On the robustness of functional equations *

Ronitt Rubinfeld †

Abstract

In this paper, we study the general question of how characteristics of functional equations influence whether or not they are robust. We isolate examples of properties which are necessary for the functional equations to be robust. On the other hand, we show other properties which are sufficient for robustness. We then study a general class of functional equations, which are of the form $\forall x, y \;\; F[f(x-y), f(x+y), f(x), f(y)] = 0$, where F is an algebraic function. We give conditions on such functional equations that imply robustness.

Our results have applications to the area of self-testing/correcting programs. We show that self-testers and self-correctors can be found for many functions satisfying robust functional equations, including algebraic functions of trigonometric functions such as $\tan x$, $\frac{1}{1+\cot x}$, $\frac{Ax}{1-Ax}$, $\cosh x$.

1 Introduction

The mathematical field of functional equations is concerned with the following prototypical problem: Given a set of properties (functional equations) over a particular domain, completely characterize the set of functions that satisfy them. For example, The *linearity property* over the integers is $\forall x, y \in \mathcal{Z} \;\; f(x+y) - f(x) - f(y) = 0$. The functions mapping from $\mathcal{Z}$ to $\mathcal{Z}$ that satisfy the linearity property, referred to as the *solution set* of the functional equation, is $\mathcal{F} = \{f | f(x) = c \cdot x, c \in \mathcal{Z}\}$. The linearity property is one of the famous, well-studied functional equations referred to as Cauchy's equations, and has been studied over many other domains and ranges with various properties (see the text by Aczél [3]). Functional equations are used widely in the study of the various functions that arise in areas such as mathematics, physics and economics. Several general classes of functional equations have been identified. For example, algebraic addition theorems, of the form

$$\forall x, y \;\; F[f(x+y), f(x), f(y)] = 0$$

where F is any algebraic function, were used as a starting point in the development of the theory of elliptic curves by Weierstrass. Other types of functional equations include difference

*A preliminary version of this work has appeared in *Proc. 35th IEEE Conference on Foundations of Computer Science*, pp. 288-299, 1994.

†Cornell University. email: ronitt@cs.cornell.edu. This work is supported by ONR Young Investigator Award N00014- 93-1-0590 and United States - Israel Binational Science Foundation Grant 92-00226.

equations, iteration equations, multivariate functional equations and systems of functional equations.

In Section 2, we present the definition of functional equations given in [3]. For the purposes of this introduction, we define functional equations as follows: Let $\mathcal{D}, \mathcal{R}$ be an arbitrary domain and range. Let $\mathcal{T}$ be a range containing 0, and $F : \mathcal{R}^k \times \mathcal{D}^k \to \mathcal{T}$ be a function that is computable via applying a finite number of known functions (in this paper we use $-, +, \times, \backslash$, hyperbolic functions, trigonometric functions and c^{th} roots for constant c). Let a *neighborhood* over the domain $\mathcal{D}$ be an ordered k-tuple in $\mathcal{D}^k$ and let $\mathcal{N} \subseteq \mathcal{D}^k$. The general form of a functional equation is then $\forall (x_1, \ldots, x_k) \in \mathcal{N}, \;\; F[f(x_1), \ldots, f(x_k), x_1, \ldots, x_k] = 0$. We denote the functional equation by $(F, \mathcal{N})$ when $\mathcal{D}, \mathcal{R}, \mathcal{T}$ are understood from the context. A particular solution of a functional equation is a function $f : \mathcal{D} \to \mathcal{R}$ for which F evaluates to 0 on all choices of neighborhoods in $\mathcal{N}$. The general solution, $\mathcal{F}$, is the family of functions that are solutions to the functional equation. Figures 1 and 2 give several examples of functional equations and their solution sets over the reals [3, 25]. In Section 2, we describe the formal definition of *characterizations* as given by Rubinfeld and Sudan in [38], which can be viewed as a generalization of functional equations.

All functional equations involve a "for all" quantifier. Here we are interested in comparing the solution to the functional equation when the "for all" quantifier is replaced by a "for most" quantifier. To illustrate, we give a simplified definition of *robustness*. For a given δ, define $\mathcal{G} \stackrel{\text{def}}{=} \{f | Pr_{(x_1,\ldots,x_k)\in\mathcal{N}}[F[f(x_1), \ldots, f(x_k), x_1, \ldots, x_k] = 0] \geq 1 - \delta\}$. Clearly $\mathcal{G}$ contains $\mathcal{F}$. However, is it the case that each function in $\mathcal{G} \backslash \mathcal{F}$ is essentially the same (equal on most inputs in $\mathcal{D}$) as some function in $\mathcal{F}$? Slightly more precisely, we say that two functions are ϵ-*close* over $\mathcal{D}$ if $\frac{|\{x \in \mathcal{D} | f(x) \neq g(x)\}|}{|\mathcal{D}|} \leq \epsilon$. For some small constant ϵ, if each function in $\mathcal{G}$ is ϵ-close to some function in $\mathcal{F}$, then in some sense, the "for most" quantifier is sufficient to characterize the same class of functions as the "for all" quantifier, and we say that the functional equation is (ϵ, δ)-*robust*. A formal and more general definition due to [38] is given in Section 2. Often it is the case that $\mathcal{N}$ and F are defined and are known to be (ϵ, δ)-robust over an infinite *set* $\mathcal{S}$ of domains and corresponding neighborhood sets. For example, the linearity property can be defined for all domains that are groups where for each group G, the corresponding neighborhood set is $\{x, y, x +_G y | x \in G\}$, ($+_G$ is the group operation for G) and the linearity properity is $\forall x, y, x +_G y \;\; f(x +_G y) -_G f(x) -_G f(y) = 0$. The linearity property is known to be $(2\delta, \delta)$-robust when the domain and range are any finite group for any $\delta < 2/9$ [26]. We are interested in the case when for all $\epsilon < 1$, there is a constant δ such that $(F, \mathcal{N})$ is (ϵ, δ)-robust over each of the domains in $\mathcal{S}$. In this case, we say that $(F, \mathcal{N})$ is robust over $\mathcal{S}$ (note that robustness is only interesting if $1/\delta$ is much smaller than $|\mathcal{N}|$).

Previous results on robust characterizations Robustness and related notions are used implicitly in a number of works [19] [9] [29] [10] [7] [6]. In the following sections, we describe the applications of robustness to program testing and to the study of probabilistically checkable proof systems.

There are many characterizations that are known to be robust: The first nontrivial characterization shown to be robust for constant ϵ, δ was the linearity property over finite groups in the work of Blum, Luby and Rubinfeld [19]. Coppersmith [26] gives a particularly elegant proof of the robustness of the linearity property as well as improves the allowable

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

Figure 1: Examples of functions satisfying addition theorems over the reals

Equation	Solution
$f(x+y)+f(x-y)=2f(x)$	$f(x)=Ax+a$
$f(x+y)+f(x-y)=2f(x)f(y)$	$f(x)=0, \cos Ax, \cosh Ax$
$f(x+y)+f(x-y)=2[f(x)+f(y)]$	$f(x)=Ax^2$
$f(x+y)-f(x-y)=2f(y)$	$f(x)=Ax$
$f(x+y)f(x-y)=f(x)^2$	$f(x)=a$
$f(x+y)-f(x-y)=4\sqrt{f(x)f(y)}$	$f(x)=Ax^2$
$f(x+y)f(x-y)=f(x)^2-f(y)^2$	$f(x)=Ax,\ k\sin Ax\ , k\sinh Ax$

Figure 2: Examples of functions satisfying $F[f(x-y), f(x+y), f(x), f(y)]=0$ over the reals

parameters of ϵ, δ to the following: If $f(x+y)-f(x)-f(y)=0$ is satisfied for a constant greater than $7/9$ fraction of the choices of x, y in the group G, then there is some function $g(z)=c\cdot z$ such that $f(x)=g(x)$ for at least $5/9$ of the x in G. Coppersmith also gives an example which shows that $\delta=7/9$ is a type of a threshold, i.e., there is a function which satisfies $f(x+y)-f(x)-f(y)=0$ for $7/9$ fraction of the choices of x, y in the group G, but which does not agree with any linear function on more than $1/3$ of the domain. Bellare, Coppersmith, Hastad, Kiwi and Sudan [11] show that one can get a tighter result on the range of δ that is useful over domains of the type $GF(2)^n$ where Coppersmith's example does not apply. Robust characterizations of total degree d polynomials are given in [38][1]. Robust characterizations of maximum degree d polynomials are given in several works [9][29][10][38][2]. These results apply to polynomials over finite fields $\mathcal{Z}_p$ (p prime), and finite subsets of rational domains. The first formal definition of robustness was given in [38]. Very recently, robust characterizations of functions satisfying linear recurrence relations have been given by Kumar and Sivakumar [33].

