Checking Approximate Computations of Polynomials and Functional Equations [*]

Funda Ergün [†] S Ravi Kumar[‡] Ronitt Rubinfeld[§]

June 10, 2004

Abstract

A majority of the results on self-testing and correcting deal with programs which purport to compute the correct results precisely. We relax this notion of correctness and show how to check programs that compute only a numerical approximation to the correct answer. The types of programs that we deal with are those computing polynomials and functions defined by certain types of functional equations. We present results showing how to perform approximate checking, self-testing, and self-correcting of polynomials, settling in the affirmative a question raised by [20, 29, 30]. We obtain this by first building approximate self-testers for linear and multilinear functions. We then show how to perform approximate checking, self-testing, and self-correcting for those functions that satisfy addition theorems, settling a question raised by [28]. In

[*]This work was partially supported by NSF Career grant CCR-9624552, the Alfred P. Sloan Research Award, and ONR grant N00014-97-1-0505. The first and second authors were also supported by NSF grant DMI-91157199. The third author was also supported by the grant No. 92-00226 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel. Parts of this research were conducted while the authors were at the Computer Science Department of Cornell University, and visiting the M.I.T. Lab. for Computer Science. A peliminary version of this paper appeared in the 37th IEEE Conference on Foundations of Computer Science, 1996.

[†]Case Western Reserve University, Cleveland, OH 44106.

[‡]IBM Almaden Research Center, San Jose, CA 95120.

[§]NEC Research Institute, Princeton, NJ 08540.

both cases, we show that the properties used to test programs for these functions are both robust (in the approximate sense) and stable. Finally, we explore the use of reductions between functional equations in the context of approximate self-testing. Our results have implications for the stability theory of functional equations.

1 Introduction

Program checking was introduced by Blum and Kannan [7] in order to allow one to use a program safely, without having to know apriori that the program is correct on all inputs. Related notions of self-testing and self-correcting were further explored in [8, 24]. These notions are seen to be powerful from a practical point of view (c.f., [9]) and from a theoretical angle (c.f., [5, 4]) as well. The techniques used usually consist of tests performed at run-time which compare the output of the program either to a predetermined value or to a function of outputs of the same program at different inputs. In order to apply these powerful techniques to programs computing real-valued functions, several issues dealing with *precision* need to be dealt with. The standard model, which considers an output to be wrong even if it is off by a very small margin, is too strong to make practical sense due to reasons such as the following: (i) In many cases, the algorithm is only intended to compute an approximation, e.g., Newton's method. (ii) Representational limitations and round-off/truncation errors are inevitable in real-valued computations. (iii) The representation of some fundamental constants (e.g., $\pi = 3.14159\ldots$) is inherently imprecise.

The framework presented by [20, 3] accommodates these inherently inevitable or acceptably small losses of information by overlooking small precision errors while detecting actual "bugs", which manifest themselves with greater magnitude. Given a function f, a program P that purports to compute f, and an error bound Δ, if $|P(x) - f(x)| \leq \Delta$ (denoted $P(x) \approx_\Delta f(x)$) under some appropriate notion of norm, we say $P(x)$ is *approximately correct on input x*. *Approximate result checkers* test if P is approximately correct for a given input x. *Approximate self-testers* are programs that test if P is approximately correct for most inputs. *Approximate self-correctors* take programs that are approximately correct on most inputs and turn them into programs that are approximately correct on every input.

Domains. We work with finite subsets of fixed point arithmetic that we refer to as *finite rational domains*. For $n, s \in \mathbb{Z}^+$, $\mathcal{D}_{n,s} \overset{\text{def}}{=} \{\frac{i}{s} : |i| \leq n, i \in \mathbb{Z}\}$. Usually, $s = 2^l$ where l is the precision. We allow s and n to vary for generality. For a domain $\mathcal{D}$, let $\mathcal{D}^+$ and $\mathcal{D}^-$ denote the positive and negative elements in $\mathcal{D}$.

Testing using Properties. There are many approaches to building self-testers. We illustrate one paradigm that has been particularly useful. In this approach, in order to test if a program P computes a function f on most inputs, we test if P satisfies certain properties of f.

As an example, consider the function $f(x) = 2x$ and the property "$f(x+1) = f(x) + 2$" that f satisfies. One might pick random inputs x and verify that $P(x+1) = P(x) + 2$. Clearly, if for some x, $P(x+1) \neq P(x) + 2$, then P is incorrect. The program, however, might be quite incorrect and still satisfy $P(x+1) = P(x) + 2$ for most choices of random inputs. In particular, there exists a P (for instance, $P(x) = 2x \bmod K$)[1] such that: (i) with high probability, P satisfies the property at random x and hence will pass the test, and (ii) there is no function that satisfies the property for all x such that P agrees with this function on most inputs. Thus we see that this method, when used naively, does not yield a self-tester that works according to our specifications. Nevertheless, this approach has been used as a good heuristic to check the correctness of programs [13, 14, 35].

As an example of a property that does yield a good tester, consider the linearity property "$f(x+y) = f(x) + f(y)$", satisfied only by functions mapping $\mathcal{D}_{n,s}$ to $\mathbb{R}$ of the form $f(x) = cx, c \in \mathbb{R}$. If, by random sampling, we conclude that the program P satisfies this property for most x, y, it can be shown that P agrees with a linear function g on most inputs [8, 28]. We call the linearity property, and any property that exhibits such behavior, a *robust property*.

We now describe more formally how to build a self-tester for a *class* $\mathcal{F}$ of functions that can be characterized by a robust property. The two-step approach, which was introduced in [8], is: (i) test that P satisfies the robust property (*property testing*), and (ii) check if P agrees with a *specific* member of $\mathcal{F}$ (*equality testing*). The success of this approach depends on finding robust properties which are both easy to test and lead to efficient equality tests.

A *property* is a pair $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)} \rangle$, consisting of an equation $\mathcal{I}^f(x_1, \ldots, x_k) = 0$ that relates the values of function f at various tuples of locations $\langle x_1, \ldots, x_k \rangle$, and a distribution $\mathcal{E}_{\tau(n,s)}$ over $\mathcal{D}^k_{\tau(n,s)}$ from which the locations are picked. The property $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)} \rangle$ is said to characterize a function family $\mathcal{F}$ in the following way. A function f is a member of $\mathcal{F}$ if and only if $\mathcal{I}^f(x_1, \ldots, x_k) = 0$ for every $\langle x_1, \ldots, x_k \rangle$ that has non-zero support under $\mathcal{E}_{\tau(n,s)}$. For

[1]We naturally extend the mod function to $\mathcal{D}_{n,s}$ by letting $x \bmod K$ stand for $\frac{j \bmod k}{s}$, for $x, K \in \mathcal{D}_{n,s}$, and $x = \frac{j}{s}$, $K = \frac{k}{s}$.

instance, the linearity property can be written as $\mathcal{I}^f(x_1, x_2, x_3) \equiv f(x_1) + f(x_2) - f(x_3) = 0$, and $\mathcal{E}^{\mathrm{Lin}}_{\tau(n,s)}$ is a distribution on $\langle x_1, x_2, x_1 + x_2\rangle$, where x_1 and x_2 are chosen randomly from some distribution[2] over the domain $\mathcal{D}_{\tau(n,s)}$. In this case $\langle \mathcal{I}, \mathcal{E}^{\mathrm{Lin}}_{\tau(n,s)}\rangle$ characterizes $\mathcal{F} = \{f(x) = cx \mid c \in \mathbb{R}\}$, the set of all linear functions over $\mathcal{D}_{\tau(n,s)}$. We will adhere to this definition of a property throughout the paper; however, for simplicity of notation, when appropriate, we will talk about the distribution and the equality together. For instance, we express the linearity property as $f(x + y) = f(x) + f(y)$, giving the distributions of x, y.

We first consider robust properties in more detail. Suppose we want to infer the correctness of the program on inputs from the domain $\mathcal{D}_{n,s}$. Then we allow calls to the program on a larger domain $\mathcal{D}_{\tau(n,s)}$, where $\tau : \mathbb{Z}^2 \to \mathbb{Z}^2$ is a fixed function that depends on the structure of $\mathcal{I}$. Ideally, we would like $\tau(n, s) = (n, s)$, i.e., $\mathcal{D}_{\tau(n,s)} = \mathcal{D}_{n,s}$. But, for technical reasons, we allow $\mathcal{D}_{\tau(n,s)}$ to be a proper, but not too much larger, superset of $\mathcal{D}_{n,s}$ (in particular, the description size of an element in $\mathcal{D}_{\tau(n,s)}$ should be polynomial in the description size of an element in $\mathcal{D}_{n,s}$).[3]

To use a property in a self-tester, one must prove that the property is robust. Informally, the $(\delta, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-*robustness* of the property $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)}\rangle$ implies that if, for a program P, $\mathcal{I}^P(x_1, \ldots, x_k) = 0$ is satisfied with probability at least $1-\epsilon$ when $\langle x_1, \ldots, x_k\rangle$ is chosen from the distribution $\mathcal{E}_{\tau(n,s)}$, then there is a function $g \in \mathcal{F}$ that agrees with P on $1 - \delta$ fraction of the inputs in $\mathcal{D}_{n,s}$. In the case of linearity, it can be shown that there is a distribution $\mathcal{E}^{\mathrm{Lin}}_{11n,s}$ on $\langle x_1, x_2, x_1 + x_2\rangle$ where $x_1, x_2 \in \mathcal{D}_{11n,s}$ such that the property is $(2\epsilon, \epsilon, \mathcal{D}_{11n,s}, \mathcal{D}_{n,s})$-robust for all $\epsilon < 1/48$ [8, 28]. Therefore, once it is tested that P satisfies $P(x_1)+P(x_2) = P(x_1+x_2)$ with large enough probability when the inputs are picked randomly from $\mathcal{E}^{\mathrm{Lin}}_{11n,s}$, it is possible to conclude that P agrees with some linear function on most inputs from $\mathcal{D}_{n,s}$. A somewhat involved definition of *robust* is given in [28]. Given a function τ such that for all n, s, $\mathcal{D}_{n,s}$ is a large enough subset of $\mathcal{D}_{\tau(n,s)}$, in this paper we say that a property is *robust* if: for all $0 < \delta < 1$, there is an ϵ such that for all n, s the property is $(\delta, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-robust.

[2]For example, choosing x_1 and x_2 uniformly from $\mathcal{D}_{\tau(n,s)}$ suffices for characterizing linearity. To prove robustness, however, [28] uses a more complicated distribution that we do not describe here.

[3]Alternatively, one could test the program over the domain $\mathcal{D}_{n,s}$ and attempt to infer the correctness of the program on most inputs from $\mathcal{D}_{n',s'}$, where $\mathcal{D}_{n',s'}$ is a large subdomain of $\mathcal{D}_{n,s}$.

We now consider equality testing. Recall that once it is determined that P satisfies the robust property, then equality testing determines that P agrees on most inputs with a *specific* member of $\mathcal{F}$. For instance, in the case of linearity, to ensure that P computes the specific linear function $f(x) = x$ on most inputs, we perform the equality test which ensures that $P(x + \frac{1}{s}) = P(x) + \frac{1}{s}$ for most x. Neither the property test nor the equality test on its own is sufficient for testing the program. However, since $f(x) = x$ is the *only* function that satisfies both the linearity property and the above equality property, the combination of the property test and the equality test can be shown to be sufficient for constructing self-testers.

This combined approach yields extremely efficient testers (that only make $O(\epsilon^{-1} \log 1/\delta)$ calls to the program for fixed δ and ϵ) for programs computing homomorphisms (e.g., multiplication of integers and matrices, exponentiation, logarithm). This idea is further generalized in [28], where the class of functional equations called *addition theorems* is shown to be useful for self-testing. An addition theorem is a mathematical identity of the form $\forall x, y, f(x + y) = G[f(x), f(y)]$. Addition theorems characterize many useful and interesting mathematical functions [1, 11]. When G is algebraic, they can be used to characterize families of functions that are rational functions of x, e^{cx}, and doubly periodic functions (see Table 1 for examples of functional equations and the families of functions that they characterize over the reals). Polynomials of degree d can be characterized via several different robust functional equations (e.g., [6, 26, 4, 30]).

Approximate Robustness and Stability. When the program works with finite precision, the properties upon which the testers are built will rarely be satisfied, even by a program whose answers are correct up to the required (or hardware-wise maximal) number of digits, since they involve strict equalities. Thus, when testing, one might be willing to pass programs for which the properties are only approximately satisfied. This relaxation in the tests, however, leads to some difficulties, for in the approximate setting: (i) it is harder to analyze which function families are solutions to the robust properties, and (ii) equality testing is more difficult. For instance, it is not obvious which family of functions would satisfy both $P(x_1) + P(x_2) \approx P(x_1 + x_2)$, for all $x, y \in \mathcal{D}_{\tau(n,s)}$, (approximate linearity property) and $P(x + \frac{1}{s}) \approx P(x) + \frac{1}{s}$ for all $x \in \mathcal{D}_{\tau(n,s)}$. (approximate equality property).

$G[f(x), f(y)]$	$f(x)$	$G[f(x), f(y)]$	$f(x)$
$f(x) + f(y)$	Ax	$f(x)f(y) - \sqrt{1 - f(x)^2}\sqrt{1 - f(y)^2}$	$\cos Ax$
$\frac{f(x)+f(y)}{1-f(x)f(y)}$	$\tan Ax$	$\frac{f(x)+f(y)-2f(x)f(y)}{1-2f(x)f(y)}$	$\frac{1}{1+\cot Ax}$
$\frac{f(x)f(y)-1}{f(x)+f(y)}$	$\cot Ax$	$\frac{f(x)+f(y)-2f(x)f(y)\cos a}{1-f(x)f(y)}$	$\frac{\sin Ax}{\sin Ax+a}$
$\frac{f(x)+f(y)-1}{2f(x)+2f(y)-2f(x)f(y)-1}$	$\frac{1}{1+\tan Ax}$	$\frac{f(x)+f(y)-2f(x)f(y)\cosh a}{1-f(x)f(y)}$	$\frac{\sinh Ax}{\sinh Ax+a}$
$\frac{f(x)+f(y)-2f(x)f(y)}{1-f(x)f(y)}$	$\frac{-Ax}{1-Ax}$	$\frac{f(x)+f(y)+2f(x)f(y)\cosh a}{1-f(x)f(y)}$	$\frac{-\sinh Ax}{\sinh Ax+a}$
$\frac{f(x)+f(y)}{1+[f(x)f(y)]/A^2}$	$A \tanh Bx$	$\frac{f(x)+f(y)+2f(x)f(y)}{1-f(x)f(y)}$	$\frac{Ax}{1-Ax}$
$\frac{f(x)f(y)}{f(x)+f(y)}$	$\frac{A}{x}$	$f(x)f(y) + \sqrt{f(x)^2 - 1}\sqrt{f(y)^2 - 1}$	$\cosh Ax$

Table 1: Some Addition Theorems of the form $f(x + y) = G[f(x), f(y)]$.

To construct approximate self-testers, our approach is to first investigate a notion of *approximate robustness* of the property to be used. We first require a notion of distance between two functions.

Definition 1 (Chebyshev Norm) *For a function f on a domain $\mathcal{D}$,* $\|f\|_{\mathcal{D}} = \|f\| = \sup_{x\in\mathcal{D}}\{|f(x)|\}$.

