property $\forall x, y \in \mathcal{Z}_p$, $f(x)+f(y) = f(x+y)$ is not useful to test a purported linear function since we cannot hope to efficiently test that this holds for all possible pairs x, y. Thus for a characterization to be useful for testing, it needs to be "robust", involving the words "for most" rather than "for all". Specifically, let $\mathcal{F}$ be the function family that

satisfies the properties at all inputs, and let f be any function that satisfies the properties at most inputs. Then f must be close to some $g \in \mathcal{F}$ (see Section 2 for a more formal definition). In our example, if $f(x) + f(y) = f(x + y)$ is satisfied by f for most x, y, then $f(x) = c \cdot x$ for most x and some constant c.

Our results on function family testing One of the main emphases of this paper is to find robust characterizations for the family of low degree univariate and multivariate polynomials. In Section 3 we start by describing some (well-known) local characterizations of univariate and multivariate polynomials and then prove that some of these characterizations are actually robust characterizations. As an immediate consequence we get function family testers for all low-degree polynomials over finite fields. For the case of polynomials over Z_p, our testers are very simple and do not even need to multiply elements of the field. Our testers are the first testers that directly attempt to test the total degree of a polynomial (as opposed to the testers of [BFLS91, FGLSS91, AS92], all of which test that the degree in each variable is not too large). The proof of correctness of our tester also is different from the proofs of correctness of the other testers in that it does not rely on an inductive argument based on the number of variables. This allows for its "efficiency" to be independent of the number of variables and provides the hope for the existence of a tester with nearly optimal efficiency.

A second emphasis of this paper is the notion of test sets that allows us to use the results on function family testing to obtain self-testers for specific functions. Informally, a test set is a set of points from the domain, such that no two functions from the family $\mathcal{F}$ agree with each other on all the points from the test set. Our self-tester for a specific function f would require, as a description of f, its value on all points in a test set. The complexity (running time) of the self-tester will depend on the size of the test set.

Other implications of low-degree testing The task of constructing family testers for the family of low-degree polynomials is closely related to the task of error-detection in Reed Solomon codes. In fact, a low-degree test can be described as a "randomized" error-detector that determines whether the number of errors in a received word is small or not. In this sense, the error-detectors we construct have the feature that they are highly efficient and can be used to get estimates on the distance of a received word from a valid codeword. This perspective can similarly be applied to the results of [BLR90] to get randomized error-detecting and correcting schemes for the Hadamard codes that probe the received word in only a constant number of bits to detect an error or find any bit of the codeword closest to the received word. In fact, it has been shown by M. Naor [Nao92] that these results can be used to construct codes for which error-detection/correction can be performed by uniform quasi-polynomial sized circuits of constant depth. In Section 7 we define the notion of a "locally testable code" - a notion that precisely describes the relationship between testing and error-correcting codes. We also provide applications of our testers to the construction of "locally testable codes" in the section.

A different perspective on the construction of family testers is to view it as the following approximation problem:

Given a family of functions $\mathcal{F}$ and a function P, estimate the distance $d(P, \mathcal{F})$ between P and $\mathcal{F}$ to within a small multiplicative error.

A tester for a function family $\mathcal{F}$ essentially yields such an approximator (provided $d(P, \mathcal{F})$ is smaller than half) by defining some new quantities $\delta(P, \mathcal{F})$ that are easy to estimate by random sampling and then showing that some approximate relations hold between $\delta(P, \mathcal{F})$ and $d(P, \mathcal{F})$. For example, the linearity test of [BLR90] may be viewed as trying to approximate the distance $d(f, \mathcal{F}_{\text{linear}})$. To approximate this distance they define the quantity $\delta(f, \mathcal{F}_{\text{linear}}) \equiv \Pr[f(x) + f(y) \neq f(x+y)]$ which is easy to approximate. Then they show that $\delta(f, \mathcal{F}_{\text{linear}})/3 \leq d(f, \mathcal{F}_{\text{linear}}) \leq 9/2\delta(f, \mathcal{F}_{\text{linear}})$. The testers given here define similar quantities related to low-degree polynomials and show similar approximate relationships. Such inequalities may be of independent interest.

The task of low-degree testing forms a central ingredient in the proof of MIP = NEXPTIME due to [BFL91]. The tester given here provides an alternate mechanism that works in their setting. The efficiency of low-degree testing also becomes very important to the ensuing results on hardness of approximations [FGLSS91, ALMSS92] and therefore a lot of attention has been paid to this problem [BFL91, BFLS91, FGLSS91, AS92]. However all these results focus on tests that are close variants of the test given in [BFL91]. The low-degree test given here is fundamentally different from the ones mentioned above and originated from independent considerations in the work of [GLRSW91]. The efficiency of the tester shown here may also be found in [RS92]. It turns out that this tester is particularly well-suited to such multiple prover applications and provides a one round, constant prover proof that a function is a low degree polynomial over finite fields. This is observed in subsequent work of [ALMSS92] (see also [Sud92]) and follows by using an improved analysis for Lemma 11 from [AS92]. This turns out to play a crucial role in the NP = PCP($\log n, O(1)$) result of [ALMSS92], which in turn provides hardness results for a wide variety of approximation problems. An exact description of the relevance of the various testers and the chronology of contributions maybe found in Section 8.

Organization of Paper The rest of this paper is organized as follows. In Section 2 we formally define the notions of local characterizations - exact and robust. Section 3 lists some (well-known) exact characterizations of low-degree polynomials. Sections 4 and 5 show that two of these exact characterizations are robust. In Section 6 we describe the applications of these characterizations to self-testing of programs. In Section 7 we define a notion of locally-testable codes (based on the notion of probabilistically checkable proofs) and show applications of our testers to such codes. Section 8 contains some concluding remarks.

2 Local Characterizations: Exact and Robust

In this section we make precise the notion of a local characterization and what we mean by exact and robust characterizations. We will also isolate a parameter associated with the robust characterizations that captures the efficiency of the tester suggested by the characterization.

We will use $\mathcal{D}$ to represent a finite domain. We will consider here, families of functions $\mathcal{F}$ where $f \in \mathcal{F}$ maps elements from $\mathcal{D}$ to a range $\mathcal{R}$. We illustrate these definitions using the example of linear functions. Here the domain and range are $\mathcal{Z}_p$ and the family of functions is $\{f_a | a \in \mathcal{Z}_p$ where $f_a(x) = a \cdot x\}$.

Definition 1 (Neighborhoods) *A k-*local neighborhood *N is an ordered tuple of (not necessarily distinct) k points from $\mathcal{D}$. A k-local collection of neighborhoods $\mathcal{N}$ is a set of k-local neighborhoods.*

Definition 2 (Properties) *A k-*local property *$\mathcal{P}$ is a function from $(\mathcal{D} \times \mathcal{R})^k$ to $\{0,1\}$. We say that a function f satisfies a property $\mathcal{P}$ over a neighborhood N if $\mathcal{P}(\{(x, f(x))\}_{x \in N}) = 1$.*

Definition 3 (Exact Characterizations) *A property $\mathcal{P}$ over a collection of neighborhoods $\mathcal{N}$ is an* exact characterization *of a family of functions $\mathcal{F}$ if a function f satisfies $\mathcal{P}$ over all neighborhoods $N \in \mathcal{N}$ exactly when $f \in \mathcal{F}$. The characterization is k-local if the property $\mathcal{P}$ (and the collection $\mathcal{N}$) is k-local.*

In our example, the collection of neighborhoods $\mathcal{N} = \{(x, y, x+y) | x, y \in \mathcal{Z}_p\}$. The property $\mathcal{P}$ is 3-local and is satisfied by f on the triple (x_1, x_2, x_3) if $f(x_1) + f(x_2) = f(x_3)$. Thus over the collection of neighborhoods $\mathcal{N}$, $\mathcal{P}$ gives a 3-local characterization of the family of linear functions.

Definition 4 (Robust Characterizations) *A property $\mathcal{P}$ over a collection of neighborhoods $\mathcal{N}$ is said to be an (ϵ, δ)-*robust characterization *of $\mathcal{F}$, if whenever a function f satisfies $\mathcal{P}$ on all but δ fraction of the neighborhoods in $\mathcal{N}$, it is ϵ-close to some function $g \in \mathcal{F}$. Moreover, all members of $\mathcal{F}$ satisfy $\mathcal{P}$ on all neighborhoods in $\mathcal{N}$.*

To continue with the example of linear functions, the theorem of [BLR90] can be used to say that $\mathcal{P}$ over the neighborhood $\mathcal{N}$ is a $(\frac{9}{2}(\frac{2}{9} - \alpha), \frac{2}{9} - \alpha)$-robust characterization of linear functions for any constant α.