Our Results Our goal is to characterize the fundamental characteristics of functional equations that make them robust, in order to gain an understanding of how broadly robustness applies. It happens that the structure of the neighborhoods in $\mathcal{N}$ is very important to whether a characterization is robust. We present a graph theoretic characterization of neigh-

[1]The total degree of a polynomial is the maximum over all terms of the total degree of a term. The total degree of a term is the sum of the individual degrees of each variable in the term.

[2]The maximum degree of a polynomial is the maximum over all variables of the maximum degree of the variable in any term.

borhood sets $\mathcal{N}$, which is used to quantify the connectivity of $\mathcal{N}$. In Theorem 9, we show that high connectivity of $\mathcal{N}$ is necessary for $\mathcal{N}$ to be robust. Since the functional equations which relate inputs that are linear functions of a single variable (e.g., $\forall x,\ f(x) - f(x+1) - 1 = 0$) are known not to have this connectivity property, we can conclude that they are not robust. On the other hand, in Theorem 11, we show that when $\mathcal{N} = \mathcal{D}^k$, and the set of solutions to $(\mathcal{F}, \mathcal{N})$ is rich enough, $(\mathcal{F}, \mathcal{N})$ is robust.

We next investigate conditions on the class of functional equations of the general form $\forall x, y \quad F[f(x-y), f(x+y), f(x), f(y)] = 0$ that imply robustness. We focus on domains that are finite groups and certain types of subsets of infinite groups, such as those of the form $\mathcal{D}_{n,s} = \{\frac{i}{s} | \ |i| \leq n\}$ (see beginning of Subsection 2.1) and others that are of use in studying periodic functions. In the case of domains that are finite groups and domains used for studying periodic functions, testing that a function satisfies a functional equation over a domain will involve neighborhood sets that are chosen from the same domain. In the case of domains that are subsets of infinite groups of the form $\mathcal{D}_{n,s} = \{\frac{i}{s} | \ |i| \leq n\}$, testing that a function satisfies a functional equation will involve neighborhood sets that are chosen from a larger subset of the same infinite group. Our results apply to ranges that have a group structure. In Theorems 13 and 29, we show that if the equation can be written as $\forall x, y \quad f(x+y) = G[f(x), f(y)]$ (a special case of algebraic addition theorems referred to as an *addition theorem*), then it is robust as long as G satisfies $G[a, G[b, c]] = G[G[a, b], c]$ $\forall a, b, c$ (which is satisfied by all of our examples of addition theorems). The proofs for all types of domains rely on the same techniques. Since the proofs of the results over finite group domains are simpler to state, we give them first in order to highlight the main ideas. This work leads to self-testers for several families of trigonometric functions including $\tan Ax$, $\frac{1}{1+\cot x}$, $\frac{Ax}{1-Ax}$, $\cosh Ax$, and several examples from [3][4][22] given in Figure 1. A general format for constructing self-testers is given in Sections 5,6, and a self-tester for the particular example of the cosh function is given in Section 6.3. We then give techniques that apply to functions which satisfy other functional equations (the first three examples in Figure 2), including d'Alembert's equation $\forall x, y \quad f(x+y) + f(x-y) = 2f(x)f(y)$ in Section 4.2. In this case, the range must be a field containing 2.

Robustness and self-testing/correcting. In order to allow a programmer to use programs that are not known to be correct on all inputs, result checkers were introduced by Blum and Kannan [18], and soon after, the related paradigms of self-testers and self-correctors were introduced by Blum, Luby and Rubinfeld [19]. (A notion similar to self-correctors was independently proposed by Lipton [34].) The paradigm of self-testers and self-correctors is intended to fit into the framework of result checkers, and in fact it is observed that a self-tester and a self-corrector for a function can be combined to give a checker [19]. If a function has a checker, then one can determine whether program P is giving the correct answer on a particular input or whether there is a bug in the program. If a function has a self-corrector, then given a program P for computing the function that is correct on most inputs, one can transform P into a new randomized program that is correct on each input with high probability and is almost as efficient as running P. Self-testers allow one to ascertain that P is correct on a large enough fraction of the inputs so that it is capable of being self-corrected. More formal definitions of self-testers and self-correctors are given in Section 5. If a function

has both a self-tester and a corresponding self-corrector, then an unreliable program can be used to reliably compute the function.

Problems that can be viewed as linear or low degree polynomial functions, such as matrix multiplication, integer division, sine/cosine, integer multiplication, the mod function, modular multiplication, polynomial multiplication, modular exponentiation, Fast Fourier Transform and determinant, have been shown to have self-testers and self-correctors [19][8][34][23] [31][37][38][2] [28] [21]. Although many functions can be viewed as linear functions or low degree polynomials over an appropriate group structure, one concern was that these might be the only examples of functions that have self-testers and self-correctors. Using the new robustness results, we show that self-testers and self-correctors can be found for numerical functions that previously did not have self-testers and self-correctors. The techniques used to derive our results seem amenable to further generalization, and may apply to an even wider variety of numerical functions.

We concentrate on self-testers which operate by finding properties (such as functional equations) that should be satisfied by any correct program and then *testing that the program satisfies the properties for randomly chosen inputs.* In this work, we study the characteristics of the properties that make them usable for testing. Properties that can be tested more efficiently than computing the function f are particularly interesting for constructing good tests for programs.

The idea of testing programs by verifying that programs satisfy properties known to be satisfied by the functions being computed is not new to the self-testing/correcting approach. For example, matrix multiplication routines have been tested by verifying that the outputs satisfy the distributive property [39]. The work of Cody and Stolz [25] proposes the use of Taylor series in order to test programs for exponential integrals. These techniques apply to Bessel functions and Dawson's integral. The work of Vainstein [42, 43, 44, 45], suggests the use of *polynomial checks* for testing and correcting programs. In the language previously defined, polynomial checks are those functional equations for which the function F is a polynomial and the neighborhoods are ordered sets of the form $(x, x+a_1, x+a_2, \ldots, x+a_k)$ for fixed constants $a_1, \ldots, a_k$. These functional equations can be used to test functions that are algebraic functions of trigonometric functions. The work of Cody [24] suggests the following test for programs computing the real gamma function over the reals:

$$\text{Pick random } x \text{ and verify that } P(2x) = (2\pi)^{-1/2} 2^{2x-1/2} P(x) P(x+1/2).$$

In all of the above cases, it is clear that any correct program for the function must pass these recommended tests. However, none of the works mentioned in this paragraph give any formal evidence that programs that pass these tests should be usable. On the contrary, it is easy to come up with examples of programs that pass the above tests but do not compute the correct function on a large fraction of inputs.

Still, it has been shown that in many cases, using properties to test programs is mathematically justified (cf. [19] [37] [31] [38]). Essentially one can show that some of these tests can be used in conjunction with other simple tests in order to determine that a program is correct on most inputs. In order to show that such tests work, the main technique used has been to partition the problem into three tasks: First, find properties that characterize a family of functions, $\mathcal{F}$, containing the function f. For example, one can find functional

equations satisfied by specific classes of finite degree rational functions of $x, e^x, \sin x$ using the results of [43] [16]. Second, show that these properties are robust, so that it is possible to efficiently test whether the program is computing a function that is close to some function in $\mathcal{F}$. We call this task *property testing*. Third, find other efficient tests which allow the user to determine whether or not the program is computing the correct function within $\mathcal{F}$. We call this latter task *equality testing*. Equality testing can often be done much more efficiently once it is known that the program is essentially computing some member of $\mathcal{F}$. For example, the function $f(x) = x \bmod R$ is uniquely specified by the properties that (1) f is linear, i.e., $\forall x, y \;\; f(x) + f(y) \equiv f(x+y) \bmod R$, (2) f has slope 1, i.e., $\forall x \;\; f(x) + 1 \equiv f(x+1) \bmod R$. Using the robustness of linearity, if (1) is satisfied for *most* x, y (greater than a 7/9 fraction), then there is some function $g(x) = cx \bmod R$ such that $f(x) = g(x)$ for most x. If in addition (2) is satisfied for most x then $f(x) = x \bmod R$ for most x. (Note that if R is considered to be part of the input, then it is not enough to only test that property (2) is satisfied for any constant fraction of the $x \in [0..R-1]$.) Thus, it is only necessary to check that the program satisfies the given properties at a relatively small number (in this case, a constant independent of $|x|, |R|$) of randomly selected inputs in order to guarantee that the program usually computes the correct values. This paper concentrates on the task of property testing.

It is shown in [19] that self-correctors exist for any function that is random self-reducible,[3] since if the program is known to be correct on most inputs, then the correct value of the function at any particular input x can be inferred, even though the program may be incorrect on input x. In particular, any function satisfying the linearity property is random self-reducible [19]. On a related note, the use of polynomial checks (or the functional equations that are defined by the polynomial checks) for the correction of programs with few errors is suggested in [43], and Blum, Codenotti, Gemmell and Shahoumian [16] build on the work of [43] to give self-correctors for the same functions. Here we observe that efficient self-correctors exist for functions satisfying any one of a *class* of functional equations, namely those of the general form $\forall x, y \;\; F[f(x-y), f(x+y), f(x), f(y)] = 0$, where F is an algebraic function that has the property that given three of $f(x-y), f(x+y), f(x), f(y)$, F can be used to efficiently solve for the remaining one. A similar result was obtained independently by Blum, Codenotti, Gemmell and Shahoumian [16] where self-correcting using functional equations is studied in much greater depth.