When the domain is obvious from the context, we drop it. Given functions f, g, the *distance* between them is $\|f - g\|$. Next, we define the *approximation* of a function by another:

Definition 2 *The function P* (Δ, ϵ)-approximates *f on domain $\mathcal{D}$ if $\|P - f\| \leq \Delta$ on at least $1 - \epsilon$ fraction of $\mathcal{D}$.*

Approximate robustness is a natural extension of the robustness of a property. We say that a *program satisfies a property approximately* if the property is true of the program when exact equalities are replaced by approximate equalities. Once again consider the linearity property and a program P that satisfies the property approximately (i.e., $P(x_1+x_2) \approx_{\Delta} P(x_1)+P(x_2)$)

for all but an ϵ fraction of the choices of $\langle x_1, x_2, x_1+x_2\rangle \in \mathcal{E}^{\text{Lin}}_{\tau(n,s)}$. The approximate robustness of linearity implies that there exists a function g and a choice of Δ', Δ'' such that $g(x+y) \approx_{\Delta'} g(x) + g(y)$ for all inputs $x, y \in \mathcal{D}_{n,s}$, and g $(\Delta'', 2\epsilon)$-approximates P on $\mathcal{D}_{n,s}$ [20, 28]. In general, we would like to define approximate robustness of a property $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)}\rangle$ as the following: If a program P satisfies the equation $\mathcal{I}$ approximately on most choices of inputs according to the distribution $\mathcal{E}_{\tau(n,s)}$, then there exists a function g that: (i) satisfies $\mathcal{I}$ approximately on all inputs chosen according to $\mathcal{E}_{n,s}$ (ii) approximates P on most inputs in $\mathcal{D}_{n,s}$, the support of $\mathcal{E}_{\tau(n,s)}$. The function τ relates the distributions used for describing the behaviors of P and G and depends on $\mathcal{I}$.

We now give a formal definition of approximate robustness:

Definition 3 (Approximate Robustness) *Let $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)}\rangle$ characterize the family of functions $\mathcal{F}$ over the domain $\mathcal{D}_{\tau(n,s)}$. Let $\mathcal{F}'$ be the family of functions satisfying $\mathcal{I}$ approximately on all inputs chosen according to $\mathcal{E}_{n,s}$. Let ϵ, δ be absolute constants independent of n. A property $\langle \mathcal{I}, \mathcal{E}_{\tau(n,s)}\rangle$ for a function family $\mathcal{F}'$ is* $(\delta, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s}, \Delta, \Delta', \Delta'')$-approximately robust *if $\forall P, \Pr_{\langle x_1,\ldots,x_k\rangle \sim \mathcal{E}_{\tau(n,s)}}[\mathcal{I}^P(x_1,\ldots,x_k) \approx_\Delta 0] \geq 1-\epsilon$ implies there is a $g \in \mathcal{F}'$ that (Δ'', δ)-approximates P on $\mathcal{D}_{n,s}$ and $\mathcal{I}^g(x_1,\ldots,x_k) \approx_{\Delta'} 0$ for all tuples $\langle x_1,\ldots,x_k\rangle$ with non-zero support in $\mathcal{E}_{n,s}$.*

Once we know that the property is approximately robust, the second step is to analyze the *stability* of the property, i.e., to characterize the set of functions $\mathcal{F}'$ that satisfy the property approximately and compare it to $\mathcal{F}$, the set of functions that satisfy the property exactly (*Hyers-Ulam stability* [21]). In our linearity example, the problem is the following: given g satisfying $g(x+y) \approx_\Delta g(x) + g(y)$ for all x, y in the domain, is there a homomorphism h that $(\Delta', 0)$-approximates g with Δ' depending only on Δ and *not* on the size of the domain? If the answer is affirmative, we say that the property is *stable*. In the following definition, $\mathcal{D}_{n',s'} \subseteq \mathcal{D}_{n,s}$.

Definition 4 (Stability) *A property $\langle \mathcal{I}, \mathcal{E}_{n,s}\rangle$ for a function family $\mathcal{F}$ is* $(\mathcal{D}_{n,s}, \mathcal{D}_{n',s'}, \Delta, \Delta')$-stable *if $\forall g$ that satisfies $\mathcal{I}^g \approx_\Delta 0$ for all tuples with non-zero support according to $\mathcal{E}_{n,s}$, there is a function h that satisfies $\mathcal{I}^h = 0$ for all tuples with non-zero support according to $\mathcal{E}_{n',s'}$ with $\|h - g\|_{\mathcal{D}_{n',s'}} \leq \Delta'$.*

If a property is both approximately robust and stable, then it can be used to determine whether P approximates some function in the desired family. Furthermore, if we have a method of doing approximate equality testing, then we can construct an approximate self-tester. Here, we assume that the distributions associated with approximate robustness and stability are samplable.

Previous Work. Previously, not many of the known checkers have been extended to the approximate case. Often it is rather straightforward to extend the robustness results to show approximate robustness. However, the difficulty with extending the checkers appears to lie in showing the stability of the properties. The issue is first mentioned in [20], where approximate checkers for mod, exponentiation, and logarithm are constructed. The domain is assumed to be closed in all of these results. A domain is said to be closed under an operation if the range of the operation is a subset of the domain. For instance, a finite precision rational domain is not closed under addition. In [3] approximate checkers for sine, cosine, matrix multiplication, matrix inversion, linear system solving, and determinant are given. The domain is assumed to be closed in the results on sine and cosine. In [10] an approximate checker for floating-point division is given. In [32], a technique which uses approximation theory is presented to test univariate polynomials of degree at most 9. It is left open in [20, 3, 30, 28] whether the properties used to test polynomial, hyperbolic, and other trigonometric functions can be used in the approximate setting. For instance, showing the stability of such functional equations is not obvious; if the functional equation involves division with a large numerator and a small denominator, a small additive error in the denominator leads to a large additive error in the output.

There has been significant work on the stability of specific functional equations. The stability of linearity and other homomorphisms is addressed in [21, 16, 18, 12]. The techniques used to prove the above results, however, cease to apply when the domain is not closed. The stronger property of stability in a non-closed space, called *local stability*, is addressed by Skof [31] who proves that Cauchy functional equations are locally stable on a finite interval in $\mathbb{R}$. The problem of stability of univariate polynomials over continuous domains is first addressed in [2] and the problem of local stability on $\mathbb{R}$ is solved in [19]. See [17] for a survey. These results do not extend in an obvious way to finite subsets of $\mathbb{R}$, and thus cannot be used to show the correctness of self-testers. For those that can be extended, the error bounds obtained by naive extensions are not optimal. Our different approach allows us to operate on $\mathcal{D}_{n,s}$ *and* obtain tight bounds.

Results. In this paper, we answer the questions of [20, 3, 30, 28] in the affirmative, by giving the first approximate versions of most of their testers. We first present an approximate tester for linear and multilinear functions with tight bounds. These results apply to several functions, including multiplication, exponentiation, and logarithm, over non-closed domains. We next present the first approximate testers for multivariate polynomials. Finally, we show how to approximately test functions satisfying addition theorems. Our results apply to many algebraic functions of trigonometric and hyperbolic functions (e.g., sinh, cosh). All of our results apply to non-closed discrete domains.

Since a functional equation over $\mathbb{R}$ has more constraints than the same functional equation over $\mathcal{D}_{n,s}$, it may happen that the functional equation over $\mathbb{R}$ characterizes a family of functions that is a proper subset of the functions characterized by the same functional equation over $\mathcal{D}_{n,s}$. This does not limit the ability to construct self-testers for programs for these functions, due to the equality testing performed by self-testers.

To show our results, we prove new local stability results for discrete domains. Our techniques for showing the stability of multilinearity differ from those used previously in that (i) we do not require the domain to be discrete and (ii) we do not require the range to be a complete metric space. This allows us to apply our results to multivariate polynomial characterizations. In addition to new combinatorial arguments, we employ tools from approximation theory and stability theory. Our techniques appear to be more generally applicable and cleaner to work with than those previously used.

Self-correctors are built by taking advantage of the random self-reducibility of polynomials and functional equations [8, 24] in the exact case. As in [20], we employ a similar idea for the approximate case by making several guesses at the answer and returning their median as the output. We show that if each guess is within Δ of the correct answer with high probability, then the median yields a good answer with high probability. To build an approximate checker for all of these functions, we combine the approximate self-tester and approximate self-corrector as in [8].

Subsequent to our work, our results have been extended to the case of relative error in a recent paper of [22].

Organization. Section 2 addresses the stability of the properties used to test linear and multilinear functions. Using these results, Section 3 considers approximate self-testing of polynomials. Section 4 addresses the stability and robustness of functional equations. Section 5 illustrates the actual construction of approximate self-testers and self-correctors.

2 Linearity and Multilinearity

In this section, we consider the stability of the robust properties used to test linearity and multilinearity over the finite rational domain $\mathcal{D}_{n,s}$. The results in this section, in addition to being useful for the testing of linear and multilinear functions, are crucial to our results in Section 3.

As in [20], approximate robustness is easy to show by appropriately modifying the proof of robustness [28]. This involves replacing each exact equality by an approximate equality and keeping track of the error accrued at each step of the proof. To show stability, we use two types of bootstrapping arguments: the first shows that an error bound on a small subset of the domain implies the same error bound on a larger subset of the domain; the second shows that an error bound on the whole domain implies a tighter error bound over the same domain. These results can be applied to give the first approximate self-testers for several functions over $\mathcal{D}_{n,s}$ including multiplication, exponentiation, and logarithm (Section 2.2).

2.1 Approximate Linearity

The following defines formally what it means for a function to be approximately linear:

Definition 5 (Approximate Linearity) *A function g is* Δ-approximately linear *on $\mathcal{D}_{n,s}$ if $\forall x, y \in \mathcal{D}_{n,s}, g(x+y) \approx_{\Delta} g(x) + g(y)$.*

Hyers [21] and Skof [31] obtain a linear approximation to an approximately linear function when the domain is $\mathbb{R}$. (See Appendix A for their approach). Their methods are not extendible to discrete domains.

Suppose we define h such that $h(\frac{1}{s}) \stackrel{\text{def}}{=} g(\frac{1}{s})$ and h is linear. In the 0-approximately linear case (exact linearity), since $g(\frac{i}{s}) = g(\frac{i-1}{s}) + h(\frac{1}{s})$ and $h(\frac{i}{s}) = h(\frac{i-1}{s}) + h(\frac{1}{s})$, by induction on the elements in $\mathcal{D}_{n,s}$, we can show that $h(x) = g(x), \forall x$. This approach is typically used to prove the sufficiency of the equality test. However, in the Δ-approximately linear case for $\Delta \neq 0$, using the same inductive argument will only yield a linear function h such that $h(\frac{i}{s}) \approx_{i \cdot \Delta} g(\frac{i}{s})$. This is quite unattractive since the error bound depends on the domain size. The problem of obtaining a linear function h whose discrepancy from g is *independent* of the size of the domain is non-trivial.

In [20], a solution is given for when the domain is a finite group. Their technique requires that the domain be closed under addition, and therefore does not work for $\mathcal{D}_{n,s}$. We give a brief overview of the scheme in [20] and point out where it breaks down for non-closed domains. The existence of a linear h that is close to g is done in [20] by arguing that if $\mathcal{D}$ is sufficiently large, then an error of at least Δ at the maximum error point x^* would imply an even bigger error at $2x^*$, contradicting the maximality assumption about error at x^*. Here, the crucial assumption is that $x \in \mathcal{D}$ implies $2x \in \mathcal{D}$. This step fails for domains which are not closed under addition.

Instead, we employ a different constructive technique to obtain a linear h on $\mathcal{D}_{n,s}$ given a Δ-approximately linear g. Our technique yields a tight bound of 2Δ on the error $e \equiv h - g$ (instead of 4Δ in [31]) and does not require that the domain be closed under addition. It is important to achieve the best (lowest) constants possible on the error, because these results are used in Section 3.2 where the constants affect the error in an exponential way.

The following lemma shows how to construct a linear function h that is within $2\Delta + \rho$ of a Δ-approximately linear function g in $\mathcal{D}^+_{n,s}$.

Lemma 6 *Let g be a Δ-approximately linear function on $\mathcal{D}^+_{n,s}$, and let h be linear on $\mathcal{D}_{n,s}$. Define $e(x) = h(x) - g(x)$. If $|e(\frac{n}{s})| = \rho$, then $\forall x \in \mathcal{D}^+_{n,s}, |e(x)| \leq 2\Delta + \rho$.*

Proof. We prove by contradiction that $\forall x \in \mathcal{D}^+_{n,s}, e(x) \leq 2\Delta + \rho$. A symmetric argument can be made to show that $e(x) \geq -(2\Delta + \rho)$.

Recall that $\frac{n}{s}$ is the greatest positive element of the domain, and note that e is a Δ-approximately linear function. Assume that there exists a point in $\mathcal{D}^+_{n,s}$ with error greater than $2\Delta + \rho$. Let p be the maximal such element. p has to lie between $\frac{n}{2s}$ and $\frac{n}{s}$, otherwise $2p \in \mathcal{D}^+_{n,s}$ would have error greater than $2\Delta + \rho$, contradicting the maximality of p. Let $q = \frac{n}{s} - p$. Then, $e(q) + e(p) \approx_\Delta e(\frac{n}{s})$, therefore $e(q) < -\Delta$. Also, for any $x \in (p, \frac{n}{s}] \subseteq \mathcal{D}^+_{n,s}$, by definition of p, $e(x) \leq 2\Delta + \rho$. Note that any such x can be written as $x = x' + p$, where $x' \in (0, q]$. To satisfy the approximate linearity property that $e(x') + e(p) \approx_\Delta e(x)$, x' must have error strictly less than $\Delta + \rho$.

We now know that the points in the interval $(0, q]$ have error strictly less than $2\Delta + \rho$ (in fact, less than $\Delta + \rho$), and that the point q itself has error strictly less than $-\Delta$. Putting these two facts and approximate linearity together, and since any $x \in (q, 2q]$ can be written as $q + y$ where $y \in (0, q]$, we can conclude that at any point in $(q, 2q]$, the error is at most $2\Delta + \rho$. Now we can repeat the same argument by taking y from $(0, 2q]$ rather than $(0, q]$ to bound the error in the interval $(0, 3q]$ by $2\Delta + \rho$. By continuing this argument, eventually the interval contains the point p, which means that p has error at most $2\Delta + \rho$. This contradicts our initial assumption that $e(p)$ was greater than $2\Delta + \rho$. □

In addition, since $e(0) \approx_\Delta e(0) + e(0)$, $|e(0)| \leq \Delta$. We now generalize the error bound on $\mathcal{D}^+_{n,s}$ to $\mathcal{D}_{n,s}$.

Lemma 7 *If a function g is Δ-approximately linear on $\mathcal{D}_{n,s}$, with h and e defined as in Lemma 6, and if $|e(\frac{n}{s})| = \rho$, then $\forall x \in \mathcal{D}_{n,s}, |e(x)| \leq 2\Delta + \rho$.*

Proof. Observe that if the error $e(x)$ is upper bounded by σ when $x \in [0, \frac{n}{s}]$, then $|e(x)| \leq (\sigma + \Delta)/2$ whenever $0 \leq x \leq \frac{n}{2s}$, since $e(2x) \leq \sigma$. Also, if $|e(x)| \leq \mu$ then $|e(-x)| \leq \mu + 2\Delta$ since $e(0) \leq \Delta$. By Lemma 6, $e(x) \leq 2\Delta + \rho$ for all $x \in \mathcal{D}^+_{n,s}$. We will bound the error in $\mathcal{D}^-_{n,s}$ first by $3\Delta + \rho$ and then by $2\Delta + \rho$. From the above observations, we have $e(x) \leq 4\Delta + \rho$ for $x \in \mathcal{D}^-_{n,s}$, $e(x) \leq (3\Delta + \rho)/2$ for $x \in [0, \frac{n}{2s}]$ and $e(x) \leq (5\Delta + \rho)/2$ for $x \in [-\frac{n}{2s}, 0]$.