The exact constant ϵ determining closeness is not very important for the family of multivariate polynomials. For most of the characterizations we consider here, it can be shown that any function f is $((1 + o(1))\delta)$-close to some member g of $\mathcal{F}$ if f is $\frac{1}{4}$-close to g and violates only a δ fraction of the neighborhood constraints. Thus for the purposes of this paper, we fix the value of ϵ to be $\frac{1}{4}$.

In order to test if f is close to some member of $\mathcal{F}$, one would need to sample at least $\frac{1}{\delta}$ of the neighborhoods in $\mathcal{N}$ and test if $\mathcal{P}$ holds on these neighborhoods. Hence, the parameter $\frac{1}{\delta}$ is referred to as the *efficiency* of the characterization.

3 Exact Characterizations of Polynomials

In this section we start by describing some (well-known) exact local characterizations of polynomial functions. In later sections we will show that some of these characterizations can be made robust.

The family of degree d polynomials can be characterized in a number of ways. The different characterizations arise from looking at different collections of neighborhoods $\mathcal{N}$. The property P has to remain invariant in the following sense: P will be satisfied by f on a neighborhood N if there exists a polynomial that agrees with f on all points in N. The complexity of a neighborhood test, i.e., testing whether a constraint is being satisfied by a neighborhood, is also influenced by the choice of the neighborhood. Thus by choosing the characterizations appropriately, we might be able to tradeoff the simplicity of the neighborhood test against the number of times the test needs to be repeated. The different characterizations also have to be qualified by different restrictions on the underlying ring. For instance, some characterizations hold only for finite fields while others hold only for rings of the form $\mathcal{Z}_m$. We will take care to point out the restrictions on the characterizations. We give examples of possible neighborhoods and their corresponding tests.

1. **Univariate polynomials**

   The following characterization of univariate polynomials holds for a function f mapping a ring R to itself.

   (a) Characterization: $f : R \mapsto R$ is a polynomial of degree at most d if and only if $\forall x_0, \ldots, x_{d+1} \in R$, there exists a polynomial $g_{x_0,\ldots,x_{d+1}}$ of degree at most d such that $f(x_i) = g_{x_0,\ldots,x_{d+1}}(x_i)$.

   (b) Neighborhood structure: $\mathcal{N} = R^{d+2}$, i.e., all possible (multi)-subsets of R of size $d+2$.

   (c) Complexity of neighborhood test: A test of the above nature involves finding the (unique) degree d polynomial g that agrees with f at the points $x_0, \ldots, x_d$ and then evaluating $g(x_{d+1})$ and verifying that this equals $f(x_{d+1})$. Standard interpolation techniques (see, for instance, [dW70]) imply that this is equivalent to computing coefficients $\alpha_0, \ldots, \alpha_{d+1}$, where the $\{\alpha_i\}$'s depend only on the $\{x_i\}$'s, and verifying that $\sum_{i=0}^{d+1} \alpha_i \cdot f(x_i) = 0$. The α_i's can be computed using elementary algorithms with $O(d^2)$ additions, subtractions and multiplications over R.

2. **Univariate polynomials using evenly spaced points**

   This characterization works over the ring $\mathcal{Z}_m$. Let $\alpha_i = \binom{d+1}{i}(-1)^{i+1}$. The interpolation identity for degree d polynomials on evenly spaced points, $x, x+h, \ldots, x+(d+1)\cdot h$, reduces to $\sum_{i=0}^{d+1} \alpha_i f(x + i \cdot h) = 0$. We refer to x as the *starting point* and h as the *offset*.

   (a) Characterization: $f : \mathcal{Z}_m \mapsto \mathcal{Z}_m$ is a polynomial of degree at most d if and only if $\forall x, h \in \mathcal{Z}_m$, $\sum_{i=0}^{d+1} \alpha_i f(x + i \cdot h) = 0$.

(b) Neighborhood structure: Define the neighborhood sets $N_{x,h} \equiv \{x+i\cdot h\}_{i=0}^{d+1}$. Then the neighborhood collection is $\mathcal{N} = \bigcup_{x,h\in Z_m} N_{x,h}$.

(c) Complexity of Neighborhood Test: Notice that the constants α_i are now independent of x and h and can be precomputed once and for all. In fact, due to the special relationship between the α_i's, given the value of f at the points $x + i \cdot h$, we can compute the above summation with $O(d^2)$ additions and subtractions and *no multiplications* (see appendix).

3. **Multivariate polynomials using lines**

This characterization applies to m-variate functions over a finite field F. Define the notion of a *line* through the space F^m as follows: For $\hat{x}, \hat{h} \in F^m$, the line $l_{\hat{x},\hat{h}}$ through $\hat{x}$ with offset $\hat{h}$ is the set of points $\{\hat{x} + i \cdot \hat{h} | i \in F\}$. We will often refer to the line in its parametric form $l_{\hat{x},\hat{h}}(i)$. Observe that a polynomial f of total degree d, restricted to a line $l_{\hat{x},\hat{h}}(i)$ becomes a univariate polynomial of degree at most d in the parameter i. This gives us the following characterization of degree d polynomials over sufficiently large finite fields ($|F| \geq 2d + 1$). [1]

(a) Characterization: The function $f : F^m \mapsto F$ is a polynomial of degree at most d if and only if $\forall \hat{x}, \hat{h} \in F^m$, f restricted to $l_{\hat{x},\hat{h}}(i)$ is a univariate polynomial in i of degree at most d (see appendix for a proof).

(b) Neighborhood Structure: Let the neighborhoods be lines. Then $\mathcal{N} \equiv \{N_{\hat{x},\hat{h}} = l_{\hat{x},\hat{h}} | \hat{x}, \hat{h} \in F^m\}$.

(c) Complexity of Neighborhood Test: In this form the characterization is not very local since the counterexamples are lines, i.e., collections of $|F|$ points. But this characterization is interesting to us because it says that the characterization of multivariate polynomials can be reduced to the characterization of univariate polynomials (on these lines). Thus we find that we can now use, for instance, Characterization 1 to find counterexamples of size at most $d + 2$. The complexity of a neighborhood test here is no more than the complexity of the neighborhood test in Characterization 1.

4. **Multivariate polynomials using axis parallel lines**

This characterization is a specialization of the characterization above, in terms of special lines - *axis parallel lines*. We say that a line is *axis parallel* if the offset $\hat{h}$ contains only one non-zero coordinate.

(a) Characterization: $f : F^m \mapsto F$ is a polynomial of degree at most d in each variable if and only if $\forall$ axis parallel lines, f restricted to the line is a univariate polynomial of degree at most d. Notice that here we characterize polynomials differently, i.e., in terms of individual degree in each variable rather than total degree.

[1] The above characterization is not the tightest possible in its requirement of the parameter $|F|$. Indeed, for the case of fields of prime order this can be improved to the optimal case $|F| \geq d + 2$ and this has been shown recently in [FS94]. For arbitrary finite fields it turns out that $|F| \geq d + 2$ is not a sufficient condition for this characterization to hold. A counterexample to this effect is also shown in [FS94].

(b) Neighborhood Structure: The neighborhoods here are sets of the form $N_{i,\hat{\beta}} \equiv \{(\beta_1, \ldots, \beta_{i-1}, t, \beta_i, \ldots, \beta_{m-1}) | t \in F\}$, for every choice of $\hat{\beta} \in F^{m-1}$, and every choice of $i \in \{1, \ldots, m\}$. Then $\mathcal{N} = \bigcup_{i \in \{1,\ldots,m\}, \hat{\beta} \in F^{m-1}} N_{i,\hat{\beta}}$.

(c) Complexity of Neighborhood Test: The complexity of a neighborhood test is the same as the complexity of Characterization 1.

5. **Multivariate polynomials: evenly spaced points**

A combination of Characterizations 2 and 3 gives the following characterization of polynomials over $\mathcal{Z}_p$, provided p is large enough for Characterization 3 to hold.

(a) Characterization: $f : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ is a polynomial of degree at most d if and only if $\forall \hat{x}, \hat{h} \in \mathcal{Z}_p^m$, $\sum_{i=0}^{d+1} \alpha_i f(\hat{x} + i\hat{h}) = 0$, where $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$.