Organization of paper In Section 2 we present the formal definitions of exact and robust characterizations from [38]. In Section 3 we investigate certain general properties of functional equations that influence whether they are robust. In Section 4 we present technical theorems showing conditions under which the general functional equation $F[f(x-y), f(x+y), f(x), f(y)] = 0$ is robust on domains that are finite groups. In Section 5 we present the self-testers and self-correctors based on the general form of the functional equation $\forall x, y \;\; F[f(x-y), f(x+y), f(x), f(y)] = 0$. In Section 6 we show how to convert

[3] f is random self-reducible if f can be computed at any particular input x via $f(x) = G[f(y_1), \ldots, f(y_k), y_1, \ldots, y_k]$ where G can be computed asymptotically faster than f and the y_i's are uniformly distributed, though not necessarily independent [19]. This notion of random self-reduciblity is somewhat different than other definitions given by [20] [1] [30], where the requirement on G is that it be computable in polynomial time.

the self-testers and self-correctors shown for finite groups into self-testers and self-correctors that apply to functions over rational domains. The completely specified self-tester and self-corrector for the particular example of the cosh function is described in Section 6.3. In Section 7 we discuss our conclusions and directions for further research.

2 Functional Equations and Characterizations

In this section, we give the definitions of functional equations and exact and robust characterizations. We also show a relationship between functional equations and probabilistically checkable proof systems.

2.1 Domains and ranges

Throughout this paper, we focus on the following three kinds of domains: The first are finite subsets of the rationals, of the form $\mathcal{D}_{n,s} = \{\frac{i}{s} | \ |i| \leq n\}$ where n, s are integers. These domains are not necessarily closed under addition and multiplication. This class includes domains that can be internally represented in a computer, corresponding to fixed point arithmetic, which have been used in previous work on self-testing and self-correcting [31][37]. The second type of domain that we are interested in are finite groups. Even for functions that are not defined over finite group domains, it is much simpler to first reason about the functional equations that they satisfy over finite group domains since they are closed under addition, and then to use the techniques of [31][37] (described in Section 6) for converting results on finite group domains into results on rational domains. The third class of domains are of use when studying periodic functions: $\mathcal{D}_s^b = \{i \cdot s | \ i \in \mathcal{Z}\}$ where b/s is an integer, and addition and multiplication is performed mod b. For example, $\mathcal{D}_{2\pi/10}^{2\pi} = \{0, 2\pi/10, 4\pi/10, 6\pi/10, \ldots, 18\pi/10\}$. Note that any function $f : \mathcal{D}_s^b \to \mathcal{R}$ corresponds to a function $\hat{f} : \mathcal{Z}_{b/s} \to \mathcal{R}$ by $\hat{f}(i) = f(i \cdot s \bmod b)$. Thus results on the finite group domains can be immediately applied to this third class of domains. The range of the functions considered can in general be arbitrary. If not specified, the range is assumed to be the reals.

In Figures 1 and 2, solutions to functional equations over the reals are given. It may happen that the functional equation over the reals characterizes a family of functions that is a proper subset of the functions characterized by the same functional equation over $\mathcal{D}_{p,s}$. In Section 5.2 we show that this does not limit the ability to construct self-testers for programs for these functions, due to the equality testing performed by self-testers.

2.2 Functional Equations

In the text by Aczél [3] (p.1), functional equations are defined by first defining a *term*:

Definition 1 (term) *([3] p.1)*

1. *The independent variables $x_1, \ldots, x_k$ are terms.*
2. *Given that $A_1, \ldots, A_m$ are terms and that H is a function of m variables, then $H(A_1, \ldots, A_m)$ is also a term.*

3. There are no other terms.

Definition 2 (functional equation) *([3] p.2) A* functional equation *is an equation $A_1 = A_2$ between two terms A_1, A_2 which contains k independent variables $x_1, \ldots, x_k$ and $n \geq 1$ unknown functions $H_1, \ldots, H_n$ of $j_1, \ldots, j_n$ variables respectively, as well as a finite number of known functions.*

The known functions used in [3] include addition, subtraction, division, multiplication, exponentiation, trigonometric and hyperbolic functions. In this paper, we will also include all functions computable by a Turing machine. Later in [3] (p.3) it is also noted that the functional equation must be identically satisfied for certain values of the variables $(x_1, \ldots, x_k)$ figuring in them, called the domain (we use the term neighborhood set in this paper). A *particular solution* of a functional equation is a function that satisfies the equation in the given domain (neighborhood set). The *general solution* is the set of all solutions belonging to the *class of admissible functions*, which can for example be defined by the analytic properties (measurability, differentiability, continuity, boundedness), other properties such as computability by a polynomial time Turing machine, ...), by initial and boundary conditions and/or by conditions given in the form of another functional equation.

2.3 Exact and Robust Characterizations

We now present the definitions of characterizations and robust characterizations given by [38]. $\mathcal{D}$ is used to represent a finite domain. We consider families of functions $\mathcal{F}$ where $f \in \mathcal{F}$ maps elements from domain $\mathcal{D}$ to range $\mathcal{R}$ (we use $\mathcal{R}$ to denote the range of a function and $\Re$ to denote the set of real numbers). $\mathcal{T}$ is a range containing 0. We illustrate these definitions using the example of linear functions. Here $\mathcal{D} = \mathcal{R} = \mathcal{T} = \mathcal{Z}_p$ and the family of linear functions is $\{f_a | a \in \mathcal{Z}_p$ where $f_a(x) = a \cdot x\}$.

Definition 3 (Neighborhoods) *[38] $N_{\mathcal{D}}$ is a k-*local neighborhood *if it is an ordered tuple of (not necessarily distinct) k points $(x_1, \ldots, x_k)$ from $\mathcal{D}^k$. A k-local collection of neighborhoods $\mathcal{N}_{\mathcal{D}}$ is a (multi)set of k-local neighborhoods. When $\mathcal{D}$ is understood from the context, we drop it from the subscript.*

Definition 4 (Properties) *[38] $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ is a k-*local property *if it is a function from $\mathcal{R}^k \times \mathcal{D}^k$ to $\mathcal{T}$. We say that a function $f : \mathcal{D} \to \mathcal{R}$* satisfies a property *$\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ over a neighborhood $N_{\mathcal{D}} = (x_1, \ldots, x_k)$ if $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}(f(x_1), \ldots, f(x_k), x_1, \ldots, x_k) = 0$.[4] When $\mathcal{D}, \mathcal{R}, \mathcal{T}$ are understood from the context, we drop them as subscripts.*

Definition 5 (Exact Characterizations) *[38] We say that $(\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}, \mathcal{N}_{\mathcal{D}})$ is an* exact characterization *of a family $\mathcal{F}$ of functions if a function $f : D \to R$ satisfies $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ over all neighborhoods $N_{\mathcal{D}} \in \mathcal{N}_{\mathcal{D}}$ exactly when $f \in \mathcal{F}$. The characterization is k-local if the property $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ and the collection $\mathcal{N}_{\mathcal{D}}$ is k-local. When $\mathcal{D}, \mathcal{R}, \mathcal{T}$ are understood from the context, we drop them as subscripts.*

[4] This is a slight modification of the definition in [38], where the function f satisfies $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ if $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}(f(x_1), \ldots, f(x_k), x_1, \ldots, x_k) = 1$ and the range of the function $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ is $\{0, 1\}$ instead of $\mathcal{T}$.

In our example, $\mathcal{R} = \mathcal{T} = \mathcal{D} = \mathcal{D}' = \mathcal{Z}_p$. The collection of neighborhoods is $\mathcal{N} = \{(x, y, x+y) | x, y \in \mathcal{Z}_p\}$. The property $\mathcal{P}$ which for $\{x_1, x_2, x_3\}$ computes $\mathcal{P}(f(x_1), f(x_2), f(x_3), x_1, x_2, x_3) = f(x_1) + f(x_2) - f(x_3)$ is 3-local. $(\mathcal{P}, \mathcal{N})$ is a 3-local characterization of the family of the linear functions $\{f | f(x) = c \cdot x, c \in \mathcal{Z}_p\}$.