Assume that $\exists x \in \mathcal{D}^-_{n,s}$ such that $e(x) = 3\Delta + \rho + \epsilon > 3\Delta + \rho$. Let p be such a point with minimal absolute value. such point. Then $p < -\frac{n}{2s}$, otherwise the error at $2p$ would exceed

$3\Delta + \rho$. Let t be the point with the highest error in $\mathcal{D}^+_{n,s}$ (the maximal such one if there is a tie). We consider the possible locations for t to bound $e(t)$: (i) if $t \leq \frac{n}{2s}$, then to ensure that $e(2t) \leq e(t)$, $e(t) \leq \Delta$; (ii) if $\frac{n}{2s} < t \leq |p|$, then $t + p \in [-\frac{n}{2s}, 0]$, therefore, to satisfy the bound above on $e(t+p)$, $e(t) \leq \Delta/2 - \epsilon \leq \Delta$; (iii) if $t > |p|$, then $t + p \in (0, \frac{n}{2s}]$, therefore to satisfy the bound above, $e(t) \leq -\Delta/2 - \epsilon \leq \Delta$.

Regardless of where t lies, $e(t) \leq \Delta \leq \Delta + \rho$, hence the error in $\mathcal{D}^+_{n,s}$ is bounded by $\Delta + \rho$. However, $e(\frac{n}{s} + p) \geq 3\Delta + 2\rho + \epsilon - \Delta > 2\Delta + \rho$. Since $\frac{n}{s} + p \in \mathcal{D}^+_{n,s}$, this contradicts the bound we established before. Therefore, there cannot be a point in $\mathcal{D}^-_{n,s}$ with error greater than $3\Delta + \rho$. A symmetric argument can be used to bound negative error.

Now we reduce the error bound to $2\Delta + \rho$. Assume that p is the minimal point in $\mathcal{D}^-_{n,s}$ with error at least $2\Delta + \rho$. The proof is similar to the previous stage, using the tighter bound $e(x) \leq 2\Delta + \rho/2$ for $x \in [-\frac{n}{2s}, 0]$. Cases (i) and (iii) stay the same; for case (ii) we have: $e(t + p) \leq -\epsilon \leq \Delta$. Therefore, the error cannot exceed $\Delta + \rho$ in $\mathcal{D}^+_{n,s}$. But $e(\frac{n}{s} + p) \geq 2\Delta + \epsilon + \rho - \Delta$, which is a contradiction. $\square$

The following special case proves the stability result for linearity:

Corollary 8 *The linearity property is* $(\mathcal{D}_{n,s}, \mathcal{D}_{n,s}, \Delta, 2\Delta)$*-stable.*

Proof. Suppose function g is Δ-approximately linear on $\mathcal{D}_{n,s}$. Set $h(\frac{n}{s}) = g(\frac{n}{s})$ in Lemma 7. This uniquely defines a linear h with $\rho = 0$. $\square$

The intuition that drives us to set $h(\frac{n}{s}) = g(\frac{n}{s})$ in the proof of Corollary 8 is as follows. Consider the following function of n, s: $g(\frac{n}{s}) = (\frac{n}{s} + \frac{[(n-1)/3]}{s})\Delta$ ($[x]$ denotes integer part of x). It is easy to see that $g(x + y) \approx_\Delta g(x) + g(y)$. Note that setting $h(\frac{1}{s}) = g(\frac{1}{s})$ instead of $h(\frac{n}{s}) = g(\frac{n}{s})$ does not work in general. If we set $h(\frac{1}{s}) = g(\frac{1}{s})$, then we obtain $h(\frac{n}{s}) = \frac{n}{s}\Delta$. But $||g - h||$ is a growing function of n and so there is no way to bound the error at all points.

The following example shows that the error bound obtained in Corollary 8 using our technique is tight: we have shown how to construct a linear function h so that $||h - g|| \leq 2\Delta$. We now show that there is a function g that, given our method of constructing h, asymptotically approaches this bound from below. Define g as follows: $g(n) = 0$; $g(x) = (3x/n - 1)\Delta$ for $0 \leq x \leq n - 1$; $g(-x) = -g(x)$ for $0 < x \leq n$. It is easy to see that g

is Δ-approximately linear: If $x+y<n$, $g(x+y)-g(x)-g(y)=\Delta$. If $x+y=n$, then $g(x+y)=0$ and so $g(x)+g(y)=\Delta$. Our construction sets $h(n)=0$; thus, $h\equiv 0$, the zero function. However, $\|g-h\| = |g(n-1)-h(n-1)| = (2-3/n)\Delta \longrightarrow 2\Delta$ for large enough n.

2.2 Approximate Multilinearity

In this section we focus our attention on *multilinear functions*. A multivariate function is multilinear if it is linear in any one input when all the other inputs are fixed. A multilinear function of k variables is called a k-linear function. An example of a *bilinear* function is multiplication, and bilinearity property can be stated concisely as $f(x_1+x_1', x_2+x_2') = f(x_1,x_2)+f(x_1',x_2)+f(x_1,x_2')+f(x_1',x_2')$. Note that distributivity of multiplication over addition is a special case of multilinearity.

A natural extension of this class of functions is the class of approximately multilinear functions, which are formally defined below:

Definition 9 (Approximate Multilinearity) *A k-variate function g is* Δ-approximately k-linear *on $\mathcal{D}_{n,s}^k$ if it is Δ-approximately linear on $\mathcal{D}_{n,s}$ in each variable.*

For instance, for $k=2$, a function g is Δ-approximately bilinear if $\forall x_1, x_1', x_2, x_2' \in \mathcal{D}$, $g(x_1+x_1', x_2) \approx_\Delta g(x_1,x_2)+g(x_1',x_2)$ and $g(x_1, x_2+x_2') \approx_\Delta g(x_1,x_2)+g(x_1,x_2')$.

Now we generalize Lemma 7 to Δ-approximately k-linear functions. Let g be a Δ-approximately k-linear function and h be the symmetric multilinear function uniquely defined by the condition $h(\frac{n}{s},\ldots,\frac{n}{s}) = g(\frac{n}{s},\ldots,\frac{n}{s})$. Let $e\equiv h-g$. e is a Δ-approximately k-linear function.

Since g takes k inputs from $\mathcal{D}_{n,s}$, if we consider each input to g as a coordinate, the set of all possible k-tuples of inputs of g form a $(2n+1)\times\cdots\times(2n+1)$ cube of dimension k. We show that for any point $(x_1,\ldots,x_k)$ in this cube, $|e(x_1,\ldots,x_k)|$ is bounded.

Theorem 10 *The approximate k-linearity property is $(\mathcal{D}_{n,s}^k, \mathcal{D}_{n,s}^k, \Delta, 2k\Delta)$-stable. In other words, if a function g is Δ-approximately k-linear on $\mathcal{D}_{n,s}^k$, then there exists a k-linear h on $\mathcal{D}_{n,s}^k$ such that $\|h-g\|\le 2k\Delta$.*

Proof. With h defined as above, $e(\frac{n}{s},\ldots,\frac{n}{s})=0$. First, we argue about points that have one coordinate that is different from $\frac{n}{s}$. Fix $k-1$ of the inputs to be $\frac{n}{s}$ (hard-wire into g) and vary one (say x_i). This operation transforms g from a Δ-approximately k-linear function of $x_1,\ldots,x_k$ to a Δ-approximately linear function of x_i. By Lemma 7, this function cannot have an error of more than 2Δ in $\mathcal{D}_{n,s}$. Therefore, $|e(\frac{n}{s},\ldots,\frac{n}{s},x_i,\frac{n}{s},\ldots,\frac{n}{s})|\le 2\Delta$, if $|x_i|<\frac{n}{s}$. Next we consider points which have two coordinates that are different from $\frac{n}{s}$. Consider without loss of generality an input $a,b,\frac{n}{s},\ldots,\frac{n}{s}$. By the result we just argued, we know that $e(\frac{n}{s},b,\frac{n}{s},\ldots,\frac{n}{s})\le 2\Delta$. By fixing inputs 2 through k to be $b,\frac{n}{s},\ldots,\frac{n}{s}$ and varying the first input, by Lemma 7, we have $|e(a,b,\frac{n}{s},\ldots,\frac{n}{s})|\le 4\Delta$ for any $a\in\mathcal{D}_{n,s}$. Via symmetric arguments, we can bound the error by 4Δ if any two inputs are different from $\frac{n}{s}$. Continuing this way, it can be shown that for all inputs, the error is at most $2k\Delta$. □

The following theorem shows that the error can be reduced to $(1+\mu)\Delta$ for any constant $\mu>0$ by imposing the multilinearity condition on a larger domain $\mathcal{D}'$ and fitting the multilinear function h on $\mathcal{D}$, where $|\mathcal{D}'|/|\mathcal{D}|=\lceil 2k/\mu\rceil$. Note that doubling the domain size only involves adding one more bit to the representation of a domain element.

Theorem 11 *For any $\mu>0$, the approximate multilinearity property is $(\mathcal{D}^k_{2kn/\mu,s},\mathcal{D}^k_{n,s},\Delta,(1+\mu)\Delta)$-stable.*

Proof. By Theorem 10, g is $2k\Delta$-close to a k-linear h on $\mathcal{D}_{2kn/\mu,s}$. For any $x=x_1,\ldots,x_k$, we fix all coordinates except x_i and argue in the i-th coordinate as below.

For any $\mathcal{D}_{m,s}$, first we show that if $|e(x)|_{\mathcal{D}_{m,s}}\le\rho$ then $|e(x)|_{\mathcal{D}_{m/2,s}}\le(\rho+\Delta)/2$. To observe this, note that if $x\in\mathcal{D}_{m/2,s}$, then $2x\in\mathcal{D}_{m,s}$. Therefore the function should satisfy $e(x)+e(x)\approx_\Delta e(2x)$, which implies that $|e(x)|\le(\rho+\Delta)/2$. Thus, in general, the maximum error in $\mathcal{D}_{m/2^i,s}$ is $\le\rho/2^i+\Delta(1-1/2^i)$. Since the error in $\mathcal{D}_{2kn/\mu,s}$ is at most $2k\Delta$, the error in $\mathcal{D}_{n,s}$ is at most $(1+\mu)\Delta$ by our choice of parameters. In the multilinear case, we can make a similar argument by using points which have at least one coordinate x_i within the smaller half of the axis. □

3 Polynomials

To test programs purportedly computing polynomials, it is tempting to (i) interpolate the polynomial from randomly chosen points, and then (ii) verify that the program is approximately equal to the interpolated polynomial for a large fraction of the inputs. Since a degree d k-variate polynomial can have $(d+1)^k$ terms, this leads to exponential running times. Furthermore, it is not obvious how error bounds that are *independent* of the domain size can be obtained.

Our test uses the same "evenly spaced" interpolation identity as that in [30]: f is a degree d polynomial if and only if for all $x, t \in \mathcal{D}$, $\sum_{i=0}^{d+1}(-1)^{d+1-i}\binom{d+1}{i} f(x+it) = 0$. This identity is computed by the method of successive differences which never explicitly interpolates the polynomial computed by the program, thus giving a particularly simple and efficient ($O(d^2)$ operations) test.

We can show that the interpolation identity is approximately robust by modifying the robustness theorem in [29]. (Section 3.3). Our proof of stability of the interpolation identity (Section 3.2), however, uses a characterization of polynomials in terms of multilinear functions that previously has not been applied to program checking. This in turn allows us to use our results on the stability of multilinearity (Section 2.2) and other ideas from stability theory. Section 3.4 extends these techniques to multivariate polynomials.

3.1 Preliminaries

In this section, we present the basic definitions and theorems that we will use. Define

$$\nabla_t f(x) \stackrel{\text{def}}{=} f(x+t) - f(x)$$

to be the standard *forward difference operator.* Let

$$\nabla_t^d f(x) \stackrel{\text{def}}{=} \overbrace{\nabla_t \cdots \nabla_t}^{d} f(x) = \sum_{k=0}^{d} (-1)^{d-k} \binom{d}{k} f(x+kt)$$

and $\nabla_{t_1,t_2} f(x) \stackrel{\text{def}}{=} \nabla_{t_1} \nabla_{t_2} f(x)$. The following are simple facts concerning this operator.

Facts 12 *The following are true for the difference operator* ∇*:*

1. *∇ is linear: $\nabla(f+g) = \nabla f + \nabla g$,*

2. *∇ is commutative: $\nabla_{t_1,t_2} = \nabla_{t_2,t_1}$, and*

3. *$\nabla_{t_1+t_2} - \nabla_{t_1} - \nabla_{t_2} = \nabla_{t_1,t_2} = \nabla_{t_2,t_1}$.*

Let $x^{[k]}$ denote $\overbrace{x,\ldots,x}^{k}$. For any k-ary symmetric f, let $f^*(x) = f(x^{[k]})$ denote its diagonal restriction. We use three different characterizations of polynomials [27, 15].

Fact 13 *Let $\mathcal{D}$ be a ring. The following are equivalent:*

1. *there exist $a_0,\ldots,a_d \in \mathcal{D}$ such that $\forall x \in \mathcal{D}, f(x) = \sum_{k=0}^{d} a_k x^k$,*

2. *$\forall x,t \in \mathcal{D}, \nabla_t^{d+1} f(x) = 0$,*

3. *there exist symmetric k-linear functions F_k, $0 \le k \le d$ such that $\forall x \in \mathcal{D}, f(x) = \sum_{k=0}^{d} F_k^*(x)$.*

The above fact remains true for non-closed domains so long as we insist that the arguments to f are from the domain.

The following definitions are motivated by the notions of using evenly and unevenly spaced points in interpolation.

Definition 14 (Strong Approximate Polynomial) *A function g is called* strongly Δ-approximately degree d polynomial on $\mathcal{D}$ *if $\forall x, t_1, \ldots, t_{d+1} \in \mathcal{D}$ such that $x+t_1+\cdots+t_{d+1} \in \mathcal{D}$, $|\nabla_{t_1,\ldots,t_{d+1}} g(x)| \le \Delta$.*

Definition 15 (Weak Approximate Polynomial) *A function g is called* weakly Δ-approximately degree d polynomial on $\mathcal{D}$ *if $\forall x,t \in \mathcal{D}$ such that $x + t(d+1) \in \mathcal{D}$, $|\nabla_t^{d+1} g(x)| \le \Delta$.*

3.2 Stability for Polynomials

First, we prove that if a function is strongly Δ-approximately polynomial then there is a polynomial that $(2^{d \lg d}\Delta, 0)$-approximates it. Next, we show that if a function is weakly approximately polynomial on a domain, then there is a coarser subdomain on which the function is strongly approximately polynomial. Combining these two, we can show that if a function is weakly approximately polynomial on a domain, then there is a subdomain on which the function approximates a polynomial. By using Theorem 11, we can bring the above error arbitrarily close to Δ by assuming the hypothesis on a large enough domain. In order to pass programs that err by at most Δ', we need to set $\Delta \geq (d+1) \cdot 2^d \Delta'$.

Strongly Approximate Case. One must be careful in defining polynomial h that is close to g. For instance, defining h based on the values of g at some $d+1$ points will not work. We proceed by modifying techniques in [2, 19], using the following fact:

Fact 16 *If a function f is symmetric and k-linear, then $\nabla_{t_1,\ldots,t_d} f^*(x) = k! f(t_1, \ldots, t_k)$ if $k = d$ and 0 if $k < d$.*

The following theorem shows the stability of the strong approximate polynomial property.