(b) Neighborhood Structure: The neighborhoods here are of the form $N_{\hat{x},\hat{h}} \equiv \{\hat{x} + i\hat{h} | i \in \{0, \ldots, d+1\}\}$. Then $\mathcal{N} \equiv \bigcup_{\hat{x},\hat{h} \in \mathcal{Z}_p^m} N_{\hat{x},\hat{h}}$.

(c) Complexity of Neighborhood Test: The complexity of this neighborhood test is the same as the complexity in Characterization 2.

6. **Multivariate polynomials: evenly spaced points - 2**

This characterization is a trivial consequence of the characterization above, and seems weaker since its neighborhood structure is larger than those of the ones above. But it turns out that this characterization is much more useful due to the kind of robustness it yields. This characterization holds for polynomials over $\mathcal{Z}_p$, for $p > 10d$.

(a) Characterization: $f : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ is a polynomial of degree at most d if and only if $\forall \hat{x}, \hat{h} \in \mathcal{Z}_p^m$, the values of f at the points $\{\hat{x} + i\hat{h} | i \in \{0, \ldots, 10d\}\}$ agree with some univariate polynomial g of degree at most d in t.

(b) Neighborhood Structure: The neighborhoods here are sets of the form $N_{\hat{x},\hat{h}} \equiv \{\hat{x} + i\hat{h} | i \in \{0, \ldots, 10d\}\}$. Then $\mathcal{N} \equiv \bigcup_{\hat{x},\hat{h} \in \mathcal{Z}_p^m} N_{\hat{x},\hat{h}}$.

(c) Complexity of the neighborhood test: Once again it turns out that the complexity of this test is within a constant factor of the complexity of the test in Characterization 2, i.e., $O(d^2)$ additions and subtractions and no multiplications (see appendix).

All characterizations above turn out to be robust. The robustness of Characterization 1 is straightforward and omitted here (see, for instance, [Sud92]). The robustness of 4 follows from the work of [BFL91] (see also [BFLS91, FGLSS91, AS92, Lun92]). Robustness of 2, 3, 5 and 6 are presented in Sections 4 and 5.

A typical robust characterization theorem for degree d polynomials in m variables over a finite field F would go as follows:

There exists a δ_0 (which may be a function of $d, m, |F|$) such that for $\delta \leq \delta_0$, if the fraction of neighborhoods where P satisfies the local constraints is at least $1-\delta$, then P is ϵ-close to some degree d polynomial (where ϵ is some function of δ).

An important parameter in determining the efficiency of a tester, is the relationship between δ_0 and $m, d, |F|$. For instance, if $\delta_0 = \frac{1}{dm \log |F|}$, then this implies that we will have to test that the local property holds for at least $dm \log |F|$ randomly chosen neighborhoods before we can satisfy ourselves that P is close to some polynomial. Our main thrust will be to get a theorem that holds for as high a δ_0 as possible. [2]

In what follows, we show first that Characterization 5 above is robust with $\delta_0 = \Theta(\frac{1}{d^2})$. This proof gives a simple and efficient tester for the family of multivariate polynomials that works with $O(d^3)$ probes into f. Robustness of the characterizations in 2 and 3 follow as special cases. This bound on δ_0 is in contrast to the robustness of 4 that has an inherent dependency on m.

Next we show a robustness of Characterization 6. The efficiency of this test is analyzed modulo the efficiency of a certain test for bivariate polynomials and is shown to be within a constant factor of the bivariate test. We also show that the efficiency of the bivariate test is $O(d)$, giving a test for multivariate polynomials that works with $O(d^2)$ probes into f.

4 A Robust Characterization of Polynomial Functions

In this section, we prove the robustness of Characterization 5. We consider a function (program) P mapping m variables from $\mathcal{Z}_p$ to $\mathcal{Z}_p$ and prove the following:

Theorem 5 *For $\delta_0 = \frac{1}{2(d+2)^2}$, if $P : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ satisfies*

$$\delta \equiv \Pr_{x, h \in_R \mathcal{Z}_p^m} \left[P(x) \neq \sum_{i=1}^{d+1} \alpha_i P(x + i \cdot h) \right] \leq \delta_0$$

then there exists a degree d polynomial $g : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ that is 2δ-close to P.

This theorem makes very minimal requirements on the field size required for its validity. The theorem is valid whenever Characterization 5 holds and Friedl and Sudan, [FS94], have shown that this holds for $p \geq d+2$ - the smallest conceivable field size for which the test could be defined. We do not know of other testers that work with such a minimal requirement on the field size.

We define $g(x)$ to be $\text{maj}_{h \in Z_p^m} \{\sum_{i=1}^{d+1} \alpha_i P(x + ih)\}$, where maj of a set is the function that picks the element occurring most often (choosing arbitrarily in the case of ties). First we show that g is 2δ-close to P. Later in this section we show that g is a low-degree polynomial.

[2] A secondary parameter of interest is the relationship between ϵ and δ. In all the proofs that follow, we will only show that $\epsilon = 2\delta$. Actually, once such a result is shown it can be shown again that any $\epsilon > \delta$ works.

Lemma 6 *g and P agree on more than $1-2\delta$ fraction of the inputs from Z_p^m.*

Proof: Consider the set of elements x such that $\Pr_h[P(x) = \sum_{i=1}^{d+1} \alpha_i P(x+i*h)] < 1/2$. If the fraction of such elements is more than 2δ then it contradicts the condition that $\Pr_{x,h}[\sum_{i=0}^{d+1} \alpha_i P(x+i*h) = 0] = \delta$. For all remaining elements, $P(x) = g(x)$. $\square$

In the following lemmas, we show that the function g satisfies the interpolation formula for all x, h and is therefore a degree d polynomial. We do this by first showing that for all x, $g(x)$ is equal to the interpolation of P at x by most offsets t. We then use this to show that the interpolation formula is satisfied by g for all x, h.

Lemma 7 *For all $x \in Z_p^m$, $\Pr_h[g(x) = \sum_{i=1}^{d+1} \alpha_i P(x+i*h)] \geq 1-2(d+1)\delta$.*

Proof: Observe that $h_1, h_2 \in_R Z_p^m$ implies

$$x + i*h_1 \in_R Z_p^m \text{ and } x + j*h_2 \in_R Z_p^m$$

$$\Rightarrow \Pr_{h_1,h_2}[P(x+i*h_1) = \sum_{j=1}^{d+1} \alpha_j P(x+i*h_1+j*h_2)] \geq 1-\delta$$

$$\Rightarrow \Pr_{h_1,h_2}[P(x+j*h_2) = \sum_{i=1}^{d+1} \alpha_i P(x+i*h_1+j*h_2)] \geq 1-\delta$$

Combining the two we get

$$\begin{aligned} \Pr_{h_1,h_2}[\textstyle\sum_{i=1}^{d+1} \alpha_i P(x+i*h_1) &= \sum_{i=1}^{d+1}\sum_{j=1}^{d+1} \alpha_i\alpha_j P(x+i*h_1+j*h_2) \\ &= \sum_{j=1}^{d+1} \alpha_j P(x+j*h_2)] \\ &\geq 1-2(d+1)\delta \end{aligned}$$

The lemma now follows from the observation that the probability that the same object is drawn twice from a set in two independent trials lower bounds the probability of drawing the most likely object in one trial: Suppose the objects are ordered so that p_i is the probability of drawing object i, and $p_1 \geq p_2 \geq \ldots$. Then the probability of drawing the same object twice is $\sum_i p_i^2 \leq \sum_i p_1 p_i = p_1$. $\square$

Lemma 8 *For all $x, h \in Z_p^m$, if $\delta \leq \frac{1}{2(d+2)^2}$, then $\sum_{i=0}^{d+1} \alpha_i g(x+i*h) = 0$ (and thus g is a degree d polynomial [dW70]).*

Proof: Observe that, since $h_1 + ih_2 \in_R Z_p^m$, we have for all $0 \leq i \leq d+1$

$$\Pr_{h_1,h_2}[g(x+i*h) = \sum_{j=1}^{d+1} \alpha_j P((x+i*h)+j*(h_1+ih_2))] \geq 1-2(d+1)\delta$$