Definition 6 (Robust Characterizations) *[38] Let $\mathcal{D}' \subseteq \mathcal{D}$. Let $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ be a property over a collection of neighborhoods $\mathcal{N}_{\mathcal{D}}$; let $\mathcal{F}$ be such that $(\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}, \mathcal{N}_{\mathcal{D}})$ is an exact characterization of $\mathcal{F}$. We say that the characterization $\mathcal{A} \equiv (\mathcal{D}', \mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}, \mathcal{N}_{\mathcal{D}})$ is an (ϵ, δ)-*robust characterization of *$\mathcal{F}$ in $\mathcal{G}$ if whenever a function $f \in \mathcal{G}$ satisfies $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$ on all but δ fraction of the neighborhoods in $\mathcal{N}_{\mathcal{D}}$, it is ϵ-close on domain $\mathcal{D}'$ to some function $g \in \mathcal{F}$.*[5] *When $\mathcal{D}, \mathcal{D}', \mathcal{R}, \mathcal{T}$ are understood from the context, we drop those parameters.*

We remark that in order for a robust characterization to be useful, membership in $\mathcal{G}$ should be efficient to test, choosing a random neighborhood in $\mathcal{N}_{\mathcal{D}}$ should be efficient, and $\mathcal{D}'$ should be a fairly large subset of $\mathcal{D}$. All of our results have these properties. In most examples, $\mathcal{G}$ will be the set of all functions, however we will see examples in which it is useful to have $\mathcal{G}$ be a smaller, efficiently recognizable, set of functions.

To continue with the example of linear functions, a theorem of [19] can be used to say that for any finite group G and any $\delta < \frac{2}{9}$, $(\mathcal{P}_G, \mathcal{N}_G)$ is a $(2\delta, \delta)$-robust characterization of the linear functions mapping G to G.

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

We now define what it means for a characterization to be robust over a class.

Definition 7 *Let $\mathcal{S} = \{(\mathcal{A}_1, \mathcal{F}_1, \mathcal{G}_1), (\mathcal{A}_2, \mathcal{F}_2, \mathcal{G}_2), \ldots\}$ be such that for all i, $\mathcal{A}_i = (\mathcal{D}_i', \mathcal{P}_{\mathcal{D}_i,\mathcal{R}_i,\mathcal{T}_i}, \mathcal{N}_{\mathcal{D}_i})$ and $(\mathcal{P}_{\mathcal{D}_i,\mathcal{R}_i,\mathcal{T}_i}, \mathcal{N}_{\mathcal{D}_i})$ is an exact characterization of $\mathcal{F}_i$. We say that $(\mathcal{P}, \mathcal{N})$ is* robust over the family $\mathcal{S}$, *if:*

1. *there is a function $\mathcal{N}$ which takes as input i and returns a Turing machine M such that M on input a random string chooses a random member $N \in \mathcal{N}_{\mathcal{D}_i}$*
2. *there is a function $\mathcal{P}$ which takes as input i and returns a Turing machine that on input $N \in \mathcal{N}_{\mathcal{D}_i}$, computes $\mathcal{P}_{\mathcal{D}_i,\mathcal{R}_i,\mathcal{T}_i}(N)$*
3. *for all $\epsilon < 1$ there is a $\delta < 1$ such that for all i, $\mathcal{A}_i$ is an (ϵ, δ)-robust characterization.*[6]

In order for a robust characterization over a class to be useful, the functions $(\mathcal{P}, \mathcal{N})$ should have a uniform and concise description. In particular, functional equations have a natural interpretation as a concise description of robust characterizations over a class. In this paper, we consider variations of the following two basic types of classes:

[5] It is convenient for our results in this paper to define $\mathcal{N}_{\mathcal{D}}$ as a multiset and to define robust characterizations in terms of picking neighborhood sets from the uniform distribution on $\mathcal{N}_{\mathcal{D}}$. Alternatively, one can define robust characterizations in terms of a distribution on ordered sets, where $\mathcal{N}_{\mathcal{D}}$ would correspond to the support of the distribution.

[6] The only interesting case is when the size of $\mathcal{S}$ is infinite, since otherwise there are always constants ϵ, δ such that all characterizations in the collection are (ϵ, δ)-robust.

In the first type of class, we capture the property that a functional equation is (ϵ, δ)-robust over all domains that have a certain structure, such as finite groups. The functional equation $\mathcal{P}, \mathcal{N}$ is described with a generic group or field operation. Let $\mathcal{D}_i = \mathcal{D}'_i = \mathcal{R}_i = \mathcal{T}_i = G_i$, where G_i is the i^{th} group (for an arbitrary ordering of the groups) with group operator $+_{G_i}$. $\mathcal{P}_{\mathcal{D}_i,\mathcal{R}_i,\mathcal{T}_i}$ and $\mathcal{N}_{\mathcal{D}_i}$ are then the functions obtained by using the group operator $+_{G_i}$. In our linearity example, $\mathcal{P}(i) = \mathcal{P}_{\mathcal{D}_i,\mathcal{R}_i,\mathcal{T}_i}$ is the function $\mathcal{P}(f(x_1), f(x_2), f(x_3), x_1, x_2, x_3) = f(x_1) +_{G_i} f(x_2) -_{G_i} f(x_3)$. The collection of neighborhoods $\mathcal{N} = \{(x, y, x +_{G_i} y) | x, y \in G_i\}$.

In the second type of class, we concentrate on finite subsets of various sizes of an infinite group. The functional equation is defined over a large, possibly infinite domain such as the rationals. However, the robust characterization is defined over a finite subset of the domain. Let $\mathcal{D}_i = \mathcal{D}_{n_i,s}$, $\mathcal{D}'_i = \mathcal{D}_{i,s}$ for $n_i \geq i$ (the exact value of n_i is determined by the robust characterization) and $\mathcal{R} = \mathcal{T} = \Re$. $\mathcal{P}, \mathcal{N}$ always return the same function which maps the rationals to the reals. In our linearity example, $\mathcal{P}$ is the function $\mathcal{P}(f(x_1), f(x_2), f(x_3), x_1, x_2, x_3) = f(x_1) + f(x_2) - f(x_3)$ and $\mathcal{N}$ is a carefully chosen subset of $\{(x, y, x + y) | x \in \mathcal{D}_{n_i,s}, y \in \mathcal{D}_{n_i,s}\}$ (see Section 6). Operations $+, -$ are the usual group operations over the reals.

Our results specify the class of domains and ranges over which the functional equation is robust. When $\mathcal{S}$ is understood from the context, we say that $(\mathcal{P}, \mathcal{N})$ is robust.

2.4 Robustness and Probabilistically Checkable Proofs

A language L in NP has a *probabilistically checkable proof system* if there is a probabilistic polynomial time Turing machine V (the verifier) that has read access to a source of random strings R and to a proof P for the membership of x in L, such that (1) if $x \in L$, there exists a proof P of membership in the language such that V accepts P with probability 1 (where the probability is over the random strings R) and (2) if x is not in L, for all proofs P', V accepts proof P' with probability at most 1/4 [29]. The linearity test and tests for low total degree polynomial functions that are given in [19] [37] [6] have been used to construct probabilistically checkable proof systems in the recent results of [6] [14] [12] (tests that functions are low degree in each variable are given and used in [9] [29] [7]). Much recent research has been devoted to expanding the range of the robustness parameter δ for which these tests work, as it directly influences the strength of results showing that it is computationally difficult to approximate certain NP-complete problems [36][11] [12].

Conversely, Sudan [41] has noted that the property of being a probabilistically checkable proof can actually be viewed as an example of a robust functional equation (where the definition of F is generalized to include all polynomial time circuits): In the work of Arora, Lund, Motwani, Sudan and Szegedy [6], each probabilistically checkable proof P can be viewed as a truth table of a function. If P is n bits long then P can be thought of the function $P : [1 \ldots n] \rightarrow \{0, 1\}$, i.e., $P(i)$ is the i^{th} bit of the proof. The protocol followed by V is to choose an r-bit random string y, perform a computation in order to determine a constant number of locations $\sigma_1(y), \ldots, \sigma_k(y)$, query the proof at those locations, and then perform another computation on input $(y, P(\sigma_1(y)), \ldots, P(\sigma_k(y)))$ in order to determine whether to accept or reject the proof. More formally, let $\mathcal{N} = \{(\sigma_1(y), \ldots, \sigma_k(y)) | y \in \{0, 1\}^r\}$. The verifier's choice of a random string in $\{0, 1\}^r\}$ determines a choice of a neighborhood

$(y_1, \ldots, y_k)$ from $\mathcal{N}$ by the computation $y_i = \sigma_i(y)$. The verifier then tests whether a relationship $\forall (y_1, \ldots, y_k) \in \mathcal{N},\ F[P(y_1), P(y_2), \ldots, P(y_k), y] = 0$ is satisfied, where F is computable by a polynomial time Turing machine and describes the computation of the verifier that determines whether to accept or reject the proof. In [6] it is shown that one can construct an $(F, \mathcal{N})$ that characterizes the set of valid proofs, i.e., valid proofs are exactly those bit strings P for which $F[P(y), P(\sigma_1(y)), \ldots, P(\sigma_k(y)), y] = 0$ is satisfied for all random strings y. Furthermore only proofs that are close (equal on most bits) to some valid probabilistically checkable proof are passed with probability $\geq 3/4$.