Theorem 17 *The strong approximate polynomial property is $(\mathcal{D}_{n(d+2),s}, \mathcal{D}_{n,s}, \Delta, O(2^{d \lg d})\Delta)$-stable. In other words, if g is a strongly Δ-approximately degree d polynomial on $\mathcal{D}_{n(d+2),s}$, then there is a degree d polynomial h_d such that $\|g - h_d\|_{\mathcal{D}_{n,s}} \leq O(2^{d \lg d})\Delta$.*

Proof. Note that if $x, t_1, \ldots, t_{d+1} \in \mathcal{D}_{n,s}$, then $x + t_1 + \cdots + t_{d+1} \in \mathcal{D}_{(d+2)n,s}$. Now, the hypothesis that g is a strongly Δ-approximately degree d polynomial on $\mathcal{D}_{n(d+2),s}$ guarantees that $\forall x, t_1, \ldots, t_{d+1} \in \mathcal{D}_{n,s}$, $|\nabla_{t_1,\ldots,t_{d+1}} g(x)| \leq \Delta$. The rest of the proof uses this "modified hypothesis" and works with $\mathcal{D}_{n,s}$.

We induct on the degree. Let $e_d \overset{\text{def}}{=} |g - h_d|$. When $d = 0$, by the modified hypothesis, we have $\forall x, t \in \mathcal{D}_{n,s}, |\nabla_t g(x)| \leq \Delta$ i.e., $|\nabla_t g(0)| = |g(t) - g(0)| \leq \Delta$ for all $t \in \mathcal{D}_{n,s}$. Setting $h_0 = g(0)$, a constant, we are done.

Suppose the lemma holds when the degree is strictly less than $d+1$. Now, by the modified hypothesis, we have $\forall t_1, \ldots, t_{d+1} \in \mathcal{D}_{n,s}$, $|\nabla_{t_1,\ldots,t_{d+1}} g(x)| \leq \Delta$. Using Fact 12 and then our modified hypothesis, we have $|\nabla_{t_1+t'_1,t_2,\ldots,t_d} g(x) - \nabla_{t_1,t_2,\ldots,t_d} g(x) - \nabla_{t'_1,t_2,\ldots,t_d} g(x)| = |\nabla_{t_1,t'_1,\ldots,t_d} g(x)| \leq \Delta$. By symmetry of the difference operator, we have a Δ-approximate symmetric d-linear function on $\mathcal{D}_{n,s}$, say $G(t_1, \ldots, t_d) \overset{\text{def}}{=} \nabla_{t_1,\ldots,t_d} g(0)$. Theorem 10 on multilinearity guarantees a symmetric d-linear H with $\|G - H\| \leq 2d\Delta$. Let $H_d(x_1, \ldots, x_d) = H(x_1, \ldots, x_d)/d!$. Let $g'(x) = g(x) - H_d^*(x)$ for $x \in \mathcal{D}_{n,s}$.

Now, we have $\forall x, t_1, \ldots, t_d \in \mathcal{D}_{n,s}$,

$$\begin{aligned}
|\nabla_{t_1,\ldots,t_d} g'(x)| &= |\nabla_{t_1,\ldots,t_d}(g(x) - H_d^*(x))| \qquad \text{(definition of } g'\text{)} \\
&\le |\nabla_{t_1,\ldots,t_d} g(x) - \nabla_{t_1,\ldots,t_d} g(0)| + |\nabla_{t_1,\ldots,t_d} g(0) - \nabla_{t_1,\ldots,t_d} H_d^*(x)| \qquad \text{(triangle inequality)} \\
&= |\nabla_{t_1,\ldots,t_d,x} g(0)| + |\nabla_{t_1,\ldots,t_d} g(0) - \nabla_{t_1,\ldots,t_d} H_d^*(x)| \qquad \text{(definition of } \nabla\text{)} \\
&= |\nabla_{t_1,\ldots,t_d,x} g(0)| + |G(t_1,\ldots,t_d) - \nabla_{t_1,\ldots,t_d} H_d^*(x)| \qquad \text{(definition of } G\text{)} \\
&= |\nabla_{t_1,\ldots,t_d,x} g(0)| + |G(t_1,\ldots,t_d) - d! H_d(t_1,\ldots,t_d)| \qquad \text{(Fact 16)} \\
&= |\nabla_{t_1,\ldots,t_d,x} g(0)| + |G(t_1,\ldots,t_d) - H(t_1,\ldots,t_d)| \qquad \text{(definition of } H_d\text{)} \\
&\le \Delta + |G(t_1,\ldots,t_d) - H(t_1,\ldots,t_d)| \qquad \text{(modified hypothesis on } g\text{)} \\
&\le (2d+1)\Delta \qquad \text{(since } \|G - H\| \le 2d\Delta\text{)}.
\end{aligned}$$

Now we apply the induction hypothesis. g' satisfies the hypothesis above for d and larger error $\Delta' = (2d+1)\Delta$ and so by induction, we are guaranteed the existence of a degree $d-1$ polynomial h_{d-1} such that $\|g' - h_{d-1}\| \le e_{d-1}\Delta'$. Set $h_d = h_{d-1} + H_d^*$. By Fact 13(3) about the characterization of polynomials, h_d is a degree d polynomial. Now, $e_d = \|g - h_d\| = \|g - h_{d-1} - H_d^*\| = \|g' - h_{d-1}\| \le e_{d-1}\Delta' = e_{d-1}(2d+1)\Delta$.

Unwinding the recurrence, the final error $\|g - h_d\| = \Delta \prod_{i=1}^{d}(2i+1)$. □

Weakly Approximate Case. We first need the following useful fact [15] which helps us to go from equally spaced points to unequally spaced points:

Fact 18 *For any $\lambda_1, \ldots, \lambda_d \in \{0,1\}$, if $t'_{\lambda_1,\ldots,\lambda_d} = -\sum_{i=1}^{d} \lambda_i t_i / i$ and $t''_{\lambda_1,\ldots,\lambda_d} = \sum_{i=1}^{d} \lambda_i t_i$ then*

$$\nabla_{t_1,\ldots,t_d} f(x) = \sum_{\lambda_1,\ldots,\lambda_d \in \{0,1\}} (-1)^{\lambda_1 + \cdots + \lambda_d} \nabla^d_{t'_{\lambda_1,\ldots,\lambda_d}} f(x + t''_{\lambda_1,\ldots,\lambda_d}).$$

Using this fact, we obtain the following theorem. Let $\mu(d) = \mathrm{lcm}\{1, 2, \ldots, d\}$.

Theorem 19 *If g is weakly $(\Delta/2^{d+1})$-approximately degree d polynomial on $\mathcal{D}_{n(d+1),s\mu(d+1)}$, then g is strongly Δ-approximately degree d polynomial on $\mathcal{D}_{n,s}$.*

Proof. For $t_1, \ldots, t_{d+1} \in \mathcal{D}_{n,s}$, and for any $\lambda_1, \ldots, \lambda_{d+1} \in \{0,1\}$, we have by our choice of parameters that $t'_{\lambda_1,\ldots,\lambda_{d+1}}, t''_{\lambda_1,\ldots,\lambda_{d+1}} \in \mathcal{D}_{n(d+1),s\mu(d+1)}$. Therefore, for $x \in \mathcal{D}_{n,s}$,

$$|\nabla_{t_1,\ldots,t_{d+1}} g(x)| \le \sum_{\lambda_1,\ldots,\lambda_{d+1} \in \{0,1\}} |\nabla^{d+1}_{t'_{\lambda_1,\ldots,\lambda_{d+1}}} g(x + t''_{\lambda_1,\ldots,\lambda_{d+1}})| \le 2^{d+1}(\Delta/2^{d+1}) \le \Delta.$$

□

3.3 Approximate Robustness for Polynomials

This section shows that the interpolation equation for degree d polynomials is in some sense, approximately robust. All the results in this subsection are modifications of the exact robustness of polynomials given in [29]. Let $\alpha_k = (-1)^{d+1-k}\binom{d+1}{k}$. To self-test P on $\mathcal{D}_{n,s}$, we use the following domains. These domains are used for technical reasons that will become apparent in the proofs of the theorems in this section.

1. $\mathcal{D}_{(d+2)n,s}$
2. $\mathcal{T} = \mathcal{D}_{K_n,L_s}$ where $K_n = n(d+2)(n(d+1)!)^3$ and $L_s = s((d+1)!)^3$
3. $\mathcal{T}_j = \{jx : x \in \mathcal{T}\}$ for $0 \le j \le d+1$
4. $\mathcal{T}_{i,j} = \{ix : x \in \mathcal{T}_j\}$ for $0 \le i, j \le d+1$

All $\mathcal{T}_j, \mathcal{T}_{i,j}$ contain $\mathcal{D}_{(d+2)n,s}$. Now, assume that P satisfies the following properties, which are similar to the low-degree test in an approximate setting and over different domains. Note that these properties can be tested by sampling. We use $\Pr_{x\in\mathcal{D}}[\cdot]$ to denote the probability of an event when x is chosen uniformly from domain $\mathcal{D}$.

1. $\Pr_{0\le k\le d+1, x\in\mathcal{D}_{n,s}, t\in\mathcal{T}_k}[\sum_{i=0}^{d+1} \alpha_i P(x+it) \approx_\Delta 0] \ge 1-\epsilon$,
2. for each $0 \le j \le d+1$, $\Pr_{0\le k,l\le d+1, x\in\mathcal{T}_{k,j}, t\in\mathcal{T}_l}[\sum_{i=0}^{d+1} \alpha_i P(x+it) \approx_\Delta 0] \ge 1-\epsilon$, and
3. for each $0 \le i, j \le d+1$, $\Pr_{0\le k\le d+1, x\in\mathcal{T}_{i,j}, t\in\mathcal{T}_k}[\sum_{l=0}^{d+1} \alpha_l P(x+lt) \approx_\Delta 0] \ge 1-\epsilon$.

Define $g(x) = \operatorname{median}_{0\le k\le d+1, t\in\mathcal{T}_k}\{\sum_{i=1}^{d+1} \alpha_i P(x+it)\}$. We obtain the following theorem that shows the approximate robustness of polynomials. Let $\mathcal{E}_{\mathcal{T}(n,s)}$ be the distribution that flips a fair three-sided die and on outcome $i \in \{1,2,3\}$, chooses inputs according to distribution given in the i-th property above. Let $\mathcal{D}_{\mathcal{T}(n,s)}$ be the union of the domains used in the above properties.

Theorem 20 *The interpolation equation, where inputs are picked according to the distribution $\mathcal{E}_{\mathcal{T}(n,s)}$, is $(2\epsilon, \epsilon, \mathcal{D}_{\mathcal{T}(n,s)}, \mathcal{D}_{n,s}, \Delta, 2^{d+3}\Delta, \Delta)$-approximately robust.*

The rest of this section is devoted to proving the above theorem.

By Markov's inequality, g's definition, and properties (1) and (3) of P, it is easy to show that P $(\Delta, 2\epsilon)$-approximates g:

Theorem 21 *If program P satisfies the above three properties, then, for all $i, j \in \{0, \ldots, d+1\}$, $\Pr_{x \in \mathcal{T}_{i,j}}[P(x) \approx_\Delta g(x)] \geq 1 - 2\epsilon$ and $\Pr_{x \in \mathcal{D}_{n,s}}[P(x) \approx_\Delta g(x)] \geq 1 - 2\epsilon$.*

Now, we set out to prove that g is a weakly approximate polynomial. Let $\delta(p_1, p_2) = p_1$ if $p_1 = p_2$ and 0 otherwise. For two domains $\mathcal{A}, \mathcal{B}$, subsets of a universe $\mathcal{X}$, let $\delta(\mathcal{A}, \mathcal{B}) = \sum_{s \in \mathcal{X}} \delta(\Pr_{x \in \mathcal{A}}[x = s], \Pr_{y \in \mathcal{B}}[y = s])$ and call the domains *ϵ-close* if $\delta(\mathcal{A}, \mathcal{B})$ is at least $1 - \epsilon$. Using the definitions of $\mathcal{T}, \mathcal{T}_j, \mathcal{T}_{i,j}$, the following fact can be shown:

Fact 22 *For any $x \in \mathcal{D}_{(d+2)n,s}$, the domains $\mathcal{T}_j$ and $\{x + t : t \in \mathcal{T}_j\}$ are $\epsilon_1 = O(1/n^3)$-close. For any x, the domains $\mathcal{T}_{i,j}$ and $\{x + t : t \in \mathcal{T}_{i,j}\}$ are $\epsilon_2 = O(1/n^3)$-close.*

The following lemma shows that, in some sense, g is well-defined, and links it to an interpolation obtained from P:

Lemma 23 *For all $x \in \mathcal{D}_{(d+2)n,s}$, $\Pr_{0 \leq k \leq d+1, t \in \mathcal{T}_k}[g(x) \approx_{2^{d+2}\Delta} \sum_{j=1}^{d+1} \alpha_j P(x + jt)] \geq 1 - \epsilon_3$ and $\forall i, \Pr_{t \in \mathcal{T}_i}[g(x) \approx_{2^{d+2}\Delta} \sum_{j=1}^{d+1} \alpha_j P(x + jt)] \geq 1 - \epsilon_4$ where $\epsilon_3 = 2(d+1)(\epsilon + \epsilon_2)$ and $\epsilon_4 = (d+1)\epsilon_3$.*

Proof. Consider $0 \leq k, l \leq d+1$ and $t_1 \in \mathcal{T}_k, t_2 \in \mathcal{T}_l$. For a fixed $0 \leq j \leq d+1$, using properties of P, and since $\mathcal{T}_{j,k}$ and $\{x + jt_1 : t_1 \in \mathcal{T}_k\}$ are ϵ_2-close (Fact 22), we get $\Pr[P(x + jt_1) \approx_\Delta \sum_{i=1}^{d+1} \alpha_i P(x + jt_1 + it_2)] \geq 1 - \epsilon - \epsilon_2$ and $\Pr[P(x + it_2) \approx_\Delta \sum_{j=1}^{d+1} \alpha_j P(x + jt_1 + it_2)] \geq 1 - \epsilon - \epsilon_2$. Summing over all $0 \leq i, j \leq d+1$ and noting that $\sum_{i=1}^{d+1} \alpha_j \Delta \leq 2^{d+1}\Delta$, $\Pr[\sum_{j=1}^{d+1} \alpha_j P(x + jt_1) \approx_{2^{d+1}\Delta} \sum_{i=1}^{d+1} \alpha_i P(x + it_2)] \geq 1 - 2(d+1)(\epsilon + \epsilon_2) = 1 - \epsilon_3$. Using Lemma 43 (see Section 4.3), we can show that with a relaxation of twice the error, this probability lower bounds the probability in the first part of the lemma. The second part of the lemma follows from the first via a simple averaging argument. □

Now, the following theorem completes the proof that g is a weakly approximate degree d polynomial.