Furthermore, we have for all $1 \leq j \leq d+1$

$$\Pr_{h_1,h_2}[\sum_{i=0}^{d+1} \alpha_i P((x+j*h_1)+i*(h+j*h_2)) = 0] \geq 1-\delta$$

Putting these two together we get

$$\Pr_{h_1,h_2}[\sum_{i=0}^{d+1}\alpha_i g(x+i*h)=\sum_{j=1}^{d+1}\alpha_j\sum_{i=0}^{d+1}\alpha_i P((x+j*h_1)+i*(h+j*h_2))=0]>0$$

The lemma follows since the statement "$\sum_{i=0}^{d+1}\alpha_i g(x+i*h)=0$" is independent of h_1,h_2, and therefore if its probability is positive, it must be 1. □

Proof (of Theorem 5): Theorem 5 follows from Lemmas 6 and 8 □

5 Efficient tester for polynomials

In this section we prove the robustness of Characterization 6. Recall that this characterization uses the collection of neighborhoods $\mathcal{N}=\{N_{x,h}|x,h\in\mathcal{Z}_p^m\}$ where $N_{x,h}=(x,x+h,\ldots,x+10dh)$. The following theorem shows that the efficiency of this characterization is $O(d)$, i.e., if a function satisfies the consistency test on all but a $O(\frac{1}{d})$ fraction of the neighborhoods then it is close to a polynomial.

Theorem 9 *There exists a constant c such that for $0\le\delta\le\frac{1}{cd}$, if f is a function from $\mathcal{Z}_p^m$ to $\mathcal{Z}_p$ that satisfies the neighborhood consistency test on all but a δ fraction of the collection of neighborhoods $\mathcal{N}=\{N_{x,h}|x,h\in\mathcal{Z}_p^m\}$ (where $N_{x,h}=\{x,x+h,\ldots,x+10dh\}$), then there exists a polynomial $g:\mathcal{Z}_p^m\to\mathcal{Z}_p$ of total degree at most d such that $d(f,g)\le(1+o(1))\delta$ (provided $p>10d$.)*

In the rest of this section we prove this theorem for the case $d\ge 1$. (The case $d=0$ amounts to proving that f is a constant and is omitted as a straightforward exercise.)

Fix a function f that satisfies the neighborhood constraints on all but a δ fraction of the neighborhoods.

The proof follows the same basic outline as the one in Section 4, but in order to achieve the better efficiency, we use ideas that can be thought of in terms of error-correction. Thus many of the steps that were quite simple in Section 4 require more work here. In Section 4 the function g was defined to be the value that occurs most often (for most h) when one looks at the evaluation at x of the unique polynomial that agrees with the values of f at $x+h,...,x+(d+1)h$. Here we view the values of a polynomial at $x+h,...,x+10dh$ as a code word. Intuitively, the values of f at $x+h,...,x+10dh$ will often have enough good information in it to allow us to get back to a correct codeword. The function g defined below can be thought of as the value that occurs most often (for most h) when one looks at the polynomial defined by the *error correction* of the values of f at $x,x+h,\ldots,x+10dh$ evaluated at x. We then show that g has the following properties:

1. $g(x)=f(x)$ with probability at least $1-\delta-o(\delta)$ if x is picked randomly from $\mathcal{Z}_p^m$.
2. On every neighborhood $N_{x,h}$, g is described by a univariate polynomial of degree d.

Notice that Characterization 6 now implies that g is a degree d polynomial.

Notation: For $x, h \in \mathcal{Z}_p^m$, we let $P_{x,h}(i)$ be (the unique) polynomial in i that satisfies $P_{x,h}(i) = f(x + ih)$ for at least $6d$ values of $i \in \{0, \ldots, 10d\}$. If no such polynomial exists then $P_{x,h}$ is defined to be `error`.

$$\text{Let } g : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p \text{ be } g(x) \equiv \text{plurality}_h\{P_{x,h}(0)\}$$

where the plurality is taken over P's that are not `error`.

In Section 4 it is shown that if one computes the value of a polynomial function at x by interpolating from the values of the function along offset h_1 that are in turn computed by interpolating from the values of the function along offset h_2, then one would get the same answer as if one had computed the value of the function at x by interpolating from the values of the function along offset h_2 which in turn are computed by interpolating from the values of the function along offset h_1. This is not hard to see because it turns out that an interpolation is a weighted summation and obtaining the identity amounts to changing the order of a double summation (see for instance Lemma 7). Here g is actually an interpolation of the *error-correction* of the values of the function, which is no longer a simple algebraic function of the observed values. We repair the situation by constructing a bivariate polynomial $Q(i, j)$ that agrees with $f(x + i \cdot h_1 + j \cdot h_2)$ for most values of i and j. This allows us to get back to simple interpolation where we work with the function $Q(i, j)$ rather than f. Lemma 10 shows when such a bivariate polynomial can be set up to agree with a matrix of values m_{ij}. Lemma 11 shows how to use this polynomial to simulate the effect of the interchange in the order of the summation.

The following lemma follows directly from the axis parallel characterization of polynomials.

Lemma 10 *For $X, Y \subset \mathcal{Z}_p$ with $|X|, |Y| > d + 2$, if $\{r_i\}_{i \in X}$ and $\{c_j\}_{j \in Y}$ are univariate (degree d) polynomials such that for all $i \in X$ and $j \in Y$, $r_i(j) = c_j(i)$, then there exists a polynomial $Q(.,.)$ such that for all i, j $Q(i, j) = r_i(j) = c_j(i)$.*

Lemma 11 (Matrix Polynomial Lemma) *Given families of univariate degree d polynomials $\{r_i\}_{i=0}^{10d}$ and $\{c_j\}_{j=0}^{10d}$ and a matrix $\{m_{ij}\}_{i=0,j=0}^{10d,10d}$ that satisfy:*

- *For 90% of the i's in $\{0, \ldots, 10d\}$, $r_i(j) = m_{ij}$ for all $j \in \{0, \ldots, 10d\}$.*
- *For 90% of the j's in $\{0, \ldots, 10d\}$, $c_j(i) = m_{ij}$ for all $i \in \{0, \ldots, 10d\}$.*

Then there exists a bivariate polynomial $Q(\cdot, \cdot)$ of degree d in each variable such that for all $i_0, j_0 \in \{0, \ldots, 10d\}$ the following holds:

- *For at least 90% of the i's in $\{0, \ldots, 10d\}$, $Q(i, j_0) = m_{ij_0}$.*
- *For at least 90% of the j's in $\{0, \ldots, 10d\}$, $Q(i_0, j) = m_{i_0 j}$.*

Proof: Let X be the set of good rows of M, i.e., those with the property that $r_i(j)$ equals $c_j(i)$ for all values of $j \in \{0, \ldots, 10d\}$. Similarly, let Y be the set of good columns. It can now be seen that the conditions of Lemma 10 are applicable, implying that there exists a polynomial $Q(i,j)$ such that $Q(i,j) = r_i(j) = c_j(i)$ for all $(i,j) \in X \times Y$, where $|X|$ and $|Y|$ are both at least $9d$. But for any $i \in X$, there exists a unique polynomial describing all the elements in its row and Q agrees with it on 90% of its elements. Thus, for $i \in X$, $Q(i,j) = m_{ij}$ for all $j \in \{0, \ldots, 10d\}$. In particular this holds for $j = j_0$, i.e., for all $i \in X$, $Q(i, j_0) = m_{ij_0}$. Similarly by using the columns indexed by Y one can show that $Q(i_0, j) = m_{i_0 j}$ for all $j \in Y$. The lemma follows since the cardinalities of X and Y are at least $9d$. $\square$

The following lemmas are analogous to Lemmas 6, 7 and 8 of Section 4. Lemma 12 and Corollary 13 roughly correspond to Lemma 7. Lemma 12 essentially states that the plurality in the definition of g is actually an overwhelming majority. This may be obtained by setting $i_0 = 0$ in the statement of the lemma. The slightly stronger statement used here is needed later. Lemma 14 is similar to Lemma 6 and shows that g and f agree at all but a $\delta + o(\delta)$ fraction of the places. Lemma 15 shows that g is a multivariate polynomial of degree d.