3 Characterizing Robust Functional Equations

We turn to the general question of how to distinguish functional equations that are robust from those that are not, in order to arrive at a better understanding of what makes a functional equation robust. It turns out that the structure of the neighborhood set is a very important determining factor to whether or not the functional equation is robust. To illustrate, the following are three characterizations of the lines $\mathcal{F} = \{f | f : Z_p \rightarrow Z_p, f(x) = ax + b \text{ for } a, b \in \mathcal{Z}_p\}$ (this family of lines is different from the one discussed previously) and over the class $\mathcal{S}$ in which $\mathcal{R}_i = \mathcal{T}_i = \mathcal{D}_i = \mathcal{D}'_i = \mathcal{Z}_{p_i}$ where p_i is the i^{th} prime:

1. $\forall x_1, x_2, x_3 \in Z_{p_i}, \frac{f(x_1)-f(x_2)}{x_1-x_2} = \frac{f(x_2)-f(x_3)}{x_2-x_3}$.
2. $\forall x_1, x_2 \in Z_{p_i},\ f(x_1) - 2f(x_2) + f(2x_2 - x_1) = 0$, or equivalently, $\mathcal{N}_{\mathcal{D}} = \{(x_1, x_2, 2x_2 - x_1) | x_1, x_2 \in Z_{p_i}\}$ and $\forall (x_1, x_2, x_3) \in \mathcal{N}_{\mathcal{D}}, f(x_1) - 2f(x_2) + f(x_3) = 0$.
3. $\forall x_1 \in Z_{p_i},\ f(x_1) - 2f(x_1 + 1) + f(x_1 + 2) = 0$, or equivalently, $\mathcal{N}_{\mathcal{D}} = \{(x_1, x_1 + 1, x_1 + 2) | x_1 \in Z_{p_i}\}$ and $\forall (x_1, x_2, x_3) \in \mathcal{N}_{\mathcal{D}}, f(x_1) - 2f(x_2) + f(x_3) = 0$.

In all three characterizations, the property is the same (although simplified in the latter two characterizations because of the specially chosen neighborhoods), and ensures that the points $(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))$ all lie on a single line. The only difference in the three characterizations is the collection of neighborhoods over which it is defined. However, the choice of neighborhoods heavily influences the robustness of the characterizations. A simple counting argument (similar to the one described later in Section 3.2) shows that the first property is (δ, δ)-robust for all $\delta < 1$. The second property is $(2\delta, \delta)$-robust for $\delta < 1/1082$ [38]. It is easy to see that for all ϵ, the third property is not (ϵ, δ)-robust over $\mathcal{S}$ for any constant δ. Thus the richness of the neighborhood set influences the robustness as well as the complexity of computing $\mathcal{P}_{\mathcal{D},\mathcal{R},\mathcal{T}}$. Another interesting quantity related to the neighborhood set is the number of random bits required to choose a random element of the neighborhood set (or the logarithm of the size of the neighborhood set). Reducing this quantity, even by a constant factor, while not significantly affecting the range of δ, ϵ achievable for maintaining a robust characterization (and thus not significantly reducing the efficiency of the characterization), has been useful for constructing more efficient probabilistically checkable proofs [14] [12].

We begin by investigating two extreme types of functional equations:

1. A *k-minimal neighborhood set* is one in which $\mathcal{N}_\mathcal{D}$ is described by $k-1$ functions $\sigma_1(x), \ldots, \sigma_{k-1}(x)$ where the σ_i's are arbitrary functions mapping $\mathcal{D}$ to $\mathcal{D}$. $\mathcal{N}_\mathcal{D}$ is of the form $\{(x, \sigma_1(x), \ldots, \sigma_{k-1}(x)) | x \in \mathcal{D}\}$. Since once the first element of the neighborhood is chosen, the other elements are uniquely determined, the cardinality of the neighborhood set is at most $|\mathcal{D}|$. $\mathcal{N}_\mathcal{D}$ in the third example uses a 3-minimal neighborhood set. A *minimal functional equation* is one in which the neighborhood set is k-minimal for some constant k.

2. A *k-total neighborhood set* is one in which $\mathcal{N}_\mathcal{D} = \mathcal{D}^k$, relating the function at each input x to function values at *all* subsets of $k-1$ other inputs. $\mathcal{N}_\mathcal{D}$ in the first example uses a 3-total neighborhood set. A *total functional equation* is one in which the neighborhood set is k-total for some constant k.

We isolate a key combinatorial property of the neighborhood sets of functional equations and show that having this property is a necessary condition for robustness. We apply this combinatorial property to show a general condition under which functional equations with minimal neighborhood sets are not robust. As we will see later on, this result implies that certain methods of testing programs used in practice are related to this class and are therefore provably faulty. For example, our techniques apply to the functional equation that is used to test the real gamma function [24] (page 6). On the other hand, we mention an example, given by Sudan [41], of a minimal equation that is robust. We then show conditions under which k-total equations are always robust.

In the following, we assume that $\mathcal{D} = \mathcal{D}'$.

3.1 Minimal functional equations.

One might conjecture that minimal equations cannot be robust, since for most inputs x, the function value at x is related to the function values at very few other inputs. We show a class of minimal equations that are provably not robust. We then describe an example of a minimal robust functional equation.

A Combinatorial Property of Robustness. We first define a graph which captures much of the information in the neighborhood set $\mathcal{N}_\mathcal{D} = \{(x, \sigma_1(x, \bar{y}), \ldots, \sigma_k(x, \bar{y})) | x \in \mathcal{D}, \bar{y} \in \mathcal{D}^k\}$: Given the functional equation $\forall x \in \mathcal{D}, \bar{y} \in \mathcal{D}^k, \quad F[f(x), f(\sigma_1(x, \bar{y})), \ldots, f(\sigma_k(x, \bar{y}))] = 0$, for $\sigma_i : \mathcal{D} \times \mathcal{D}^k \rightarrow \mathcal{D}$, we define the undirected multigraph $G_{\mathcal{N}_\mathcal{D}} = (V, E)$ where the vertices correspond to elements of $\mathcal{D}$ and the edges are $E = \{(x, \sigma_j(x, \bar{y})) | x \in \mathcal{D}, \bar{y} \in \mathcal{D}^k, 1 \leq j \leq k\}$ (there may be more than one edge between u and v if there is more than one $i, \bar{y}$ such that $\sigma_i(u, \bar{y}) = v$).

For example, the graph of $\forall x \in \mathcal{Z}, \ f(x) + 1 - f(x+1) = 0$ corresponds to a path, and the graph of $\forall x, y, \ f(x) + f(y) - f(x+y) = 0$ corresponds to a complete graph with two edges between every pair of nodes.

The following result applies to functional equations defining classes of functions that can be thought of as codewords with very large ($>> 1/2$) distance.

Definition 8 *An α-separated function family $\mathcal{F}$ over domain $\mathcal{D}$ is one for which $|\mathcal{F}| \geq 2$ and $\forall f_i, f_j \in \mathcal{F}, \ Pr_{x \in \mathcal{D}}[f_i(x) = f_j(x)] \leq \alpha$.*

For example, for all of the functional equations mentioned in Figures 1,2, for all α, there are integers n, s such that the functional equations over the domain $\mathcal{D}_{n,s}$ characterize α-separated function families.

The following theorem shows a relationship between the connectivity of $G_{\mathcal{N}_\mathcal{D}}$ and the robustness of the functional equation $(F, \mathcal{N}_\mathcal{D})$: If $(F, \mathcal{N}_\mathcal{D})$ is (ϵ, δ)-robust, then more than δ/k fraction of the edges in the graph $G_\mathcal{N}$ must be removed in order to separate $G_\mathcal{N}$ into two "large" components, each of size $\geq (\epsilon + \alpha)|V|$.

Theorem 9 *Let $\mathcal{N} = \{(x, \sigma_1(x, \bar{y}), \ldots, \sigma_k(x, \bar{y}))|x \in \mathcal{D}, \bar{y} \in \mathcal{D}^k\}$. Suppose that $\mathcal{F}$, characterized by $(F, \mathcal{N})$, is an α-separated function family. If $G_\mathcal{N}$ has a set of edges E' such that (1) $|E'| \leq \frac{\delta}{k}|E|$ and (2) removing E' separates the vertices of $G_\mathcal{N}$ into two components, each of size $\geq (\epsilon + \alpha)|V|$, then $(F, \mathcal{N})$ is not an (ϵ, δ)-robust characterization.*

Proof: Suppose E' separates G_F into sets A, B. Consider the function h which labels vertices in A according to f_1 and vertices in B according to f_2, for some $f_1, f_2 \in \mathcal{F}$. Since $\mathcal{F}$ is α-separated, we have that $\forall f_i \in \mathcal{F}, Pr_{x \in \mathcal{D}}[f_i(x) \neq h(x)] \geq \epsilon$. However, only tests using edges that cross the cut will fail. Since $x, \bar{y}$ are chosen uniformly, the edges are also chosen uniformly. Thus tests will fail with probability $\leq \delta$. $\square$

Application to minimal functional equations It is easy to see that for any minimal equation F of the form $F[f(x), f(\sigma(x))] = 0$, G_F can be separated into two large components by removing very few edges (by Theorem 9), and thus for all classes $\mathcal{S}$ in which the domain size is not bounded, the functional equation $(F, \mathcal{N})$ for $\mathcal{N} = \{(x, \sigma(x))|x \in \mathcal{D}\}$ over $\mathcal{S}$ is not robust.