Theorem 24 *For all* $x \in \mathcal{D}_{(d+2)n,s}$*,* $\forall i \in \{0, \ldots, d+1\}$*,* $\Pr_{t \in \mathcal{T}_i}[g(x) \approx_{2^{d+3}\Delta} \sum_{j=1}^{d+1} \alpha_j g(x+jt)] \geq 1 - \epsilon_5$ *where* $\epsilon_5 = \epsilon_4 + (d+1)(2\epsilon + \epsilon_2)$ *and* $\forall x, t \in \mathcal{D}_{n,s}$*,* $\Pr_{t_1 \in \mathcal{T}}[|\nabla_t^{d+1} g(x)| \leq 2^{d+3}\Delta] \geq 1 - (d+1)(2\epsilon_5 + \epsilon_1)$

Proof. It is implied by Theorem 21, Lemma 23 and the closeness of the domains $\mathcal{T}_{i,j}$ and $\{x + t : t \in \mathcal{T}_{i,j}\}$ that $\forall x \in \mathcal{D}_{(d+2)n,s}, \forall i, \Pr_{t \in \mathcal{T}_i}[g(x) \approx_{2^{d+2}\Delta} \sum_{j=1}^{d+1} \alpha_j P(x+jt)] \geq 1 - \epsilon_4$ and $\Pr_{t \in \mathcal{T}_i}[g(x+jt) \approx_\Delta P(x+jt)] \geq 1 - 2\epsilon - \epsilon_2$. Summing the latter expression and putting them together, we have the first part of the lemma. The second part follows from the first part and the fact that $\mathcal{T}_j$ and $\{t + jt_1 : t_1 \in \mathcal{T}\}$ are ϵ_1-close (Fact 22). $\square$

For an appropriate choice of $\epsilon, \epsilon_1, \epsilon_2$, we have a g that is a weakly $(2^{d+3}\Delta)$-approximately degree d polynomial on $\mathcal{D}_{n,s}$ with g $(\Delta, 2\epsilon)$-approximating P on $\mathcal{D}_{n,s}$.

3.4 Multivariate Polynomials

The following approach is illustrated for bivariate polynomials. We can easily generalize this to multivariate polynomials. It is easy to show that the approximate robustness holds when the interpolation equation [30] is used as in Section 3.3, i.e., for any k-variate polynomial P of total degree d, the following interpolation equation is satisfied for all $\bar{x}, \bar{t} \in \mathcal{D}_{n,s}^k$:

$$\sum_{i=0}^{d+1} \alpha_i P(\bar{x} + i\bar{t}) = 0.$$

An *horizontal axis parallel line* for a fixed y is the set of points $l_{x,h} = \{(x + kh, y) : k \in \mathbb{Z}\}$. A vertical axis parallel line is defined analogously. As a consequence of approximate robustness, we have a bivariate function $g(x, y)$ that is a strongly approximately degree d polynomial along every horizontal and vertical line. We use this consequence to prove stability.

The characterization we will use is: $f(x, y)$ is a bivariate polynomial (assume degree in both x and y is d) if and only if there are $d+1$ symmetric k-linear functions $F_k(y_1, \ldots, y_k) : \mathcal{D}^k \to \mathcal{P}_\mathcal{D}[x]$ where the range is the space of all degree d univariate polynomials in x.

For each value of y, $g_y(x)$ is a strongly approximately degree d polynomial. Using the univariate case (Theorem 17), there is an exact degree d polynomial $P_y(x)$ such that for all x, $g(x, y) \approx_{2^{d \lg d}\Delta} P_y(x)$. Construct the function $g'(x, y) = P_y(x)$. Let $\Delta' = 2^{d \lg d}\Delta$.

Now, for a fixed x (i.e., on vertical line) for any y, using $\nabla_{t_1,\ldots,t_{d+1}} g(x,y) \approx_\Delta 0$, we have $\nabla_{t_1,\ldots,t_{d+1}} g'(x,y) \approx_{\Delta'} 0$. Thus, $g'(x,y)$ is a bivariate function where along every horizontal line, it is an *exact* degree d polynomial and along every vertical line, it is a strongly Δ'-approximate degree d polynomial. Interpreting $g'(x,y)$ as $g'_x(y)$ and using the same idea as in univariate case, we can conclude that $\nabla(t_1,\ldots,t_d) : \mathcal{D}^d \to \mathcal{P}_\mathcal{D}[x]$ is a symmetric approximate d-linear function (here, we used the fact that $g'_x(y) \in \mathcal{P}_\mathcal{D}[x]$). The rest of the argument in Theorem 17 goes through because our proofs of approximate linearity (Lemma 7) and multilinearity (Theorem 10) assume that the range is a metric space (which is true for $\mathcal{P}_\mathcal{D}[x]$ with, say, the Chebyshev norm). The result follows from the above characterization of bivariate polynomials.

4 Functional Equations

Extending the technique in Lemma 7 to addition theorems $f(x+y) = G[f(x), f(y)]$ is not straightforward, since G can be an arbitrary function. In order to prove approximate robustness (Section 4.3) and stability (Section 4.2), several related properties of G are required. Proving that G satisfies each individual one is tedious; however, the notion of *modulus of continuity* from approximation theory gives a general approach to this problem. We show that bounds on the modulus of continuity imply bounds on all of the quantities of G that we require. The stability of G is shown by a careful inductive technique based on a canonical generation of the elements in $\mathcal{D}_{n,s}$ (Section 4.2). The scope of our techniques is not only limited to addition theorems; we also show that Jensen's equation is approximately robust and stable. (Section 4.2.4)

4.1 Preliminaries

For addition theorems, we can assume that G is algebraic and a symmetric function (the latter is true in general under some technical assumptions as in [28]). We need a notion of "smoothness" of G. The following notions are well-known in approximation theory [25, 33].

Definitions 25 (Moduli of Continuity) *The* modulus of continuity *of the function $f : \mathcal{D} \to \mathbb{R}$ is the following function of $\delta \in [0,\infty)$:*

$$\omega(f;\delta) = \sup_{\substack{|x_1-x_2|\le\delta \\ x_1,x_2\in\mathcal{D}}} \{|f(x_1) - f(x_2)|\}.$$

The modulus of continuity *of the function $f : \mathcal{D}^2 \to \mathbb{R}$ is the following function of $\delta_x, \delta_y \in [0,\infty)^2$:*

$$\omega(f;\delta_x,\delta_y) = \sup_{\substack{|x_1-x_2|\le\delta_x, |y_1-y_2|\le\delta_y \\ x_1,y_1,x_2,y_2\in\mathcal{D}}} \{|f(x_1,y_1) - f(x_2,y_2)|\}.$$

The partial moduli of continuity *of the function $f : \mathcal{D}^2 \to \mathbb{R}$ are the following functions of $\delta \in [0,\infty)$:*

$$\omega(f;\delta,0) = \sup_{y\in\mathcal{D}} \sup_{\substack{|x_1-x_2|\le\delta \\ x_1,x_2\in\mathcal{D}}} \{|f(x_1,y) - f(x_2,y)|\} \text{ and } \omega(f;0,\delta) = \sup_{x\in\mathcal{D}} \sup_{\substack{|y_1-y_2|\le\delta \\ y_1,y_2\in\mathcal{D}}} \{|f(x,y_1) - f(x,y_2)|\}.$$

We now present some facts which are easily proved.

Facts 26 *The following are true of the modulus of continuity:*

1. $0 \le \omega(f;\delta) \le \omega(f;\delta')$ *if* $\delta \le \delta'$;
2. *If f', the derivative of f exists, and is bounded in $\mathcal{D}$, then $\omega(f;\delta) \le \delta\|f'\|_{\mathcal{D}}$;*
3. $\omega(f;\delta,\delta) \le \omega(f;0,\delta) + \omega(f;\delta,0)$, *and if $f(\cdot,\cdot)$ is symmetric, then $\omega(f;\delta,\delta) \le 2\omega(f;\delta,0)$; and*
4. *If f'_x is the partial derivative of f with respect to x, then $\omega(f;\delta,0) \le \delta\|f'_x\|_{\mathcal{D}}$.*

We need a notion of an "inverse" of G. If $G[x,y] = z$, denote $G_1^{-1}[z,y] = x, G_2^{-1}[x,z] = y$. Since G is symmetric, $G_1^{-1} \equiv G_2^{-1}$ and we denote $G^{-1}[z,y] = x$.

An Example. Wherever necessary, we will illustrate our scheme using the functional equation $f(x+y) = \dfrac{f(x)f(y)}{f(x)+f(y)}$, i.e., $G[x,y] = xy/(x+y)$. The solution to this functional equation is $f(x) = C/x$ for some constant C. The following fact [34] is useful in locating the maxima of analytic functions.

Fact 27 (Maximum Modulus Principle) *If f is analytic in a compact set $\mathcal{D}$, then f attains extremum only on the boundary of $\mathcal{D}$.*

Over a bounded rectangle $\mathcal{D} = [L, U]^2$, where $0 < L \leq U$, G is analytic and hence by Fact 27, attains its maximum on the boundary. $G \in C^1[L, U]$ in $\mathcal{D}$ (i.e., is continuously differentiable). We have $G'_x[x, y] = y^2/(x + y)^2$ which is a decreasing function of x. By Fact 27, $\|G'_x\|$ attains a maximum when $x = L$, giving $\widehat{G} = \|G'_x(\cdot, y)\| = y^2/(L + y)^2$. Therefore, using Fact 26(4), $\omega(G; \delta, 0) \leq \sup\limits_{y \in [L,U]} \dfrac{\delta y^2}{(L + y)^2} = \dfrac{\delta U^2}{(L + U)^2}$.

4.2 Stability for Functional Equations

In this section, we prove (under some assumptions) that, if a function g satisfies a functional equation approximately everywhere, then it is close to a function h that satisfies the functional equation exactly everywhere. Our functional equations are of the form $g(x + y) = G[g(x), g(y)]$, where G is a symmetric algebraic function.

Example. If g satisfies $g(x + y) \approx_\Delta \dfrac{g(x)g(y)}{g(x) + g(y)}$ for some $\Delta > 0$ and for all valid x, y, then there is a function h such that $h(x + y) = \dfrac{h(x)h(y)}{h(x) + h(y)}$ for all valid x, y, and $h(x) \approx_{\Delta'} g(x)$ for some $\Delta' > 0$ and all valid x. The domains for the valid values of x, y, as well as the relationship between Δ and Δ' will be discussed later.

In the following sections we show how to construct the function h that is close to g, satisfying a particular functional equation. Given such an h, let $e(x)$ denote $|h(x) - g(x)|$, i.e., $h(x) \approx_{e(x)} g(x)$. For simplicity, let $H_1(x) \overset{\text{def}}{=} G[x, x]$. Note that $H_1(h(x)) = h(2x)$. We assume that $\omega(H_1; \delta) \leq c\delta$; our results thus hold for functions where the modulus of continuity is linear in δ. We will be making this assumption for our moduli of continuity when appropriate.

We consider the cases when $c < 1$, $c = 1$, and $c > 1$, first show how to obtain h, and then obtain bounds on $e(x)$. Then, we can conclude that h, which satisfies the functional equation everywhere, also approximates g; i.e., the functional equation is stable.

Call $\frac{x}{s}$ even (resp. odd) if x is even (resp. odd).

4.2.1 When $c < 1$

We begin by assuming that n is a power of 2, i.e., let $n = 2^k$ in $\mathcal{D}_{n,s}$. We first construct h by setting $h(\frac{1}{s}) = g(\frac{1}{s})$. This determines h for all values in $\mathcal{D}$ by the fact that h satisfies the functional equation.

We obtain a relationship between the error at x and $2x$ using the functional equation.

Lemma 28 *$e(2x) \leq ce(x) + \Delta$.*

Proof. $e(2x) = |g(2x) - h(2x)| \leq \Delta + |G[g(x), g(x)] - G[h(x), h(x)]|$. But, rewriting, and using the definition of the modulus of continuity, $|H_1(g(x)) - H_1(h(x))| \leq \omega(H_1; e(x)) \leq ce(x)$ □

We explore the relationship between $e(x+\frac{1}{s})$ and $e(x)$. For simplicity, let $H_2(x) \overset{\text{def}}{=} G[x, g(\frac{1}{s})]$. Note that $H_2(h(x)) = h(x+1)$. We again consider functions where the modulus of continuity is bounded by a linear function in δ, i.e., $\omega(H_2; \delta) = |H_2'(\cdot, g(\frac{1}{s}))| \leq d\delta$ for some constant d. Now,

Lemma 29 *$e(x + \frac{1}{s}) \leq de(x) + \Delta$.*

Proof. $e(x + \frac{1}{s}) = |g(x + \frac{1}{s}) - h(x + \frac{1}{s})| \leq \Delta + |G[g(x), g(\frac{1}{s})] - G[h(x), h(\frac{1}{s})]|$. But, $|G[g(x), g(\frac{1}{s})] - G[h(x), h(\frac{1}{s})] = |H_2[g(x)] - H_2[h(x)]| \leq \omega(H_2; e(x)) \leq de(x)$. □

We will show a scheme to bound $e(x)$ for all x when $d < 1$. This scheme can be thought of as an enumeration scheme, where at each step of the process, certain constraint equations have to be satisfied. We construct a binary tree T_k with nodes labeled with elements from $\mathcal{D}^+_{n,s}$ where $2^k = n$. The root is labeled $\frac{1}{s}$. If x is the label of a node, then $2x$ is the label of its left child (if $2x$ is not already in the tree). and $x + \frac{1}{s}$ is the label of its right child (if $x + \frac{1}{s}$ is not already in the tree). It is easy to see that, if x is even (except root), then x is a left child; if x is odd, then x is a right child.

Lemma 30 *Let $\omega(H_1; \delta) \leq c\delta, \omega(H_2; \delta) \leq d\delta$ with $c, d < 1$. For all $x \in \mathcal{D}^+_{n,s}$, if x is even, then $e(x) \leq \frac{1+c}{1-c}\Delta$ and if x is odd, then $e(x) \leq \frac{2}{1-c}\Delta$.*

Proof. We will prove this by induction on the preorder enumeration of T_k. Let x be the next element to be enumerated. By preorder listing, its parent has already been enumerated and hence, its error is known. If $x = 2y$ is even, it is a left child, and hence generated by a $2y$ operation. $e(y) \leq \frac{2}{1-c}\Delta$ by the induction hypothesis. This together with Lemma 28 yields $e(x) \leq ce(y) + \Delta \leq c\frac{2}{1-c}\Delta + \Delta \leq \frac{1+c}{1-c}\Delta$, preserving the induction hypothesis. If $x = y + \frac{1}{s}$

is odd, it is a right child, and hence generated by a $y+1$ operation. But, y is even, so $e(y) \leq \frac{1+c}{1-c}\Delta$ by the induction hypothesis. This together with Lemma 29 and $d \leq 1$ yields $e(x) \leq de(y) + \Delta \leq e(y) + \Delta \leq \frac{1+c}{1-c}\Delta + \Delta \leq \frac{2}{1-c}\Delta$, preserving the induction hypothesis. □

This yields the following theorem:

Theorem 31 *Let $\omega(H_1;\delta) \leq c\delta, \omega(H_2;\delta) \leq d\delta$ with $c, d < 1$ and let n be a power of 2. Then, the addition theorem is $(\mathcal{D}^+_{n,s}, \mathcal{D}^+_{n,s}, \Delta, \frac{2}{1-c}\Delta)$-stable.*

With our example, we have $H_1(x) = G[x,x] = x/2$ and so $c = 1/2$. Also, $H_2(x) = G[x, g(\frac{1}{s})]$ from which $H_2'(x) \leq 1$ as $0 < L \leq x, g(\frac{1}{s})$. Thus, $d \leq 1$. By Theorem 4.2.1, we have $e(x) \leq 4\Delta$ for all $x \in \mathcal{D}^+_{n,s}$.