Lemma 12 *There exists a constant c_1 such that for $\delta_1 = c_1\delta$, the following holds:*

$$\forall x \in F^m, i_0 \in \{0, \ldots, 10d\}, \quad \Pr_{h_1,h_2}\left[P_{x,h_1}(i_0) = P_{x+i_0 h_1, h_2}(0)\right] \geq 1 - \delta_1$$

Proof: Pick $h_1, h_2 \in_R \mathcal{Z}_p^m$ and define $M = \{m_{ij}\}$ to be the matrix given by $m_{ij} = f(x + ih_1 + jh_2)$. We show that M satisfies the conditions required by Lemma 11 (with $j_0 = 0$), with probability at least $1 - \delta_1$. This suffices to prove the lemma since this implies that the polynomial P_{x,h_1} is the polynomial $Q(i,j)$ restricted to $j = 0$ and that $P_{x+i_0 h_1, h_2}$ is $Q(i_0, j)$. Thus $P_{x,h_1}(i_0) = P_{x+i_0 h_1, h_2}(0) = Q(i_0, 0)$.

Any row of the matrix, other than the 0th row, represents a random neighborhood (independent of x) and satisfies the neighborhood constraint with probability $1 - \delta$. Thus with probability at least $1 - 10\delta$ we have that the fraction of rows that don't have a degree d polynomial describing them is at most 0.1. An analogous argument can be made for the columns. Thus M satisfies the conditions required by Lemma 11 with probability at least $1 - 20\delta$. The lemma is satisfied with the choice of $c_1 = 20$. $\square$

Corollary 13 *For $x \in \mathcal{Z}_p^m, i \in \{0, \ldots, 10d\}$, $\Pr_h\left[g(x + ih) = P_{x,h}(i)\right] \geq 1 - 2\delta_1$.*

Proof: Let B be the set of h's that violate $P_{x,h}(i) = \text{majority}_{h_1}\{P_{x+ih,h_1}(0)\}$. For all $h \notin B$ notice that $g(x + ih) = P_{x,h}(i)$. Also for h in B, the probability, for a randomly chosen h_1, that $P_{x+ih,h_1}(0) \neq P_{x,h}(i)$ is at least $1/2$. Thus with probability at least $\frac{|B|}{2p^m}$, we find that a randomly chosen pair (h, h_1) violates the condition $P_{x+ih,h_1}(0) = P_{x,h}(i)$. Applying Lemma 12 we get that $\frac{|B|}{p^m}$ is at most $2\delta_1$. $\square$

We next show that g and f agree in most places:

Lemma 14 $d(f,g) \leq \delta(1+o(1))$.

Proof: Let B' be the set of x's that satisfy $f(x) \neq P_{x,h}(0)$ for at least $1-2\delta_1$ fraction of the h's in $\mathcal{Z}_p^m$. Observe that due to Corollary 13, for all $x \notin B'$, $f(x)$ is the same as $g(x)$ ($g(x)$ can disagree with $P_{x,h}(0)$ on at most $2\delta_1$ fraction of the h's). The size of B' as a fraction of $\mathcal{Z}_p^m$ can be at most $\frac{\delta}{1-2\delta_1}$. Thus we find that $d(f,g) \leq \frac{\delta}{1-2\delta_1} = \delta(1+o(1))$. $\square$

Notation: For $x,h \in \mathcal{Z}_p^m$, we define $P_{x,h}^{(g)}(i)$ to be (the unique) polynomial in i that satisfies $P_{x,h}^{(g)}(i) = g(x+ih)$ for at least $9d$ values of $i \in \{0,\ldots,10d\}$. If no such polynomial exists then $P_{x,h}^{(g)}$ is defined to be `error`.

Lemma 15 *There exists a constant γ such that if $\delta \leq \frac{1}{\gamma d}$ then $\forall x,h \quad g(x) = P_{x,h}^{(g)}(0)$.*

Proof: As in the proof of Lemma 12 we will pick a convenient matrix on which we will apply Lemma 11. This time the matrix of choice is obtained by picking $h_1, h_2 \in_R \mathcal{Z}_p^m$ and letting $m_{ij} = g(x+ih+j(h_1+ih_2))$.

We will now show that Lemma 11 can be applied to this matrix with high probability (for $i_0 = j_0 = 0$). Observe that every row $\{m_{ij}\}_{j=0}^{10d}$ represents a random neighborhood containing the fixed point $x+ih$ and hence Corollary 13 implies that $P_{x+ih,h_1+ih_2}(j)$ agrees with m_{ij} for any choice of j with probability $1-2\delta_1$. Thus, for every i, with probability at least $1-2cd\delta_1$, $P_{x+ih,h_1+ih_2}(j)$ agrees with m_{ij} for all but $\frac{1}{cd}$ fraction of the j's. Thus with probability at least $1-22cd\delta_1$, this holds for at least 90% of the rows, including the row $i=0$. By picking $c > 10$ we satisfy the conditions required of the rows in Lemma 11. A similar argument based on the columns shows that the conditions required of the columns are also true with probability $1-20cd\delta_1 - o(1)$ (all columns except for the 0th one represent random neighborhoods). Thus the conditions required for Lemma 11 are satisfied with probability at least $1-42cd\delta_1 - o(1)$.

Applying Lemma 11 we find that there exists a bivariate polynomial $Q(i,j)$ such that it agrees with m_{i0} for 90% of the i's. Thus $P_{x,h}^{(g)}(i) = Q(i,0)$. We now argue that $m_{00} = Q(0,0)$ and this will complete the proof, since $m_{00} = g(x)$.

By Lemma 11 we find that $m_{0j} = Q(0,j)$ for 90% of the j's, implying $P_{x,h_1}^{(g)}(j) = Q(0,j)$. By Corollary 13 we also find that $m_{00} = P_{x,h_1}(0)$ with probability at least $1-2\delta_1$. In order to show that this equals $Q(0,0)$ it now suffices to show that $P_{x,h_1}^{(g)}(\cdot) = P_{x,h_1}(\cdot)$.

This last part follows from the following observation: For $j \neq 0$, $x+jh_1$ is distributed uniformly over F^m and thus with probability $1-(1+o(1))\delta$ we have $g(x+jh_1) = f(x+jh_1)$ (by Corollary 13). Hence with probability at least $1-10\delta-o(1)$, $g(x+jh_1) = f(x+jh_1)$ for 90% of the j's. But both the polynomials $P_{x,h_1}(j)$ and $P_{x,h_1}^{(g)}(j)$ agree with $f(x+jh_1)$ and $g(x+jh_1)$ for 90% of the j's respectively. Thus $P_{x,h_1}^{(g)}(\cdot)$ must agree with $P_{x,h_1}(\cdot)$ on at least 80% of the inputs, implying $P_{x,h_1}^{(g)}(\cdot) = P_{x,h_1}(\cdot)$.

Thus with probability at least $1-(42cd\delta_1 + 2\delta_1 + 10\delta + o(1))$ (over random choices of h_1 and h_2) the identity $g(x) = P_{x,h}^{(g)}(0)$ holds. But this event is deterministic (independent of h_1 and

h_2) and hence if its probability is positive then it must always hold. If $\delta < 1/((20)(541)d)$, then $\delta_1 < 1/(541d)$ and then the above probability is positive. □

Proof (of Theorem 9): Lemma 15 implies that along each line $l_{x,h}$, g can be described by a univariate polynomial of degree at most d. Characterization 6 can now be applied to infer that g is a polynomial of total degree at most d. From Lemma 14 we now know that f and g differ in at most $\delta(1 + o(1))$ fraction of the places. This completes the proof. □

6 Self-Testing Polynomials

In this section we complement the results of [BF90][Lip91] by showing how to construct a self-tester for any polynomial function. The results can also be generalized to give self-testers and self-correctors for functions in finite dimensional function spaces that are closed under shifting and scaling.

Previously, program testing was thought of as the following: *pick a random input x and verify that $P(x) = f(x)$ by computing f via another program.* This method has two problems: first, it relies on believing the other program to be correct, and secondly, since testing is often done at runtime [BLR90], it negates the benefits of designing faster programs, since the computation time will be dominated by the computation time of the old program.

As in [BLR90], our testers are of a nontraditional form and use the robust characterization of the function being tested: the tester is given a short specification of the function in the form of properties that the function must have, and verifies that these properties "usually" hold. We show that these properties are such that if the program "usually" satisfies these properties, then it is essentially computing the correct function.