It was shown by Klawe [32] that for any given ϵ, any graph on n nodes whose edges are defined by a constant number of linear functions has a cut containing $o(n)$ edges which separates the graph into two large portions, each containing an ϵ fraction of the nodes. The following corollary applies to functional equations relating points x to points that are linear functions of x:

Corollary 10 *Given n, s, let $\mathcal{D}_{n,s} = \mathcal{D} = \mathcal{D}'$. $\mathcal{R} = \mathcal{T} = \Re$. Let $\sigma_1, \ldots, \sigma_k$ be any family of linear functions over the rationals of the form $\sigma_i(x) = ax+b$, where a, b are rational, and let $\mathcal{F}$ be an α-separated function family satisfying the equation $F[f(x), f(\sigma_1(x)), f(\sigma_2(x)), \ldots, f(\sigma_k(x))] = 0$. Then there exists a constant $0 < \epsilon < 1$ such that $(F, \mathcal{N})$ is not (ϵ, δ)-robust for any constant δ.*

Thus, if $\mathcal{S}$ is a class such that $\mathcal{D}_{i,s} = \mathcal{D}_i = \mathcal{D}_i'$, $\mathcal{R}_i = \mathcal{T}_i = \Re$, and $\sigma_1, \ldots, \sigma_k$ are any family of linear functions over the rationals of the form $\sigma_i(x) = ax + b$, where a, b are rational, the functional equation $(F, \mathcal{N})$ for $\mathcal{N} = \{(x, \sigma_1(x), \ldots, \sigma_k(x))|x \in \mathcal{D}\}$ over $\mathcal{S}$ is not robust. A similar result applies for linear functions over finite groups.

This corollary shows that many tests that are used in practice to test programs should be used with more care. For example, in the functional equation

$$\forall x,\ f(2x) - (2\pi)^{-1/2} 2^{2x-1/2} f(x) f(x + 1/2) = 0$$

used for testing the real gamma function by [24], all of the σ_i's are linear functions ($\sigma_1(x) = 2x, \sigma_2(x) = x + 1/2$). Thus the corollary implies that there exist programs which are very

different from any solution to this functional equation, yet pass the test most of the time. The direct use of polynomial checks suggested in [43] also yields functional equations which are not robust due to Corollary 10, however, the work of [16] shows how to transform the polynomial checks into more general functional equations for which the negative results in this section do not apply.

A robust minimal functional equation Previous examples of robust functional equations have always been usable for self-correction as well. This might lead one to think that usability for self-correction might be another necessary condition for robustness. However, this may note be the case: It is not known how to use minimal functional equations for self-correction. Even so, there are minimal functional equations that are robust and can therefore be used to self-test. We describe an example of a minimal functional equation that is robust. This example was given by Sudan [41]:

We say that a graph $G(V,E)$ is an α-expander if for all $S \subseteq V, |S| \leq |V|/2$, the set of nodes that are neighbors of S (not including nodes in S), is of size $\geq \alpha|S|$. Fix constants d and α. Let $G_i(V,E)$ be any degree d α-expander on i nodes, such that the vertices are labelled by elements of the domain D. Let the functional equation be $\forall (u,v) \in E,\ f(u) - f(v) = 0$. Since G_i is connected, the only functions which are solutions to this functional equation are the constant functions. Assume that the functional equation is satisfied for most $(u,v) \in E$. Suppose one deletes all edges for which the functional equation does *not* hold. Since G_i is an expander, there must exist a large connected component in G_i (containing at least a constant fraction of the nodes), even after deleting the edges. The large connected component will correspond to elements of the domain that agree with a single constant function. Let $\mathcal{S}$ be the class corresponding to the above functional equation on $G_0, G_1, \ldots$. Then for all ϵ, there is a δ such that the above functional equation is (ϵ, δ)-robust on $\mathcal{S}$.

3.2 Total functional equations.

On the other end of the spectrum, we consider a class of functional equations where there are no restrictions on the way inputs are related, and show that if some technical conditions are satisfied, then they are always robust.

Given $\mathcal{D}$ and $\mathcal{R}$, let $F[f(x), f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}] = 0,\ \forall x, y_1, \ldots, y_k$ be a k-total functional equation characterizing functions $f : \mathcal{D} \to \mathcal{R}$. Assume further that F can be solved for $f(x)$, namely $f(x) = G[f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}]\ \forall y_1, \ldots, y_{k-1}$ (because of the totality of the equation, F and G depend on $x, y_1, \ldots, y_{k-1}$ as well as the function values at those points). We say that the solution $\mathcal{F}$ to equation F is $(k-1)$-*complete* if $\forall((y_1, w_1), \ldots, (y_{k-1}, w_{k-1})) \in (\mathcal{D}, \mathcal{R})^{k-1},\ \exists f \in \mathcal{F}$ such that $f(y_i) = w_i$ for all $1 \leq i \leq k-1$. An example of a $(k-1)$-complete function family is the family of degree $(k-1)$ polynomials. The following theorem, which says that k-total functional equations that characterize $k-1$-complete function families are necessarily robust, can be viewed as a generalization of a known theorem for degree $(k-1)$ univariate polynomials (cf. [40]).

Theorem 11 *Let $\mathcal{N} = \mathcal{D}^k$. Suppose the k-total functional equation $F[f(x), f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}] \equiv f(x) - G[f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}] = 0\ \forall x, y_1, \ldots, y_{k-1}$ has a $k-1$-complete solution $\mathcal{F}$. Then $(F, \mathcal{N})$ is (δ, δ)-robust $\forall 0 < \delta < 1$.*

Proof: Suppose f satisfies $Pr_{x,\bar{y}}[f(x) = G[f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}]] \geq 1 - \delta$. Then there exists $z_1, \ldots, z_{k-1}$, a particular setting of the y_i's, such that the test works for $\geq 1 - \delta$ of the x's. Let $g(x)$ be the function in $\mathcal{F}$ determined by the values of the program at $z_1, \ldots, z_{k-1}$. Then $Pr_x[g(x) = f(x)] \geq 1 - \delta$. Thus $\forall x, y_1, \ldots, y_{k-1}\ g(x) - G[g(y_1), \ldots, g(y_{k-1}), x, y_1, \ldots, y_{k-1}] = 0$. $\square$

Any function f, that satisfies such a total functional equation F can be computed, given the value of the function at any fixed k locations, as efficiently as evaluating the functional equation G. Thus, if F and G have the same complexity, the functional equation is not useful for self-testing, since it does not have the "little-oh property" described in [18]. However, it is possible that more efficient self-testers can be constructed by looking at a smaller, carefully chosen, set of neighborhoods $\mathcal{N}$ and showing that the functional equation is still robust over $\mathcal{N}$. Given $\vec{\sigma} = (\sigma_1, \ldots, \sigma_{k-1})$, such that $\sigma_i : \mathcal{D} \times \mathcal{D}^{k-1} \rightarrow \mathcal{D}$, suppose functional equations $F[f(x), f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}] = 0\ \forall x, y_1, \ldots, y_{k-1}$, and $F[f(x), f(\sigma_1(x, \bar{z})), \ldots, f(\sigma_{k-1}(x, \bar{z})), x, \sigma_1(x, \bar{z}), \ldots, \sigma_{k-1}(x, \bar{z})] = 0\ \ \forall x, \bar{z}$ both have the same complete solution $\mathcal{F}$. Due to the structure of the σ's, it might be the case that F is easier to compute on those tuples defined by the σ's (for example, efficient polynomial degree tests have been constructed by only performing tests on points that are evenly spaced: $\sigma_i(x, z) = x + iz$ [38]). If F is also robust over random choices of $x, \bar{z}$, then a more efficient tester can be constructed.

We use a bound on the runtime of the program being tested to devise a tester. The works of Blum, Evans, Gemmell, Kannan and Naor [17] and Micali [35] also construct checkers based on bounds on the runtime of the program being checked, but use very different methods. Let the distribution $\mathcal{V}_{\vec{\sigma}}$ be the distribution defined by picking $x \in \mathcal{D}, \bar{z} \in \mathcal{D}^{k-1}$ randomly, and outputting $(x, \sigma_1(x, \bar{z}), \ldots, \sigma_{k-1}(x, \bar{z}))$. Let $\mathcal{U}$ be the distribution defined by picking $x, y_1, \ldots, y_{k-1} \in \mathcal{D}$ and outputting $(x, y_1, \ldots, y_{k-1})$. If the σ's look "random enough", we have the following theorem showing a sense in which F is robust over random choices of $x, \bar{z}$. This theorem implies that it is enough to test points related by the σ's.