When n is not a power of 2, we can argue in the following manner. From our proof, we see that we use very specific values of x, y in the approximate functional equation. Let i be such that $2^{i-1} \leq n \leq 2^i$ and let $\mathcal{D}' = \mathcal{D}_{2^i,s}$. We extend $\mathcal{D}^+_{n,s}$ to $\mathcal{D}'$ and define values of g at $\mathcal{D}' \backslash \mathcal{D}$: at even x $(= 2y)$ let $g(x) = H_1(g(y))$ and at odd x $(= y+1)$ let $g(x) = H_2(g(y))$. These can be thought of new assumptions on g which are satisfied "exactly" (i.e., without error Δ). We can use Lemma 30 to conclude that there is a linear h on $\mathcal{D}'$ that is $\frac{2}{1-c}\Delta$ close to g. Hence, h is close to g even on $\mathcal{D}^+_{n,s}$. To argue about $\mathcal{D}^-_{n,s}$, we pick a "pivot" point in $\mathcal{D}_{n,s}$ (0 for simplicity). Now, we have $h(-x) = G^{-1}[h(x), h(0)]$. Therefore, as in Theorem , we have $e(-x) \leq \omega(G^{-1}; \frac{2c}{1-c})$.

When $d \geq 1$, the error can no longer be bounded. In this case, we have $c \leq 1 < d$. Let $r = cd$. We can see from the structure of T_k that the maximum error can occur at $\frac{2^k-1}{s}$. By simple induction on the depth of the tree, the error is given by $e(\frac{2^k-1}{s}) \leq \sum_{i=0}^{k-2}(d^{i+1} + d^i)c^i\Delta = (d+1)\sum_{i=0}^{k-2} r^i\Delta = (d+1)\frac{r^{k-1}-1}{r-1}\Delta$. If $r < 1$, we obtain a constant error bound of $e(x) \leq (d+1)\frac{1}{1-r}\Delta$ by geometric summation. Otherwise, we obtain $e(x) = O(r^{\lg n})$.

4.2.2 When $c > 1$

In this case, we require additional assumptions. We define the quantity

$$\omega^{-1}(f;\delta) = \sup_{\substack{|f(x_1)-f(x_2)|\leq\delta \\ x_1,x_2\in\mathcal{D}}} \{|x_1 - x_2|\}.$$

Note that $\omega(f;\delta) \leq c\delta$ implies $\omega^{-1}(f;\delta) \geq \delta/c$. Now, we assume that $\omega^{-1}(f;\delta) \leq \delta/c'$, for some $c' > 1$.

Set $h(\frac{2^k}{s}) = g(\frac{2^k}{s})$. Since h satisfies the addition theorem, this can be used to fix all of h, if H_1^{-1} is well-defined. Let $e(x) = |g(x) - h(x)|$ as before.

As before, we first obtain a relationship between the error at x and at $2x$ using the addition theorem.

Lemma 32 $e(x) \leq (e(2x) + \Delta)/c'$.

Proof. We have as in Lemma 28, $|H_1(g(x)) - H_1(h(x))| \leq e(2x) + \Delta$. By definition of ω^{-1} and our assumption, we get $e(x) \leq \omega^{-1}(H_1; e(2x) + \Delta) \leq (e(2x) + \Delta)/c'$. □

For simplicity, let $H_3(x) \overset{\text{def}}{=} G^{-1}[g(\frac{2^k}{s}), \frac{2^k}{s} - x]$. We assume that $\omega(H_3;\delta) \leq d\delta$ for some constant d. The following lemma can be proved easily.

Lemma 33 $e(x) \leq de(\frac{2^k}{s} - x) + \Delta$.

$e(\frac{2^k}{s}) = 0$ by our construction. We adopt a scheme similar to the one in the previous section. Construct a binary tree T_k with nodes labeled with elements from $\mathcal{D}^+_{n,s}$. The root is labeled $\frac{2^k}{s}$. If x is the label of a node and x is even, then $x/2$ is the label of its left child (if $x/2$ is not already in the tree). and $\frac{2^k}{s} - x$ is the label of its right child (if $\frac{2^k}{s} - x$ is not already in the tree). It is easy to see that if $x \leq \frac{2^{k-1}}{s}$ (except the root), then x is a left child and if $x > \frac{2^{k-1}}{s}$, then x is a right child. We use the preorder enumeration of $\mathcal{D}^+_{n,s}$ using T_k to prove the following lemma, in the spirit of the proof of Lemma 30.

Lemma 34 *For all $x \in \mathcal{D}^+_{n,s}$, if $x \leq \frac{2^{k-1}}{s}$ and $d \leq 1$, then $e(x) \leq \frac{2c'}{1-c'}\Delta$ and if $x > \frac{2^{k-1}}{s}$ then $e(x) \leq \frac{1+c'}{1-c'}\Delta$.*

This yields (under the assumptions on $\omega^{-1}(H_1;\delta)$ and $\omega(H_3;\delta)$), the following theorem:

Theorem 35 *Let $\omega^{-1}(H_1;\delta) \leq \delta/c', \omega(H_3,\delta) \leq d\delta$ with $d \leq 1$ and let n be a power of 2. Then, the addition theorem is $(\mathcal{D}^+_{n,s}, \mathcal{D}^+_{n,s}, \Delta, \frac{1+c'}{1-c'}\Delta)$-stable.*

This case arises for linearity where $H_1(x) = G[x,x] = 2x$ and so $c' = 2$. Using the above theorem, we get a weaker bound of $e(x) \leq 3\Delta$ (as opposed to $\leq 2\Delta$ by Corollary 8). Similar techniques as in previous section can be used to argue about $\mathcal{D}^-_{n,s}$ and when n is not a power of 2.

The case when $d > 1$ can be handled by schemes as in the previous section.

4.2.3 When $c = 1$

In this case, it means that $\omega(H_1; \delta) = \delta$ or in other words, by Fact 26(2), $||H_1'|| = 1$. By Fact 27, the maximum occurs only at the boundary of the domain. Hence, we can test by looking at a subdomain in which the maximum is less than 1.

4.2.4 Jensen's Equation

Jensen's equation is the following: $\forall x, y \in \mathcal{D}_{n,s}, f(\frac{x+y}{2}) = \frac{f(x)+f(y)}{2}$. The solution to this functional equation is the set of affine linear functions i.e., $f(x) = ax + b$ for some constants a, b. Jensen's equation can be proved approximately robust by modifying the proof of its robustness in [28]. We will show a modified version of our technique for proving its stability. As before, we have $\forall x, y \in \mathcal{D}_{n,s}, g(\frac{x+y}{2}) \approx_\Delta \frac{g(x)+g(y)}{2}$. To prove the stability of this equation, we construct an affine linear h. Note that two points are necessary and sufficient to fully determine h. We set $h(\frac{n}{s}) = g(\frac{n}{s})$ and $h(0) = g(0)$.

Lemma 36 $e(\frac{x+y}{2}) \leq e(x)/2 + e(y)/2 + \Delta$.

Proof. $e(\frac{x+y}{2}) = |g(\frac{x+y}{2}) - h(\frac{x+y}{2})| \leq \Delta + |\frac{g(x)+g(y)}{2} - \frac{h(x)+h(y)}{2}|$. But, $|\frac{g(x)-h(x)}{2} + \frac{g(y)-h(y)}{2}| = e(x)/2 + e(y)/2$. □

The following corollary is immediate.

Corollary 37 *$e(\frac{x}{2}) \leq \Delta + e(x)/2$ and $e(\frac{x+\frac{n}{s}}{2}) \leq \Delta + e(x)/2$.*

Proof. Since for $y = 0$ and $y = \frac{n}{s}$, $e(y) = 0$ in Lemma 36. □

We construct a slightly different tree T_k in this case. The root of T_k is labeled by $\frac{n}{s}$ and if x is the label of a node, then $x/2$ (if integral and not already present) is label of its left child and $(\frac{n}{s} + x)/2$ (if integral and not already present) is the label of its right child.

Theorem 38 *The Jensen equation is $(\mathcal{D}^+_{n,s}, \mathcal{D}^+_{n,s}, \Delta, 2\Delta)$-stable.*

Proof. The proof is by induction on an enumeration order of T_k given by, say, a breadth-first traversal. Clearly, at the root, $e(\frac{n}{s}) = 0 \leq 2\Delta$. Now, if $e(x) \leq 2\Delta$, then, consider its children. Its left (resp. right) child (if exists) is $x/2$ (resp. $(x + \frac{n}{s})/2$). Thus, by Corollary 37, we have $e(\frac{x}{2}) \leq \Delta + e(x)/2 \leq 2\Delta$ (resp. $e(\frac{x+\frac{n}{s}}{2}) \leq \Delta + e(x)/2 \leq 2\Delta$). □

4.3 Approximate Robustness for Functional Equations

As in [20, 29], we test the program on $\mathcal{D}_{2p,s}$ and make conclusions about its correctness on $\mathcal{D}_{n,s}$. The relationship between p and n will be determined later. The domain has to be such that G is analytic in it. Therefore, we consider the case when f is bounded on $\mathcal{D}_{2p,s}$, i.e., $0 < L \leq f(x) \leq U$. Let $\mathcal{G}$ be the family of functions f that satisfy the following conditions:

1. $\Pr_{x \in \mathcal{D}_{2p,s}} [f(x) \geq L] \geq 1 - \epsilon$,
2. $\Pr_{x \in \mathcal{D}_{2p,s}} [f(x) \leq U] \geq 1 - \epsilon$,
3. $\Pr_{x,y \in \mathcal{D}_{2p,s}} [G[f(x), f(y)] \geq L] \geq 1 - \epsilon$, and
4. $\Pr_{x,y \in \mathcal{D}_{2p,s}} [G[f(x), f(y)] \leq U] \geq 1 - \epsilon$.

Note that the membership in $\mathcal{G}$ is easy to determine by sampling. We can define a distribution $\mathcal{E}_{\mathcal{T}(n,s)}$ such that if P satisfies the functional equation on $\mathcal{E}_{\mathcal{T}(n,s)}$ with probability at least $1-\epsilon$, then P also satisfies the following four properties.

1. $\Pr_{x,y \in \mathcal{D}_{p,s}} [P(x+y) \approx_\Delta G[P(x), P(y)]] \geq 1 - \epsilon$,
2. $\Pr_{x,y \in \mathcal{D}_{p,s}} [P(x) \approx_\Delta G[P(x-y), P(y)]] \geq 1 - \epsilon$,
3. $\Pr_{x,y \in \mathcal{D}_{p,s}} [P(x) \approx_\Delta G[P(y), P(x-y)]] \geq 1 - \epsilon$, and
4. $\Pr_{x \in \mathcal{D}_{n,s}, y \in \mathcal{D}_{p,s}} [P(x) \approx_\Delta G[P(x-y), P(y)]] \geq 1 - \epsilon$.

$\mathcal{E}_{\mathcal{T}(n,s)}$ is defined by flipping a fair four-sided die and on outcome $i \in \{1, 2, 3, 4\}$, choosing inputs according to the distribution given in the i-th property above. Let $\widehat{G} = \|G'_x\|_{\mathcal{D}}$. We can then show the following:

Theorem 39 *The addition theorem with the distribution $\mathcal{E}_{\tau(n,s)}$ is $(2\epsilon, \epsilon, \mathcal{D}_{2p,s}, \mathcal{D}_{n,s}, \Delta, (9\widehat{G}^2 + 5\widehat{G})\Delta, \Delta)$-approximately robust.*

Define for $x \in \mathcal{D}_{p,s}$, $g(x) = \operatorname{median}_{y \in \mathcal{D}_{p,s}}\{G[P(x-y), P(y)]\}$. By Markov's inequality, definition of g, and the properties of P, it is easy to show the following:

Lemma 40 $\Pr_{x \in \mathcal{D}_{n,s}}[g(x) \approx_\Delta P(x)] > 1 - 2\epsilon$.

Proof. Consider the set of elements $x \in \mathcal{D}_{n,s}$ such that $\Pr_{y \in \mathcal{D}_{p,s}}[P(x) \approx_\Delta G[P(x-y), P(y)]] < \frac{1}{2}$. If the fraction of such elements is more than 2ϵ, then it contradicts hypothesis (4) on P that $\Pr_{x \in \mathcal{D}_{n,s}, y \in \mathcal{D}_{p,s}}[P(x) \approx_\Delta G[P(x-y), P(y)]] \geq 1 - \epsilon$. For the rest, for at least half of the y's, $P(x) \approx_\Delta G[P(x-y), P(y)]$. By defining g to be the median (over y's in $\mathcal{D}_{p,s}$), we have for these elements $g(x) \approx_\Delta P(x)$. □

For simplicity of notation, let P_x denote $P(x)$ for any $x \in \mathcal{D}_{p,s}$ and $G_{x,y}$ denote $G[P(x), P(y)]$ for any $x, y \in \mathcal{D}_{p,s}$. Since G is fixed, we will drop G from the modulus of continuity.

A distribution U' on $\mathcal{D}$ is said to be *ϵ-uniform on $\mathcal{D}$* if $\sum_{x \in \mathcal{D}} |U'(x) - 1/|\mathcal{D}|| \leq \epsilon$. Let $\gamma = n/2p$.

Fact 41 *(1) For all $x \in \mathcal{D}_{2n,s}$, the distribution of $x + y$ is γ-uniform on $\mathcal{D}_{p,s}$.*

(2) For any event $E(x)$ and for an ϵ-uniform distribution U' on $\mathcal{D}$, $|\Pr_{x \sim U'}[E(x)] - \Pr_{x \in \mathcal{D}}[E(x)]| \leq \epsilon$.

Lemma 42 *For $x \in \mathcal{D}_{2n,s}$, $\Pr_{y,z \in \mathcal{D}_{p,s}}[G_{x-y,y} \approx_{2\omega(\Delta,0)} G_{x-z,z}] \geq 1 - 12\epsilon - 4\gamma$.*

Proof. $\Pr_{y,z \in \mathcal{D}_{p,s}}[G_{x-y,y} \approx_{\omega(\Delta,0)} G[G_{x-z,z-y}, P_y] = G[P_{x-z}, G_{z-y,y}] \approx_{\omega(0,\Delta)} G_{x-z,z}] > 1 - 12\epsilon - 4\gamma$. The error in the first step (due to computation of P_{x-y}) is $\omega(\Delta, 0)$ and the equation holds with probability at least $1 - \epsilon - \gamma$ by property (3) and Fact 41. The bounds on $G_{x-z,z-y}$ also hold with probability at least $1 - 2\epsilon - 2\gamma$ by properties (3), (4) and Fact 41 and so the error is just $\omega(\Delta, 0)$. The next line is just rewriting. In a similar manner, the final equation holds with probability at least $1 - \epsilon - \gamma$ by property (2) and Fact 41 and the error bound is $\omega(0, \Delta)$ The bounds on random points $P_y, P_z, P_{x-z}, P_{z-y}$ hold with probability at least $1 - 8\epsilon$ by properties (1), (2) on P to make the error $\omega(0, \Delta)$. Hence, the total error is

$\omega(\Delta,0)+\omega(0,\Delta)=2\omega(\Delta,0)$ by Fact 26(3) and the equality holds with probability at least $1-12\epsilon-4\gamma$. □

The following lemma, which helps us to bound the error, is from [23]. The proof uses the observation that the clique number of G^2 is at least as big as the maximum degree in G. Hence, for a random node x, probability that x is present in the largest clique in G^2 is more than the probability that x is connected to the maximum degree vertex (say y) in G.