Test Sets Given that a function computes a polynomial, we want a way of specifying that it is the correct polynomial. We do this by specifying the function value of the polynomial at a number of inputs. It is easy to see that the number of inputs required is exactly the number of inputs necessary to determine whether two degree d polynomials are distinct. Since any two degree d univariate polynomial functions can only agree on d points, it suffices to check whether or not the polynomial functions agree at *any* $d+1$ points to determine whether or not they are distinct. On the other hand, distinct multivariate polynomials can agree at an unbounded number of points. However, it is well known that there *exists* a set of $(d+1)^m$ points such that no two degree d, m-variate polynomials can agree at all points in the set. We make the following definition:

Definition 16 *We say that $\mathcal{T} = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ is a (d, m)-*polynomial test set *if there is only one degree d, m variable polynomial f such that for all $i \in [1, ..., t]$, $f(x_i) = y_i$.*

A (d, m)-test set need only be of size $(d+1)^m$.

When the number of variables is small, the provision that the value of the function is known on at least $(d+1)^m$ points is not very restrictive since the degree is assumed to be small with

respect to the size of the field: Suppose one has a program for the RSA function $x^3 \bmod m$. Traditional testing requires that the tester know the value of $f(x)$ for random values of x. Here one only needs to know the following simple and easy to generate specification: f is a degree 3 polynomial in one variable, and $f(0) = 0, f(1) = 1, f(-1) = -1, f(2) = 8$. These function values are the same over any ring Z_m of size at least 9.

6.1 Testing Algorithm

Our self-tester for a polynomial of degree d with m variables assumes that the specification of the polynomial is given by the value of the function on a (d, m)-polynomial test set.

Theorem 17 *If f is a degree d polynomial in m variables over $\mathcal{Z}_p$, and the value of f is given on a (d, m)-polynomial test set, then for $\epsilon \leq O(1/d^2)$, f has an $(\frac{\epsilon}{2(d+2)}, 4\epsilon)$-self-tester on $\mathcal{Z}_p$ with $O((d+1)^m/\epsilon + d \cdot \max(d^2, \frac{1}{\epsilon}))$ calls to P.*

The self-testing is done in two phases, one verifying that the program is essentially computing some degree d polynomial function g, and the other verifying that the g is the correct polynomial function by verifying that g (rather than P) is correct on the polynomial test set.

We now give the self-testing program that is used to prove Theorem 17.

For simplicity, in the description of our self-testing program, we assume that whenever the self-tester makes a call to P, it verifies that the answer returned by P is in the proper range, and if the answer is not in the proper range, then the program notes that there is an error.

We use $x \in_R Z_p^m$ to denote that x is chosen uniformly at random in Z_p^m.

```
program Polynomial-Self-Test(P, ε, β, T = ((x_1, f(x_1)), ..., (x_t, f(x_t))))

Degree Test
   Repeat Θ((1/ε) log(1/β)) times
         Pick x, h ∈_R Z_p^m and test that Σ_{i=0}^{d+1} α_i P(x + i*h) = 0
   Reject P if the test fails more than an ε fraction of the time.

Equality Test
   for j going from 1 to t do
      Repeat Θ(log(d/β)) times
         Pick h ∈_R Z_p^m and test that f(x_j) = Σ_{i=1}^{d+1} α_i P(x_j + i*h).
      Reject P if the test fails more than 1/4th of the time.
```

6.2 Correctness of Algorithm

Notation: Let $\delta \equiv \Pr_{x,h}[\sum_{i=0}^{d+1} \alpha_i P(x + i * h) \neq 0]$
We say program P is ϵ-good if $\delta \leq \frac{\epsilon}{2}$ and $\forall j \in \{1, \ldots, t\}$, $\Pr_h[f(x_j) = \sum_{i=1}^{d+1} \alpha_i P(x_j + i * h)] \geq 3/4$. We say P is ϵ-bad if either $\delta > 2\epsilon$ or if $\exists j$ such that $\Pr_h[f(x_j) = \sum_{i=1}^{d+1} \alpha_i P(x_j + i * h)] < 1/2$. (Note that there are programs that are neither ϵ-good or ϵ-bad).

The following lemma is easy to prove :

Lemma 18 *With probability at least $1 - \beta$ an ϵ-good program is passed by Polynomial-Self-Test. With probability at least $1 - \beta$ an ϵ-bad program is rejected by Polynomial-Self-Test.*

It is easy to see that if a program P $\frac{\epsilon}{2(d+2)}$-computes f, then it is ϵ-good. On the other hand, we need to show that if P does not 4ϵ-compute f then it is ϵ-bad. We show the contrapositive, i.e. that if P is not ϵ-bad, then it 4ϵ-computes f.

If P is not ϵ-bad, then $\delta \leq 2\epsilon$. Under this assumption, we show that there exists a function g with the following properties:

1. $g(x) = P(x)$ for most x.
2. $\forall x, t \quad \sum_{i=0}^{d+1} \alpha_i g(x + it) = 0$, and thus g is a degree d polynomial.
3. $g(x_j) = f(x_j)$ for $j \in \{0, 1, \ldots, d\}$.

The function g is as defined in the previous section on robust characterizations, and properties (1) and (2) follow from the lemmas proved there. In order to show property (3), we also have:

Lemma 19 $g(x_j) = f(x_j)$

Proof: Follows from the definition of g and the fact that P is not ϵ-bad. □

Theorem 20 *The program* **Polynomial-Self-Test** *is a $(\frac{\epsilon}{2(d+2)}, 4\epsilon)$-self-testing program for any degree d polynomial function over Z_p^m specified by its values at any (d, m)-polynomial test set $\mathcal{T}$, if $\epsilon \leq \frac{1}{4(d+2)^2}$.*

Proof: Follows from Lemmas 18,8, and 19. □

7 Locally Testable Codes

In this section we introduce some definitions related to coding and show the implications of low-degree testing to generating codes with nice properties. [3] We start by describing some standard parameters associated with error-correcting codes.

A n-letter string over the alphabet Σ is an element of Σ^n. Given a string $w \in \Sigma^n$, the ith character of w is denoted w_i. Given strings $w, w' \in \Sigma$, the relative distance between w and w', denoted $d(w, w')$ is the fraction of indices $i \in \{1, \ldots, n\}$ where $w_i \neq w'_i$. (Here onwards we will drop the term relative from the description of this parameter).

Definition 21 (Error Correcting Code) *A* (k, n, Δ, a)-code *consists of an alphabet* Σ *such that* $\log |\Sigma| = a$ *and a function* $C : \Sigma^k \to \Sigma^n$*, such that for any two strings* $m, m' \in \Sigma^k$*, the distance between* $C(m)$ *and* $C(m')$ *is at least* Δ*.*

For the purposes of this section we will restrict our attention to error-correcting codes within a small range of the above parameters which are interesting for the applications to probabilistically checkable proofs. We call these the *good* codes. Such codes need to have constant relative distance. The encoded message is allowed to be much larger than the original message size, as long as the final length is polynomially bounded. Perhaps the most interesting aspect is the alphabet size. While the ultimate goal would be to get codes which work over a constant sized alphabet, getting an alphabet size which is significantly smaller than the message size (smaller than any non-constant polynomial) turns out to be an important intermediate goal. Here we choose this parameter to be polylogarithmic in the message size.

Definition 22 (Good Code) *A family of codes* $\{C_i\}$ *with parameters* $(k_i, n_i, \Delta_i, a_i)$ *is* good *if* $k_i \to \infty$*,* n_i *is upper bounded by some polynomial in* k_i*,* $\Delta_i > 0$*, and* a_i *is upper bounded by some function growing as* $\mathrm{polylog}(k_i)$*.*

A wide variety of codes described in practice satisfy the properties required of a good code. In particular we describe the polynomial codes.

Definition 23 (Polynomial Codes) *Fix some* $\epsilon > 0$*. The* polynomial codes $\{P_m\}$ *are chosen by letting* $d = \lceil m^{1+\epsilon} \rceil$ *and picking a finite field* F *of size between* $10d$ *and* $20d$*. The code achieves* $k_m = \binom{m+d}{m}$ *and* $n_m = |F|^m$ *over the alphabet* F *and works as follows: The message is viewed as specifying the coefficients of a degree* d *polynomial in* m *variables and the encoding consists of the value of this polynomial at all inputs.*

It may be verified that $\{P_m\}$ forms a good code with distance at least 0.9. In what follows we will try to describe how this family of codes and a related code have extremely "good" local checkability properties. The following definition formalizes the notion of local checkability. Informally, the definition expects that by probing a string in just p (randomly chosen) letters, the verifier can test if it close to a valid codeword and if not rejects it with probability at least δ.