Theorem 12 *Let $Time(c) \equiv \{f | f$ is computable on inputs of length n in time $n^c\}$. Let c be an arbitrary constant, and fix $0 \leq \epsilon \leq 1$ and $0 \leq \delta \leq 1$. Let E_1 denote the functional equation $F[f(x), f(y_1), \ldots, f(y_{k-1}), x, y_1, \ldots, y_{k-1}] = 0\ \forall x \in \mathcal{D}, \bar{y} \in \mathcal{D}^{k-1}$, and E_2 denote the functional equation $F[f(x), f(\sigma_1(x, \bar{z})), \ldots, f(\sigma_{k-1}(x, \bar{z})), x, \sigma_1(x, \bar{z}), \ldots, \sigma_{k-1}(x, \bar{z})] = 0\ \forall x \in \mathcal{D}, \bar{z} \in \mathcal{D}^{k-1}$. Assume that:*

1. *E_1 and E_2 have the same complete solution $\mathcal{F}$,*
2. *F can be computed by a circuit of size $(\log |\mathcal{D}|)^c$,*
3. *no circuit of size $\leq (k+1)(\log |\mathcal{D}|)^c$ can distinguish inputs from $\mathcal{V}_{\vec{\sigma}}$ and $\mathcal{U}$ with more than δ advantage,*
4. *E_1 is is $(\epsilon, 2\delta)$-robust.*

Then E_2 can be used to test all programs running in time $(\log |\mathcal{D}|)^c$: if $Pr_{x,R}[F[P(x), P(\sigma_1(x, \bar{z})), \ldots, P(\sigma_{k-1}(x, \bar{z})), x, \sigma_1(x, \bar{z}), \ldots, \sigma_{k-1}(x, \bar{z})] = 0] \geq 1 - \delta$, and P runs in $\leq (\log |\mathcal{D}|)^c$ steps, then there is a $f \in \mathcal{F}$ such that $Pr_x[P(x) = f(x)] \geq 1 - \epsilon$. Thus, $(\mathcal{F}, \mathcal{D}^k)$ is an (ϵ, δ)-robust characterization of $\mathcal{F}$ in $Time(c)$.

Proof: Suppose that there exists some program P running in time $(\log |\mathcal{D}|)^c$ which is wrong at $\geq \epsilon$ inputs in $\mathcal{D}$ but passes the tester with probability $> 1 - \delta$. We use it to construct a program $\mathcal{A}$ of size $\leq (k+1)(\log |\mathcal{D}|)^c$ that can distinguish between outputs from distributions $\mathcal{V}$ and $\mathcal{U}$ with more than δ advantage (which contradicts (3)). $\mathcal{A}$ receives $w, z_1, \ldots, z_{k-1}$, and tests whether $F[P(x),P(z_1)),\ldots,P(z_{k-1}),x,z_1,\ldots,z_{k-1}] = 0$. $\mathcal{A}$ outputs 1 if P passes the test and 0 if P fails. By Theorem 11, $Pr_{x,y_1,\ldots,y_{k-1}\in\mathcal{U}}[F[P(x), P(y_1), \ldots, P(y_{k-1}), x, y_1, \ldots, y_{k-1}] \neq 0] \geq 2\delta$, and by the assumption, $Pr_{x,y_1,\ldots,y_{k-1}\in\mathcal{V}}[F[P(x), P(y_1), \ldots, P(y_{k-1}), x, y_1, \ldots, y_{k-1}] \neq 0] < \delta$. $\square$

4 Robustness of $\forall x, y,\ F[f(x-y), f(x+y), f(x), f(y)] = 0$

We study conditions under which any member of the general class of functional equation $\forall x, y\ F[f(x-y), f(x+y), f(x), f(y)] = 0$ is robust. We show that addition theorems $\forall x, y\ f(x+y) = G[f(x), f(y)]$ for which G satisfies $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$ (which all of our examples satisfy) are robust over the class $\mathcal{S}$, such that the domains in $\mathcal{S}$ are finite groups, and then give a technique which applies to a number of functional equations that are not addition theorems. Our techniques apply to all functional equations in Figure 1 as well as the first three functional equations in Figure 2. We conjecture that all functional equations in this class are robust.

All results can be extended to rational domains of the form $\mathcal{D}_{p,s} = \{\frac{i}{s} | |i| \leq p\}$ using standard techniques from [31][37]. We give an example of such an extension in Section 6. Our only assumption on $\mathcal{R}$ in Subsection 4.1 is that it is an (possibly infinite) group. In Subsection 4.2 we assume that $\mathcal{R}$ is a field.

4.1 Addition theorems

We show that any addition property $\forall x, y\ \ f(x+y) = G[f(x), f(y)]$ is $(2\delta, \delta)$-robust for $\delta < 1/8$ and G that satisfies $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$ (we do not attempt to optimize the relationship between ϵ and δ in our proofs of (ϵ, δ)-robustness – see [13], [26] and [11] for techniques for improving this relationship). Therefore, knowing that $f(x+y) = G[f(x), f(y)]$ holds at more than a $\frac{7}{8}$ fraction of the (x, y) pairs is enough to conclude that f agrees with some solution of the addition theorem G on at least $\frac{3}{4}$ fraction of the inputs. One can verify that G satisfies $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$ in all of the examples given in Figure 1.

At the end of this subsection, we consider the requirement that $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$. We show that that if the domain is a subset of a field, such that rational functions are defined (a function $f(x, y) = p(x, y)/q(x, y)$ where p, q are polynomials), then we can make a general claim for any constant degree rational function G that is based on the number of zeros that a rational function can have. Similar results that apply to algebraic functions can be proven for domains over which algebraic functions are defined (see [46]).

We now show that any addition theorem satisfying $\forall a, b, c\ \ G[a, G[b, c]] = G[G[a, b], c]$ is robust. This proof follows an outline similar to Coppersmith's version of the proof of robustness of the linearity test which is described in [19]. However, the inner manipulations are different. Hence, whereas Coppersmith's proof works for any $\delta \leq 2/9$, here we require $\delta \leq 1/8$.

Theorem 13 *Let $\mathcal{D} = \mathcal{D}'$ be a finite group and $\mathcal{R} = \mathcal{T}$ a group. Let $\mathcal{N}^{\text{add}} = \{(x, y, x + y) | x, y \in \mathcal{D}\}$. Let G be such that G satisfies $\forall a, b, c \in \mathcal{R}$ $G[a, G[b, c]] = G[G[a, b], c]$. Let $F(x_1, x_2, x_3) = f(x_3) - G[f(x_1), f(x_2)]$ on neighborhoods $(x_1, x_2, x_3) \in \mathcal{N}^{\text{add}}$. Then for all $\delta < 1/8$, $(F, \mathcal{N}^{\text{add}})$ is $(2\delta, \delta)$-robust. Letting $\mathcal{S}$ be a class such that $\mathcal{D}_i = \mathcal{D}_i'$ are finite groups and $\mathcal{R}_i = \mathcal{T}_i$ are groups, then since for all $\epsilon < 1/4$, $(F, \mathcal{N}^{\text{add}})$ is $(\epsilon, \epsilon/2)$-robust, $(F, \mathcal{N}^{\text{add}})$ is robust over $\mathcal{S}$.*

Proof: [of Theorem 13] To prove the theorem, we will show that if $Pr_{x,y \in_R \mathcal{D}}[f(x + y) = G[f(x), f(y)]] \geq 1 - \delta$, for $\delta < \frac{1}{8}$, then there exists a function g such that (1) $Pr_{x \in_R \mathcal{D}}[f(x) = g(x)] \geq 1 - 2\delta$ and (2) $\forall x, y$ $g(x + y) = G[g(x), g(y)]$.

Define $g(x)$ to be $\text{maj}_{z \in \mathcal{D}}\{G(f(x - z), f(z))\}$, where maj of a set is the function that picks the element occuring most often (choosing arbitrarily in the case of ties). We first show that g is 2δ-close to f:

Lemma 14 *g and f agree on more than $1 - 2\delta$ fraction of the inputs from $\mathcal{D}$.*

Proof: Consider the set of elements x such that $\Pr_z[f(x) = G[f(x - z), f(z)]] < 1/2$. If the fraction of such elements is more than 2δ then it contradicts the condition that $\Pr_{x,y}[f(x + y) = G[f(x), f(y)]] \geq 1 - \delta$. For all remaining elements, $f(x) = g(x)$. □

Next we show a sense in which g is well-defined:

Lemma 15 *For all x, $\Pr_z[g(x) = G[f(x - z), f(z)]] \geq 1 - 2\delta$.*

Proof: [7]

$\Pr_{y,z}[G[f(x - y), f(y)]$

$$\begin{aligned} &= G[G[f(x - y - z), f(z)], f(y)] \\ &= G[f(x - y - z), G[f(z), f(y)]] \\ &= G[f(x - (y + z)), f(y + z)]] \qquad \geq 1 - 2\delta \end{aligned}$$

The first and third equality hold with probability $1 - \delta$ by our assumption on f and since $x - y, y, z, x - y - z, z + y$ are all uniformly distributed in $\mathcal{D}$. The second equality always holds since $G[a, G[b, c]] = G[G[a, b], c]$ $\forall a, b, c$.