Lemma 43 ([23]) *If $G=\langle V,E\rangle$ is a random graph with edges inserted with probability $1-\epsilon$, then $G^2=\langle V,\{(x,y):\exists z\in V,(x,z)\in E\wedge(z,y)\in E\}\rangle$ is a graph where the largest clique is of size at least $(1-\epsilon)|V|$.*

The following shows, in some sense, that g is well-defined:

Lemma 44 *For all $x\in\mathcal{D}_{2n,s}$, $\Pr_{y\in\mathcal{D}_{p,s}}[g(x)\approx_{2\Delta'}G_{x-y,y}]\geq 1-12\epsilon-4\gamma$, where $\Delta'=2\omega(\Delta,0)$.*

Proof. We have the following: for all $x\in\mathcal{D}_{2n,s}$, $\Pr_{y,z\in\mathcal{D}_{p,s}}[G_{x-y,y}\approx_{\Delta'}G_{x-z,z}]\geq 1-12\epsilon-4\gamma$. Now, we use Lemma 43. If G denotes a graph in which (y,z) is an edge iff $G_{x-y,y}\approx_{\Delta'}G_{x-z,z}$ then G^2 denotes the graph in which (y,z) is an edge iff $G_{x-y,y}\approx_{2\Delta'}G_{x-z,z}$. Now, using Lemma 43, we have that number of elements that are $2\Delta'$ away from the largest clique is at most 2ϵ. Thus, at least $1-2\epsilon$ of elements are within $2\Delta'$ of each other. If $\epsilon<1/2$ and since $g(x)$ is the median, the lemma follows. □

Now, the following theorem completes the proof that g satisfies the addition theorem approximately.

Theorem 45 *For all $x,y\in\mathcal{D}_{n,s}$, $g(x+y)\approx_{\Delta''}G[g(x),g(y)]$ with probability at least $1-56\epsilon-14\gamma$, where $\Delta''=(9\widehat{G}^2+5\widehat{G})\Delta$.*

Proof.
$$\begin{aligned}\Pr_{u,v\in\mathcal{D}_{p,s}}[G[g(x),g(y)]&\approx_{\omega(2\Delta',2\Delta')}G[G_{u,x-u},G_{v,y-v}]\\
&=\quad G[P_u,G[P_{x-u},G_{v,y-v}]]\\
&=\quad G[P_u,G[G_{x-u,v},P_{y-v}]]\\
&\approx_{\omega(0,\omega(\Delta,0))}\quad G[P_u,G_{x-u+v,y-v}]\\
&\approx_{\omega(0,\Delta)}\quad G_{u,x+y-u}\\
&\approx_{2\Delta'}\quad g(x+y)]\geq 1-56\epsilon-14\gamma\end{aligned}$$

By Lemma 44, the first equality holds with probability $1-24\epsilon-8\gamma$ and error $\omega(2\Delta', 2\Delta')$. By property (4), the bounds on $G_{u,x-u}, G_{v,y-v}$ hold with probability at least $1-4\epsilon$ to make the error $\omega(2\Delta', 2\Delta') \leq 2\omega(2\Delta', 0) = 4\omega(\Delta', 0) = 8\omega(\omega(\Delta, 0), 0)$ by Fact 26(3). The second and third equalities are always true. The fourth equality holds with probability at least $1-\epsilon-\gamma$ by property (1) and Fact 41 on P and the error accrued is $\omega(0, \omega(\Delta, 0))$. The bounds on $P_u, P_{x-u}, P_v, P_{y-v}, G_{x-u,v}$ hold with probability at least $1-10\epsilon$ by properties (1)-(4) to make the error $\omega(0, \omega(\Delta, 0)) = \omega(\omega(\Delta, 0), 0)$. The fifth equality also holds with probability at least $1-\epsilon-\gamma$ by property (1) on P and the error accrued is $\omega(0, \Delta) = \omega(\Delta, 0)$, after bounds on P_u, P_{x+y-u} (with probability at least $1-4\epsilon$). The final equality holds with probability at least $1-12\epsilon-4\gamma$ by Lemma 44 and error is $2\Delta' = 4\omega(\Delta, 0)$. Thus, the total error is $9\omega(\omega(\Delta, 0), 0) + 5\omega(\Delta, 0)$. But, $\omega(\Delta, 0) \leq \Delta\widehat{G}$ by Fact 26(4). Hence, $\omega(\omega(\Delta, 0), 0) \leq \Delta\widehat{G}^2$.

□

If $\epsilon < 1/112, p > 14n$, we have $1-56\epsilon-14\gamma > 0$ and so the above lemma is true with probability 1. In the case of our example function, we already calculated $\widehat{G} = U^2/(L+U)^2$. Hence, $\Delta'' = \Delta\left(\frac{9U^4}{(L+U)^4} + \frac{5U^2}{(L+U)^2}\right)$.

5 Approximate Self-Testing and Self-Correcting

In this section we briefly show how to apply our techniques that we developed in this paper to construct approximate self-tester and self-correctors. The approaches in this section follow [8, 20].

5.1 Definitions

The following modifications of definitions from [20] capture the idea of approximate checking, self-testing, and self-correcting in a formal manner. Let P be a program for f, $x \in \mathcal{D}_{n,s}$ an input to P, and β the confidence parameter.

Definition 46 *A* $(\Delta_1, \Delta_2, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate result checker *for f is a probabilistic oracle program T that, given P, $x \in \mathcal{D}_{n,s}$, and β, satisfies the following:*

(1) *P $(\Delta_1, 0)$-approximates f on $\mathcal{D}_{\tau(n,s)}$ $\Rightarrow$ $\Pr[T^P$ outputs "PASS"$] \geq 1-\beta$.*

(2) $P(x) \not\approx_{\Delta_2} f(x) \Rightarrow \Pr[T^P$ *outputs "FAIL"*$] \geq 1 - \beta$.

Definition 47 *A* $(\Delta_1, \Delta_2, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate self-tester *for f is a probabilistic oracle program T that, given P and β, satisfies the following:*

(1) *P $(\Delta_1, 0)$-approximates f on $\mathcal{D}_{\tau(n,s)}$* $\Rightarrow \Pr[T^P$ *outputs "PASS"*$] \geq 1 - \beta$.

(2) *P does not (Δ_2, ϵ)-approximate f on $\mathcal{D}_{n,s}$* $\Rightarrow \Pr[T^P$ *outputs "FAIL"*$] \geq 1 - \beta$.

Observe that if a property is $(\delta, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s}, \Delta_1, \Delta_2, \Delta_3)$-approximately robust, $(\mathcal{D}_{n,s}, \mathcal{D}_{n',s'}, \Delta_2, \Delta_4)$-stable, and it is possible to do equality testing for the function family satisfying the property, then it is possible to construct a $(\Delta_1, \Delta_3 + \Delta_4, \epsilon, \mathcal{D}_{n,s}, \mathcal{D}_{n',s'})$-approximate self-tester.

Definition 48 *A* $(\Delta, \epsilon, \Delta', \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate self-corrector *for f is a probabilistic oracle program SC_f^P that, given P that (Δ, ϵ)-approximates f on $\mathcal{D}_{\tau(n,s)}$, $x \in \mathcal{D}_{n,s}$, and β, outputs $\mathrm{SC}_f^P(x)$ such that* $\Pr[\mathrm{SC}_f^P(x) \approx_{\Delta'} f(x)] \geq 1 - \beta$.

Note that a $(\Delta_1, \Delta_2, \epsilon, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate self-tester and $(\Delta_2, \epsilon, \Delta_3, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate self-corrector yield a $(\Delta_1, \Delta_3, \mathcal{D}_{\tau(n,s)}, \mathcal{D}_{n,s})$-approximate result checker [8].

5.2 Constructing Approximate Self-Correctors

We illustrate how to build approximate self-correctors for functional equations. Suppose P (Δ, ϵ)-approximates f for $\epsilon < 1/8$ and $f(x+y) = G[f(x), f(y)]$. Then the self-corrector SC_f^P at input x is constructed as follows. To obtain a confidence of β:

1. choose random points $y_1, y_2, \ldots, y_N (N = O(\ln 1/\beta))$,
2. let $\mathrm{SC}_f^P(x)$ be the median of $G[P(x - y_1), P(y_1)], \ldots, G[P(x - y_N), P(y_N)]$.

By the assumption on ϵ, both the calls to P on $x - y_i$ and y_i are within Δ of f with probability greater than 3/4. In this case, the value of $G[P(x - y_i), P(y_i)]$ is $\Delta' = 2\Delta\widehat{G}$ away from $f(x)$ (see Section 4.1 for $\widehat{G}$). Using Chernoff bounds, we can see that at least half of the values $G[P(x - y_i), P(y_i)]$ are at most Δ' away from $f(x)$. Thus, their median $\mathrm{SC}_f^P(x)$ is also at most Δ' away from $f(x)$.

For degree d polynomials, a similar self-corrector works with $\Delta' = O((d+1)2^d\Delta)$. In order to pass good programs, this is almost the best Δ' possible using the evenly spaced interpolation equation since the coefficients of the interpolation equation are $\Omega(2^d)$. Using interpolation equations that do not use evenly spaced points seem to require Δ' that is dependent on the size of the domain.

5.3 Constructing Approximate Self-Testers

The following is a self-tester for any function satisfying an addition theorem $f(x+y) = G[f(x), f(y)]$ computing the function family $\mathcal{F}$ over $\mathcal{D}_{n,s}$. We use the notation from Section 4.1. To obtain a confidence of β, we choose random points $x_1, y_1, \ldots, x_N, y_N (N = O(1/\epsilon \ln 1/\beta))$ and verify the assumptions on program P in the beginning of Section 4.3. If P passes the test, then using Chernoff bounds, approximate robustness, and stability of the property, we are guaranteed that P approximates some function in $\mathcal{F}$. We next perform the equality test to ensure that P approximates the given $f \in \mathcal{F}$. Assume that $f(\frac{1}{s})$ when $c < 1$ (resp. $f(\frac{n}{s})$ when $c > 1$) is given. Using the proofs in Section 4.2, one can show that if there is a constant Δ such that $\mathrm{SC}_f^P(\frac{1}{s}) \approx_\Delta f(\frac{1}{s})$ when $c < 1$ ($\mathrm{SC}_f^P(\frac{n}{s}) \approx_\Delta f(\frac{n}{s})$ when $c > 1$), the error between SC_f^P and f can be bounded by a constant on the rest of $\mathcal{D}_{n,s}$. Since SC_f^P approximates P, the correctness of the self-tester follows.

For polynomials, we use random sampling to verify the conditions on program P required for approximate robustness that are given in the beginning of Section 3.3. If P satisfies the conditions then using the approximate robustness and stability of the evenly spaced interpolation equation, P is guaranteed to approximate some degree d polynomial h. To perform the equality test that determines if P approximates the correct polynomial f, we assume that the tester is given the correct value of the polynomial f at $\ell = (d+1)/\epsilon$ evenly spaced points $x_1 = -\frac{n}{s}, \ldots, x_\ell = \frac{n}{s} \in \mathcal{D}_{n,s}$. Using the self-corrector SC_f^P from Section 5.2, we have $\|\mathrm{SC}_f^P - h\| \le \Delta' = (d+1)2^d 2^{d \lg d}\Delta$. The equality tester now tests that for all x_i, $|f(x_i) - \mathrm{SC}_f^P(x_i)| \le (d+1)2^d\Delta$. Call an input x *bad* if $|f(x) - h(x)| > \Delta'' = \Delta' + (d+1)2^d\Delta$. If x is bad then $|f(x) - \mathrm{SC}_f^P(x)| > (d+1)2^d\Delta$. If x is a sample point, and x is bad, then the test would have failed. Define a *bad interval* to be a sequence of consecutive bad points.

If the test passes, then any bad interval in the domain can be of length at most $(2n+1)/\ell$, because any longer interval would contain at least one sample point. The two sample points immediately preceding and following the bad interval satisfy $|f(x)-h(x)| \leq \Delta''$. This implies that there must be a local maximum of $f-h$ (a degree d polynomial) inside the bad interval. Since there are only d extrema of $f-h$, there can be at most d bad intervals, and so the total number of bad points is at most $d(2n+1)/\ell$. Thus, on $1-\epsilon$ fraction of $\mathcal{D}_{n,s}$, SC_f^P's error is at most $\Delta' + \Delta''$. These arguments can be generalized to the k-variate case by partitioning the k-dimensional space into $((d+1)/\epsilon)^k$ cubes.

We have thus shown how to construct approximate self-testers and self-correctors. It is straightforward to construct approximate result-checkers using these.

5.4 Reductions Between Functional Equations

This section explores the idea of using reductions among functions (as in [7, 3]) to obtain approximate self-testers for new functions. Consider any pair of functions f_1, f_2 that are interreducible via functional equations. Suppose we have an approximate self-tester for f_1 and let there exist continuous computable functions F, F^{-1} such that $f_2(x) = F[f_1(x)]$ and $f_1(x) = F^{-1}(f_2(x))$. Given a program P_2 computing f_2, construct program P_1 computing f_1 via F^{-1}. We can then self-test P_1. Suppose P_1 is Δ-close to f_1 on a large portion of the domain. Then for every x for which $P_1(x)$ is Δ-close to $f_1(x)$, we bound the deviation of $P_2(x)$ from $f_2(x)$ by $\Delta' = F[f_1(x)+\Delta]-f_2(x)$. Then $\Delta' = F[f_1(x)+\Delta]-F[f_1(x)] \leq \omega(F;\Delta)$. If we can bound the right-hand side by a constant (at least for a portion of the domain), we can bound the maximum deviation Δ' of P_2 from f_2. This idea can be used to give simple and alternative approximate self-testers for functions like $\sin x, \cos x, \sinh x, \cosh x$ which can be reduced to e^x.

For example, suppose we are given a $(\delta_1, \epsilon_1, \delta_2, \epsilon_2, \mathcal{D}, \mathcal{D}')$- approximate self-tester for $f_1(x) = e^x$ and we want an approximate self-tester for the function f_2 given by $f_2(x) = \cos x$. By the Euler identity, $f_1(ix) = f_2(x) + if_2(x+3\pi/2)$. Given a program P_2 that supposedly computes f_2, we can build a program P_1 (for e^{ix}) out of the given P_2 (for $\cos x$) and self-test P_1. $P_1(ix) = P_2(x) + iP_2(x+3\pi/2)$.

Let the range of f_1 be equipped with the following metric: $|P_1(x) - f_1(x)| = |\Re(P_1(x) - f_1(x))| + |\Im(P_1(x) - f_1(x))|$. In other words, in our case, we have $|P_1(x) - e^{ix}| = |P_2(x) - \cos x| + |P_2(x + 3\pi/2) - \cos(x + 3\pi/2)|$. This metric ensures that P_1 is erroneous on x if and only if P_2 is erroneous on at least one of $x, x + 3\pi/2$. Alternatively, there is no "cancellation" of errors.

Suppose P_1 is (δ_1, ϵ_1)-good. Then, what can we say about P_2? For δ_1 fraction of the "bad" domain for P_1, the errors can occur in both the places where P_2 is invoked. Hence, at most $2\delta_1$ fraction of the domain for P_2 is bad. The rest of the domain for P_1 is ϵ_1-close to f_1, which by our metric implies P_2 is also ϵ_1-close to f_2. Thus, P_2 is $(2\delta_1, \epsilon_1)$-good.

Similarly, suppose P_1 is not (δ_2, ϵ_2)-good. P_1 is not good on at least δ_2 fraction of the domain, where P_1 is not ϵ_2-close to f_1. Thus, at these points in the domain, at least one of points where P_2 is called is definitely not $\epsilon_2/2$-close to f_2. Thus, P_2 is not $(\delta_1, \epsilon_2/2)$-good.

Therefore, we have an $(2\delta_1, \epsilon_1, \delta_2, \epsilon_2/2, \mathcal{D}, \mathcal{D}')$- approximate self-tester for f_2 from a $(\delta_1, \epsilon_1, \delta_2, \epsilon_2, \mathcal{D}, \mathcal{D}'$ approximate self-tester for f_1, given by [20].