[3] These definitions are motivated by subsequent work in the area of proof checking where our tester has found applications, most notably that of [ALMSS92].

Definition 24 (Locally Testable Code) *For a positive integer p and a positive real number δ, an (n, k, Δ, a)-code C over the alphabet Σ is* (p, δ)-locally testable *if the following exist*

- *A probability space Ω which can be* efficiently *sampled.*
- *Functions $q_1, q_2, \ldots, q_p : \Omega \to \{1, \ldots, n\}$.*
- *A boolean function $V : \Omega \times \Sigma^p \to \{0, 1\}$.*

with the property that for all $w \in \Sigma^n$, if

$$\Pr_{r \in \Omega}\left[V(r, w_{q_1(r)}, \ldots, w_{q_p(r)}) = 0\right] < \delta$$

then there exists a (unique) string $m \in \Sigma^k$ such that $d(w, C(m)) < \Delta/2$. Conversely, if $w = C(m)$ for some m, then $V(r, w_{q_1(r)}, \ldots, w_{q_p(r)}) = 1$ for all $r \in \Omega$.

Before we describe the kind of locally checkable codes that our testers provide we attempt to motivate the definition above by showing that (seemingly minor) modifications of the above definitions yield important concepts in proof checking - namely, probabilistically checkable proofs. We consider especially probabilistically checkable proofs over a large alphabet in which number of alphabets that a verifier is allowed to probe is a parameter. This concept is an important ingredient in the recursive construction of probabilistically checkable proofs [AS92, ALMSS92, BGLR93] and is also of independent interest in complexity theory [LS91, FL92a]. The original definition of probabilistically checkable proofs is due to [AS92] based on an implicit notion in [FGLSS91]. A very closely related notion - that of holographic proofs - appears in the work of [BFLS91]. The particular choice of parameters made in the following definition is due to [BGLR93].

Definition 25 (PCP) *Given functions $r, p, a, \delta : \mathcal{Z}^+ \to \mathcal{Z}^+$, a language $L \subset \{0, 1\}^*$ is said to be in PCP$[r, p, a, \delta]$ if there exists a polynomially growing function $n(l)$, an alphabet Σ of size $a(l)$ such that for all integers $l > 0$ the following exist:*

- *A probability space Ω which can be sampled using $r(l)$ bits.*
- *Functions $q_1, q_2, \ldots, q_{p(k)} : \Omega \to \{1, \ldots, n(l)\}$.*
- *A boolean function $V : \{0, 1\}^l \times \Omega \times \Sigma^p \to \{0, 1\}$.*

with the property that for all $x \in \{0, 1\}^l$, if $w \in \Sigma^{n(l)}$ satisfies

$$\Pr_{r \in \Omega}\left[V(x, r, w_{q_1(r)}, \ldots, w_{q_p(r)}) = 0\right] < \delta$$

then $x \in L$. Conversely, if $x \in L$, then there exists $w \in \Sigma^n(l)$ such that for all $r \in \Omega$, $V(x, r, w_{q_1(r)}, \ldots, w_{q_p(r)}) = 1$.

It turns out that there is strong correlation between PCP[log, p, polylog, δ], and good codes which are (p, δ) locally checkable. In particular the codes we describe next translate into such probabilistically checkable proofs.

The robust characterization of polynomials described in Theorem 9 shows that the polynomial codes are $(d+2, \Omega(1/d))$-locally testable. Observe further that for the polynomial codes the growth of d is polylogarithmic in k. It seems that the approach above cannot hope to give codes which are testable using fewer than $\Omega(d)$ probes. However this is not the case. We describe next a simple way of modifying the codes so as to give codes with appreciably better local-testability. These codes are obtained by observing that the codes we have constructed so far use a much smaller alphabet size than necessary for "goodness".

Definition 26 (Polynomial-Line Codes) *Fix some $\epsilon > 0$. The* polynomial-line codes *$\{L_m\}$ are chosen by letting $t = \lceil m^{1+\epsilon} \rceil$ and picking a finite field F of size between $10d$ and $20d$. The code achieves $k_m = \binom{m+d}{m}/(d+1)$ and $n_m = |F|^{2m}$ over the alphabet F^{d+1}. As in the polynomial codes, the message again consists of $\binom{m+d}{d}$ field elements and is viewed as a degree d polynomial specified by its coefficients. Given a message polynomial p, the codeword is constructed as follows: For every pair of field elements $\hat{x}, \hat{h} \in F^m$, let $l_{\hat{x},\hat{h}}$ be the line through $\hat{x}$ with offset $\hat{h}$ as in Characterization 3. p restricted to $l_{\hat{x},\hat{h}}$ is a univariate polynomial of degree d. Let $C_{\hat{x},\hat{h}} \in F^{d+1}$ be the vector of coefficients of this univariate polynomial. The codeword consists of $\{C_{\hat{x},\hat{h}}\}_{\hat{x},\hat{h} \in F^m}$.*

It is easy to see that the Polynomial-Line Codes are also good codes. The proof of Theorem 9 can be transformed to show that the Polynomial-Line Codes are locally testable with a constant number of probes. More specifically the following can be shown.

Proposition: *The Polynomial-Line Codes are $(2, \Omega(1/d))$-locally testable.*

Better analysis of some portions of our proof yields even better statements about the Polynomial-Line Codes. This is described in the next section.

8 Conclusions

There has been a spate of results about low-degree tests in the last few years. A brief listing includes the low-degree test of [BFL91, Lun92] which was the first test for multivariate polynomials, the results of [BFLS91, FGLSS91] obtained independently and concurrently with ours (from [GLRSW91, RS92]), and subsequent works of [AS92, ALMSS92, FHS94, PS94, FS94]. Here we summarize some of their achievements along with a comparison with our results. We start by distinguishing the merits of our tester from those of [BFL91, BFLS91].

Program checking The test of [BFL91][Lun92], in the program checking setting allows the self-tester to be convinced that the program is computing a multivariate polynomial

function of low degree in polynomial time. However, the tests are somewhat complicated to perform, because they involve the reconstruction of a univariate polynomial given its values at a number of points (which in turn requires multiplications and matrix inversions), and later the evaluation of the reconstructed polynomial at random points. If the given function is a function of a single variable then the [BFL91][Lun92] tester is no simpler than a program evaluating the polynomial. Therefore it does not have the "little-oh" property defined by [BK89] nor is it *different* from the program evaluating the polynomial, in the sense defined by [BLR90], and does not give a self-tester or checker. Our test in contrast is *different* since it requires no multiplications to perform the test.

Relationship with proof checking. The low-degree tester forms a crucial ingredient in the recent results on proof checking. Our result from Section 4 gives a very simple proof of one of the relatively hard parts of the proof of MIP=NEXPTIME shown by [BFL91]. The hardness of the analysis of the tester of [BFL91] (and its simplifications, see for instance, [FGLSS91]) is in their need to rely on the isoperimetric properties of the m-dimensional grid. Our proof on the other hand does not seem to require any combinatorics, and is instead based on elementary algebraic/probabilistic techniques. This difference may be explained as follows: The success of the test does indeed depend on the isoperimetric properties of a graph related to the neighborhood structure. In the case of the test of [BFL91] this graph turns out to be in the m-dimensional grid. In our case, the underlying graph turns out to be a complete graph. This graph is obviously much easier to analyse for its properties and hence the proof is devoid of any combinatorial statements.

We now describe some of the subsequent results and the role of our tester in these results. The contrast is described in terms of locally-testable codes.

Locally testable codes The low-degree test described in [BFL91, BFLS91] gives rise to good codes which also have nice local checkability property. A sequence of improvements [BFL91, BFLS91, FGLSS91] culminated in the work of [AS92] which achieves asymptotically optimal bound for such codes by showing that they are $(2, \Omega(1/m))$-locally testable. The highlight of the work of [AS92] is that the locality bounds are independent of the degree of the polynomial that they work with. However, the dependence of δ on m, is inherent for such codes and $\delta \to 0$ as $m \to \infty$. The Polynomial-Line Codes described in Section 7 seem to have no inherent reason why δ should go to zero. This turns out to be indeed the case. [ALMSS92] observe that a combination of the analysis of [AS92] and that of Section 5 implies that there exists a constant $\delta > 0$ such that the Polynomial-Line Codes are $(2, \delta)$-locally testable, provided that the field F is of cardinaltity at least d^2. As mentioned in Section 7 this translates into a proof of NP $\subset$ PCP$[\log, O(1), \text{polylog}, \Omega(1)]$ in [ALMSS92]. By employing the technique of recursive proof checking, due to [AS92], on such proof systems [ALMSS92] go on to prove that NP $\subset$ PCP$[\log, O(1), O(1), \Omega(1)]$. The local testability of the Polynomial-Line codes has been further improved in two ways recently. [PS94] have shown that the codes are $(2, \delta)$-locally checkable over this works for linear sized fields as well, for some $\delta > 0$. In a different direction [FS94] show that the Polynomial-Line codes are $(2, \delta)$-locally checkable for all $\delta < 1/8$.

9 Acknowledgments

We are very grateful to Avi Wigderson for suggesting that our tester in Section 4 can be made more efficient, as well as his technical help in proving the theorems of Section 5. We are also very grateful to Sasha Shen for pointing out that the tester given in [GLRSW91] works for multivariate polynomials. In particular, Characterization 3 and its relevance to our test are due to him. We are grateful to Dick Lipton for illuminating conversations on the use of the testers presented here, and to Mike Luby, Shafi Goldwasser and Umesh Vazirani for technical suggestions. We would also like to thank Dieter van Melkebeek and the anonymous referees for pointing out numerous errors in earlier versions.

References

[AHK] L. Adleman, M. Huang, and K. Kompella. Efficient checkers for number-theoretic computations. Submitted to *Information and Computation.*

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the intractability of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd Annual IEEE Symposium of the Foundations of Computer Science*, pages 2–13, 1992.

[Bab93] L. Babai. Transparent (holographic) proofs. *Springer-Verlag Lecture Notes on Computer Science, 10th Annual Symposium on Theoretical Aspects of Computer Science*, 665:525–533, 1993.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, Springer Verlag LNCS 415*, pages 37–48, 1990.

[BF91] L. Babai and L. Fortnow. Arithmetization: A new method in structural complexity theory. *Computational Complexity*, 1:41–66, 1991.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 294–304, 1993.

[BK89] M. Blum and S. Kannan. Program correctness checking ... and the design of programs that check their work. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 86–97, 1989.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, 1990.

[Blu88] M. Blum. Designing programs to check their work. Technical Report TR–88–009, International Computer Science Institute, 1988.

[dW70] Van der Waerden. *Algebra*, volume 1. Frederick Ungar Publishing Co., Inc., 1970.

[FGLSS91] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 2–12, 1991.

[FHS94] K. Friedl, Z. Hatsagi, and A. Shen. Low-degree tests. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 57–64, 1994.

[FL92a] U. Feige and L. Lovasz. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 733–744, 1992.

[FS94] K. Friedl and M. Sudan. Improvements to total degree tests. Manuscript, 1993.

[GLRSW91] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 32–42, 1991.

[Kan90] S. Kannan. *Program Result Checking with Applications.* PhD thesis, University of California, Berkeley, 1990.

[Lip91] R. Lipton. New directions in testing. *Distributed Computing and Cryptography, DIMACS Series in Discrete Math and Theoretical Computer Science, American Mathematical Society*, 2:191–202, 1991.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXPTIME. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 13–18, 1991.

[Lun92] C. Lund. *The Power of Interaction.* ACM Distinguished Dissertations. The MIT Press, 1992.

[Nao92] M. Naor, April 1992. Personal Communication.

[PS94] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 194–203.

[RS92] R. Rubinfeld and M. Sudan. Testing polynomial functions efficiently and over rational domains. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 23–43, 1992.

[Rub90] R. Rubinfeld. *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs.* PhD thesis, University of California at Berkeley, 1990.

[She91] A. Shen. Personal Communication, May 1991.

[Sud92] M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* PhD thesis, University of California at Berkeley, 1992.

A Appendix

A.1 Evenly spaced points

The following algorithm may be used to test if a function $f^{(0)}$ on m evenly spaced points – $x, x+h, \ldots, x+(m-1)h$ – (where $m > d+1$) agrees with a degree d polynomial.

```
for i = 1 to d+1 do
   for j = 1 to m-i
      f^(i)(x+jh) = f^(i-1)(x+(j+1)h) - f^(i-1)(x+jh)
   endfor
endfor
verify f^(d+1)(x+jh) = 0, for all j ∈ {0,...,m-d-2}.
```

The correctness of this algorithm follows from the following well-known fact:

Fact 27 *$f^{(i)}(x)$ is a degree $d-i$ polynomial if and only if $f^{(i-1)}$ is a degree $d-i+1$ polynomial.*

(Follows from the fact that $f^{(i)}$ acts as the discrete derivative of $f^{(i-1)}$.)

This implies that $f^{(d)}$ is a constant if and only if $f^{(0)}$ is a degree d polynomial, implying in turn that $f^{(d+1)}$ is identically zero if and only if $f^{(0)}$ is a degree d polynomial. Observe further that the algorithm performs $O(md)$ additions and subtractions and no multiplications. Lastly it can also be checked that in case $m = d+2$, then algorithm simply verifies that $\sum_{i=0}^{d+1} \alpha_i f^{(0)}(x+ih) = 0$, where $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$.

A.2 Characterizations

Lemma 28 (axis parallel lines) *$f : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ is a polynomial in m variables of degree at most d in each variable if and only if for all $i \in \{1, \ldots, m\}$, $\beta_j \in \mathcal{Z}_p$ $(j \neq i)$, $f(\beta_1, \ldots, \beta_{i-1}, x_i, \ldots, \beta_m)$ is a polynomial in x_i of degree at most d.*

Proof [Sketch]: It is clear that every polynomial of degree d in each variable restricted to axis parallel lines, behaves as a univariate polynomial of degree d. The other direction can be proved by induction on m. The base case $m = 1$ is obvious. For general $m > 1$, let $f_i(x_1, \ldots, x_{m-1})$ be the function $f(x_1, \ldots, x_{m-1}, i)$. By induction f_i is a polynomial of degree d in $m-1$ variables. Now consider the function $h(x_1, \ldots, x_m) \equiv \sum_{i=0}^{d} \delta_i^{(d)}(x_m) f_i(x_1, \ldots, x_{m-1})$ (where $\delta_i^{(d)}$ is the unique polynomial of degree d in one variable that is 1 at $x_m = i$ and 0 for other values of $x_m \in \{0, \ldots, d\}$).

It is clear by construction that h is a polynomial of degree at most d in each variable. We now argue that f and h are identical. Fix $x_1 = \beta_1, \ldots, x_{m-1} = \beta_{m-1}$. It is clear that $h(x_1, \ldots, x_m) = f(x_1, \ldots, x_m)$ for $x_m \in \{0, \ldots, d\}$. Moreover, both h and f are degree d polynomials in x_m which agree at $d + 1$ places. Hence f and h must agree at all values of x_m. Since this held for any choice of β_i's, f and h agree everywhere. □

Lemma 29 (general lines) *For $p \geq 2d + 1$, $f : \mathcal{Z}_p^m \mapsto \mathcal{Z}_p$ is a polynomial in m variables of total degree at most d if and only if $\forall \hat{x}, \hat{h} \in \mathcal{Z}_p^m$, $f(\hat{x} + t \cdot \hat{h})$ is a univariate polynomial in t of degree at most d.*

Proof: It is clear that every polynomial restricted to lines must become a degree d polynomial in the parameter describing the line. Here we prove the other direction of the characterization. We first observe that since the set of all lines includes the axis parallel lines, we can use Lemma 28 to show that f is a polynomial in m variables with degree at most d in each variable. Having got this weak characterization, we will now strengthen this to a tighter one. By induction on the number of variables, we can assume that f restricted to any value of the last variable x_m is a polynomial of total degree at most d in the variables $x_1, \ldots, x_{m-1}$. Thus f becomes a function in x_1 through x_m of total degree $d' \leq 2d$.

Assume for contradiction that $d' > d$. Now consider the function $f(t \cdot \hat{h})$ for $\hat{h} \in_R \mathcal{Z}_p^m$. The coefficient of $t^{d'}$ is a polynomial in $\hat{h}$ of degree d' which with probability at least $1 - \frac{d'}{p}$ should be non-zero. (Note that to make this probability positive, we need $2d < p$.) Thus f restricted to this line is a polynomial of degree $d' > d$, which violates the given condition on f. □