Acknowledgements. We would like to thank Janos Aczel (U. of Waterloo), Peter Borwein (Simon Fraser U.), Gian Luigi Forti (U. of Milan), D. Sivakumar (SUNY, Buffalo), Madhu Sudan (IBM, Yorktown), and Nick Trefethen (Cornell U.), for their suggestions and pointers. We would also like to thank the two anonymous referees for improving the presentation of this paper.

References

[1] J. Aczel. *Lectures on Functional Equations and their Applications.* Academic Press, 1966.

[2] M. Albert and J.A. Baker. Functions with bounded n-th differences. *Ann. Polonici Mathematici*, 43:93–103, 1983.

[3] S. Ar, M. Blum, B. Codenotti, and P. Gemmell. Checking approximate computations over the reals. *Proc. 25th ACM Symposium on Theory of Computing*, pp. 786–795, 1993.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *JACM*, 45(3):501–555, 1998.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *JACM*, 45(1):70–122, 1998.

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

[7] M. Blum and S. Kannan. Designing programs that check their work. *JACM*, 42(1):269–291, 1995.

[8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Sys. Sci.*, 47(3):549–595, 1993.

[9] M. Blum and H. Wasserman. Software reliability via run-time result-checking. *JACM*, 44(6):826–849, 1997.

[10] M. Blum and H. Wasserman. Reflections on the Pentium division bug. *IEEE Trans. on Computers*, 45(4):385–393, 1996. 1994.

[11] E. Castillo and M.R. Ruiz-Cobo. *Functional Equations and Modeling in Science and Engineering.* Marcel Dekker Inc., 1992.

[12] P.W. Cholewa. The stability problem for a generalized Cauchy type functional equation. *Rev. Roumaine Math. Pures Appl.*, 29:457–460, 1984.

[13] W.J. Cody. Performance evaluation of programs related to the real gamma function. *ACM Trans. on Mathematical Software*, 17(1):46–54, 1991.

[14] W.J. Cody and I. Stoltz. The use of Taylor series to test accuracy of function programs. *ACM Trans. on Mathematical Software*, 17(1):55–63, 1991.

[15] D.Z. Djokovic. A representation theorem for $(X_1 - 1)(X_2 - 1)\dots(X_n - 1)$ and its applications. *Ann. Polonici Mathematici*, 22:189–198, 1969.

[16] G.L. Forti. An existence and stability theorem for a class of functional equations. *Stochastica*, 4:23–30, 1980.

[17] G.L. Forti. Hyers–Ulam stability of functional equations in several variables. *Aeq. Mathematicae*, 50:143–190, 1995.

[18] G.L. Forti and J. Schwaiger. Stability of homomorphisms and completeness. *C.R. Math. Rep. Acad. Sci. Canada*, 11:215–220, 1989.

[19] Z. Gajda. Local stability of the functional equation characterizing polynomial functions. *Ann. Polonici Mathemtici*, 52:119–137, 1990.

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

[21] D.H. Hyers. On the stability of the linear functional equation. *Proc. Nat. Acad. Sci. U.S*, 27:222–224, 1941.

[22] M. Kiwi, F. Magniez, and M. Santha. Approximate testing with relative error. *Proc. 31st ACM Symposium on Theory of Computing*, pp. 51–60, 1999.

[23] S.R. Kumar and D. Sivakumar. Manuscript. 1995.

[24] R. Lipton. New directions in testing. *Proc. DIMACS Workshop on Distr. Comp. and Cryptography*, pp. 191–202, 1991.

[25] G. G. Lorentz. *Approximation of Functions.* Holt, Rinehart and Winston, 1966.

[26] C. Lund. *The Power of Interaction.* Ph.D. Thesis, U. of Chicago, 1991.

[27] S. Mazur and W. Orlicz. Grundlegende Eigenschaften der Polynomischen Operationen. *Erste Mitteilung, Studia Math.*, 5:50–68, 1934.

[28] R. Rubinfeld. Robust functional equations and their applications to program testing. *SIAM J. on Computing*, 28(6):1972–1997, 1999.

[29] R. Rubinfeld and M. Sudan. Testing polynomial functions efficiently and over rational domains. *Proc. 3rd SIAM Symposium on Discrete Algorithms*, pp. 23–43, 1992.

[30] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials and their applications to program testing. *SIAM J. on Computing*, 25(2):252–271, 1996.

[31] F. Skof. Sull'approssimazione delle applicazioni localmente δ-additive. *Atti della Accademia delle Scienze di Torino*, 117:377–389, 1983.

[32] M. Sudan. *Personal communication*, 1991.

[33] A.F. Timan. *Theory of Approximation of Functions of a Real Variable.* Pergamon Press, 1963.

[34] E.C. Titchmarsh. *The Theory of Functions.* Oxford U. Press, 1947.

[35] F. Vainstein. *Algebraic Methods in Hardware/Software Testing.* PhD thesis, Boston U., 1993.

A Proofs of Some Theorems for Linearity

Theorem 49 (Hyers' Theorem) *Let S be an Abelian semigroup and B be a Banach space and let $g : S \to B$ be such that for some $\Delta > 0$, g is Δ-approximately linear on S, then, for every $x \in S, h(x) = \lim_{n\to\infty} g(2^n x)/2^n$ exists, h is linear, and $\|g - h\| \leq \Delta$.*

Proof. ([17]) By induction on n, $|g(x)/2^n - g(x/2^n)| < \Delta(1 - 1/2^n)$. Let $q_n(x) = g(2^n x)/2^n$. Then, $q_n(x) - q_m(x) = (g(2^{m-n}2^n x) - 2^{m-n}g(2^n x))/2^m$. If $m < n$, we can obtain $|q_n(x) - q_m(x)| < \Delta(1 - 2^{m-n})/2^m$. Thus, for $x \in S$, $\{q_n(x)\}$ is a Cauchy sequence and by completeness of B, it has a limit function $h(x) = \lim_{n\to\infty} g(2^n x)/2^n$. The properties of h are easily proved. □

Theorem 50 (Skof's Theorem) *Let $n > 0$ and let $g : [0, n) \to \mathbb{R}$ be such that for some $\Delta \geq 0$, $|g(x + y) - g(x) - g(y)| \leq \Delta$ for all $0 \leq x, y < n$ (such that $x + y < n$), then, there exists a linear $h : \mathbb{R} \to \mathbb{R}$ such that $\|g - h\|_{[0,n)} \leq 3\Delta$.*

Proof. For any $x \in \mathbb{R}^+$, write $x = p(n/2) + q$ where $0 \leq q < n/2$. Define $g' : \mathbb{R}^+ \to \mathbb{R}$ such that $g'(x) = pg(n/2) + g(q)$. Clearly, $\|g' - g\|_{[0,n)} \leq \Delta$. Now, the claim is $g'(x+y) \approx_{2\Delta} g'(x) + g'(y)$. As before, $x = p(n/2) + q, y = r(n/2) + s$ with $0 \leq q, s < n/2$.

$0 \leq q + s < n/2$. We have, $g'(x+y) = g(q+s) + (p+r)g(n/2) \approx_{\Delta} g(q) + g(s) + pg(n/2) + rg(n/2) = g'(x) + g'(y)$.

$n/2 \leq q + s < n$. Let $q+s = t + n/2$. We have $g'(x+y) = g(t) + (p+r)g(n/2) + g(n/2) \approx_{2\Delta} g(q) + g(s) + pg(n/2) + rg(n/2) = g'(x) + g'(y)$.

To extend g' to $\mathbb{R}$, define for $x < 0$, $g'(x) = -g'(-x)$. Thus, $\forall x, y \in \mathbb{R}, g'(x+y) \approx_{2\Delta} g'(x) + g'(y)$. By Theorem 49, there is a linear h such that $\|g' - h\| \leq 2\Delta$. Therefore, $\|g - h\|_{[0,n)} \leq \|g - g'\|_{[0,n)} + \|g' - h\|_{[0,n)} \leq 3\Delta$. □

B Proofs of Some Theorems for Polynomials

B.1 Stability for Polynomials

Fact 12 $(\nabla_{t_1+t_2} - \nabla_{t_1} - \nabla_{t_2})f(x) = \nabla_{t_1,t_2} = \nabla_{t_2,t_1}$.

Proof. $(\nabla_{t_1+t_2} - \nabla_{t_1} - \nabla_{t_2})f(x) = f(x+t_1+t_2) - f(x) - f(x+t_1) + f(x) - f(x+t_2) + f(x) = f(x+t_1+t_2) - f(x+t_1) - f(x+t_2) + f(x) = \nabla_{t_1} f(x+t_2) - \nabla_{t_1} f(x) = \nabla_{t_1}(f(x+t_2) - f(x)) = \nabla_{t_1,t_2} f(x) = \nabla_{t_2,t_1} f(x)$. □

Difference operators act on multilinear functions in a nice manner, which is captured in the following fact.

Fact 51 *If f is a k-linear function, then $\nabla_{t_1,\ldots,t_d} f^*(x) = k! f(t_1, \ldots, t_k)$ if $k = d$ and 0 if $k < d$.*

Proof. Recall that, due to multilinearity, f is also symmetric. By induction on k. Chasing definitions, we have $\nabla_{t_1,\ldots,t_d} f^*(x) = \nabla_{t_1,\ldots,t_{d-1}}(f^*(x+t_d) - f^*(x)) = \nabla_{t_1,\ldots,t_{d-1}}(f((x+t_d)^{[d-1]}, x) + f((x+t_d)^{[d-1]}, t_d) - f(x^{[d]}))$, which by linearity of ∇ yields $\nabla_{t_1,\ldots,t_{d-1}} f((x+t_d)^{[d-1]}, t_d) +$

$\nabla_{t_1,\ldots,t_{d-1}}(f((x+t_d)^{[d-1]},x) - f(x^{[d]}))$. Observe that for any constant t, the restriction of k-linear f to any of its arguments being t (denoted f_t) results in a $(k-1)$-linear function. By induction, the first term in the above expression evaluates to $(d-1)!f_{t_d}(t_1,\ldots,t_{d-1}) = (d-1)!f(t_1,\ldots,t_d)$. Now, using the symmetry and linearity (in each variable) of f, we can write the second term as $\nabla_{t_1,\ldots,t_{d-1}}(\sum_{i=0}^{d-1}\binom{d-1}{i}f(x^{[i+1]},{t_d}^{[d-i-1]}) - f(x^{[d]}))$ which is $(d-1)\nabla_{t_1,\ldots,t_{d-1}}f(x^{[d-1]},t_d) + \sum_{i=0}^{d-3}\nabla_{t_1,\ldots,t_{d-1}}\binom{d-1}{i}f(x^{[i+1]},{t_d}^{[d-i-1]})$. By induction, the first term evaluates to $(d-1)(d-1)!f_{t_d}(t_1,\ldots,t_{d-1}) = (d-1)(d-1)!f(t_1,\ldots,t_d)$, which combined with the earlier result yields $d!f(t_1,\ldots,t_d)$. The second term evaluates to 0 since each of the terms inside the sum are restrictions of f to more than 1 variable, which evaluates to 0 after applying $\nabla_{t_1,\ldots,t_{d-1}}$. □

Fact 13 *Let $\mathcal{D}$ be a ring. The following characterizations of polynomials, are equivalent:*

1. $\forall x \in \mathcal{D}, f(x) = \sum_{k=0}^{d} a_k x^k$,
2. $\forall x,t \in \mathcal{D}, \nabla_t^{d+1}f(x) = 0$
3. *there exists symmetric k-linear functions F_k, $0 \le k \le d$ such that* $\forall x \in \mathcal{D}, f(x) = \sum_{k=0}^{d} F_k^*(x)$.

Proof. (1) $\Leftrightarrow$ (2) follows from Lagrangian interpolation. We first prove (1) $\Rightarrow$ (3). Given (1), just set $F_k(x_1,\ldots,x_k) = a_k\prod_{i=0}^{k}x_i$. It is easy to see that F_k's are symmetric, k-linear. We now prove (3) $\Rightarrow$ (2). Given (3), $\nabla_{t_1,\ldots,t_{d+1}}f(x) = \nabla_{t_1,\ldots,t_{d+1}}\sum_{k=0}^{d}F_k^*(x) = \sum_{k=0}^{d}\nabla_{t_1,\ldots,t_{d+1}}F_k^*(x) = 0$ by Fact 16 about difference operators. □

Fact 18 *For any $\lambda_1,\ldots,\lambda_d \in \{0,1\}$, if*

$$t'_{\lambda_1,\ldots,\lambda_d} = -\sum_{i=1}^{d}\lambda_i t_i/i, \qquad t''_{\lambda_1,\ldots,\lambda_d} = \sum_{i=1}^{d}\lambda_i t_i$$

then

$$\nabla_{t_1,\ldots,t_d}f(x) = \sum_{\lambda_1,\ldots,\lambda_d\in\{0,1\}}(-1)^{\lambda_1+\cdots+\lambda_d}\nabla^d_{t'_{\lambda_1,\ldots,\lambda_d}}f(x+t''_{\lambda_1,\ldots,\lambda_d}).$$

λ	t'_λ	t''_λ	term
00	0	0	0
01	$-t_2/2$	t_2	$-[f(x) - 2f(x + t_2/2) + f(x + t_2)]$
10	$-t_1$	t_1	$-[f(x - t_1) - 2f(x) + f(x + t_1)]$
11	$-t_1 - t_2/2$	$t_1 + t_2$	$+[f(x - t_1) - 2f(x + t_2/2) + f(x + t_1 + t_2)]$

Table 2: An Illustration of Fact 18

Proof. By a pairing argument. First, it is easy to prove that the left-hand side can be expressed as $\nabla_{h_1,\ldots,h_d} = \sum_{\lambda_1,\ldots,\lambda_d \in \{0,1\}} (-1)^{d+\lambda_1+\cdots+\lambda_d} f(x + t''_{\lambda_1,\ldots,\lambda_d})$. Now, we can expand the right-hand side as
$\sum_{\lambda_1,\ldots,\lambda_d \in \{0,1\}} (-1)^{\lambda_1+\cdots+\lambda_d} \sum_{k=0}^{d} (-1)^{d-k} \binom{d}{k} f(x + t''_{\lambda_1,\ldots,\lambda_d} + kt'_{\lambda_1,\ldots,\lambda_d})$. When $k = 0$, left-hand side is obtained. So, we have to prove that for $k > 0$, the right-hand side vanishes. The terms inside $f(\cdot)$ are linear combinations of t_i's by our construction. Note that for each $k > 0$, each term inside $f(\cdot)$ on the right-hand side has exactly one t_i absent because of its cancellation between t' and t''. So, for each $\lambda_1, \ldots, \lambda_d \in \{0,1\}$, construct its conjugate $\lambda'_1, \ldots, \lambda'_d \in \{0,1\}$ with $\lambda'_i = 1 - \lambda_i$ and $\lambda'_j = \lambda_j$ otherwise. It is easy to see that the terms $(-1)^{\lambda_1+\cdots+\lambda_d} f(x + t''_{\lambda_1,\ldots,\lambda_d} + kt'_{\lambda_1,\ldots,\lambda_d})$ and $(-1)^{\lambda'_1+\cdots+\lambda'_d} f(x + t''_{\lambda'_1,\ldots,\lambda'_d} + kt'_{\lambda'_1,\ldots,\lambda'_d})$ cancel. An illustration of this fact is given below. □

To illustrate with an example, consider the case when $d = 2$. Then, the left-hand side is given by $\nabla_{t_1,t_2} f(x) = f(x + t_1 + t_2) - f(x + t_1) - f(x + t_2) + f(x)$. The right-hand side is given by the sum of the entries in the last column of Table B.1.

It is easy to see that appropriate cancellations take place so that left-hand side equals the right-hand side.