The lemma now follows from the well known fact that the probability that the same object is drawn twice from a set in two independent trials lower bounds the probability of drawing the most likely object in one trial: Suppose the objects are ordered so that p_i is the probability of drawing object i. Without loss of generality $p_1 \geq p_2 \geq \ldots$. Then the probability of drawing the same object twice is $\sum_i p_i^2 \leq \sum_i p_1 p_i = p_1$. □

Finally, we prove that g satisfies the addition theorem everywhere:

Lemma 16 *For all x, y, $g(x + y) = G[g(x), g(y)]$.*

Proof:

[7] For conciseness, we use a somewhat nonstandard notation: For random variables a, b, c, we reason about the probability that $a = c$ by using an intermediate variable b, using $Pr[a = c] \geq Pr[a = b = c] \geq 1 - Pr[a \neq b] - Pr[b \neq c]$.

$\Pr_{u,v}[G[g(x), g(y)]$

$$\begin{aligned}
&= G[G[f(u), f(x-u)], G[f(v), f(y-v)]] \\
&= G[f(u), G[f(x-u), G[f(v), f(y-v)]]] \\
&= G[f(u), G[G[f(x-u), f(v)], f(y-v)]] \\
&= G[f(u), G[f(x-u+v), f(y-v)]] \\
&= G[f(u), f(x+y-u)] \\
&= g(x+y)] \\
&> 1-8\delta > 0
\end{aligned}$$

By Lemma 15, the first equality holds with probability $1-4\delta$ and the last equality holds with probability $1-2\delta$. By the assumption on f, the fourth and fifth equalities each hold with probability $1-\delta$. The other equalities always hold, since $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$. Since the statement is independent of u, v and holds with positive probability, it must hold with probability 1. □
□ (Theorem 13)

4.1.1 Addition theorems that satisfy $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$

If the domain is a large enough subset of a field, such that rational functions are defined (a function $f(x, y) = p(x, y)/q(x, y)$ where p, q are polynomials), and if G is a rational function such that the numerator has bounded degree then one can show that G satisfies $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c$:

Theorem 17 *Let G be a constant degree rational function, such that the degree in each variable of the numerator of the rational function $H(a, b, c) \equiv G[a, G[b, c]] - G[G[a, b], c]$ is bounded by N. Assume that G is such that one of the solutions f to the functional equation $\forall x, y\ \ f(x+y) = G[f(x), f(y)]$ takes on at least $(N+1)^3$ values (in particular, $|\mathcal{R}| > (N+1)^3$). Then G satisfies $G[a, G[b, c]] = G[G[a, b], c]\ \forall a, b, c \in \mathcal{R}$.*

Proof: Since H is a rational function, it will suffice to show that H evaluates to 0 on many inputs and therefore must be identically 0. The inputs for which we show that H evaluates to 0 will correspond to outputs of functions f that satisfy the addition theorem at all x, y.

Given any function f satisfying $\forall x, y\ \ f(x+y) = G[f(x), f(y)]$, since $\mathcal{D}$ is associative we have that $f(x+y+z) = G[f(x), f(y+z)] = G[f(x+y), f(z)]$, and so $G[f(x), G[f(y), f(z)]] = G[G[f(x), f(y)], f(z)]$.

Suppose that there exists a solution f of $f(x+y) = G[f(x), f(y)]$ that takes on at least N^3 distinct values $V = \{v_1, \ldots, v_{N^3}\}$. Since $\forall a, b, c \in V$, $H(a, b, c) = 0$, we know that $H(a, b, c) \equiv 0$ [47]. □

4.2 d'Alembert's equation and others

In this section, we show that the robustness of functional equations of the form $\forall x, y\ \ F[f(x-y), f(x+y), f(x), f(y)] = 0$, is not limited to addition theorems by showing that when the domain $\mathcal{D}$ is a finite group, and the range $\mathcal{R} = \mathcal{T}$ is a field containing 2, d'Alembert's equation $\forall x, y\ \ f(x+y) + f(x-y) = 2f(x)f(y)$ is a robust property on $\mathcal{G} = \{f | Pr_{x \in \mathcal{D}}[f(x) = 0] \leq$

$1/20\}$. Since membership in $\mathcal{G}$ is easy to test, these robustness results lead to self-testers as described later. The techniques in this section can also be used to show that the equations $\forall x, y \;\; f(x+y)+f(x-y) = 2[f(x)+f(y)]$ and $\forall x, y \;\; f(x+y)+f(x-y) = 2f(x)$ are robust over $\mathcal{G}$.

This result does not allow us to test functions that are in $\mathcal{F}$ but not in $\mathcal{G}$, such as the 0-function. For carefully chosen domains, other functions that are solutions to these functional equations (see Figure 2) can also take the value 0 on more than $1/20$ fraction of the domain: For example, $\cos x$ takes the value 0 on half of the domain $\mathcal{D} = \{i \cdot \pi/2 | 0 \leq i \leq 3\}$. The result can still be used to construct self-testers for functions satisfying D'Alembert's equation. We discuss this further in Section 5.

For the following three robustness results, let $\mathcal{N}^{\text{d'Alembert}} = \{(x, y, x+y, x-y) | x, y \in \mathcal{D}\}$.

Theorem 18 *Let $F(x_1, x_2, x_3, x_4) = 2f(x_1) \cdot f(x_2) - f(x_3) - f(x_4)$. Let $\mathcal{F}$ be the function family characterized by $(F, \mathcal{N}^{\text{d'Alembert}})$. Then for $\delta \geq \frac{1}{80}$, $(F, \mathcal{N}^{\text{d'Alembert}})$ is a $(4\delta, \delta)$-robust characterization of $\mathcal{F}$ in $\mathcal{G}$. In particular, if $Pr_{x,y}[f(x+y)+f(x-y) = 2f(x)f(y)] \geq 1-\delta \geq \frac{79}{80}$ and $f \in \mathcal{G}$, then the function $g(x) \equiv \text{maj}_{y \in \mathcal{D},\ f(y) \neq 0}\{\frac{f(x+y)+f(x-y)}{2f(y)}\}$ satisfies (1) $Pr_x[f(x) = g(x)] \geq 1 - 4\delta$ and (2) $\forall x, y \;\; g(x+y)+g(x-y) = 2g(x)g(y)$.*

Theorem 19 *Let $F(x_1, x_2, x_3, x_4) = 2f(x_1)+2f(x_2)-f(x_3)-f(x_4)$. Let $\mathcal{F}$ be the function family characterized by $(F, \mathcal{N}^{\text{d'Alembert}})$. Then for $\delta \geq \frac{1}{80}$, $(F, \mathcal{N}^{\text{d'Alembert}})$ is a $(4\delta, \delta)$-robust characterization of $\mathcal{F}$ in $\mathcal{G}$. In particular, if $Pr_{x,y}[f(x+y)+f(x-y) = 2(f(x)+f(y))] \geq 1-\delta \geq \frac{79}{80}$ and $f \in \mathcal{G}$, then the function $g(x) \equiv \text{maj}_{y \in \mathcal{D},\ f(y) \neq 0}\{\frac{f(x+y)+f(x-y)}{2} - f(y)\}$ satisfies (1) $Pr_x[f(x) = g(x)] \geq 1 - 4\delta$ and (2) $\forall x, y \;\; g(x+y)+g(x-y) = 2(g(x)+g(y))$.*

Theorem 20 *Let $F(x_1, x_2, x_3, x_4) = 2f(x_1) - f(x_3) - f(x_4)$. Let $\mathcal{F}$ be the function family characterized by $(F, \mathcal{N}^{\text{d'Alembert}})$. Then for $\delta \geq \frac{1}{80}$, $(F, \mathcal{N}^{\text{d'Alembert}})$ is a $(4\delta, \delta)$-robust characterization of $\mathcal{F}$ in $\mathcal{G}$. In particular, if $Pr_{x,y}[f(x+y)+f(x-y) = 2f(x)] \geq 1-\delta \geq \frac{79}{80}$ and $f \in \mathcal{G}$, then the function $g(x) \equiv \text{maj}_{y \in \mathcal{D},\ f(y) \neq 0}\{\frac{f(x+y)+f(x-y)}{2}\}$ satisfies (1) $Pr_x[f(x) = g(x)] \geq 1 - 4\delta$ and (2) $\forall x, y \;\; g(x+y)+g(x-y) = 2g(x)$.*

The proofs in this section are similar in flavor to the proofs of the robustness of the addition theorems, but since the functional equation is defined on inputs that are related in different ways, we have to take advantage of different aspects of the structure of their relationship in order to get the desired results. The proofs of all three theorems follow the same outline. In the following, we give the proof of Theorem 18.

Proof: [of Theorem 18] Using techniques identical to those in Lemma 14, we have:

Lemma 21 *g and f agree on more than $1 - 4\delta$ fraction of the inputs from $\mathcal{D}$.*

Lemma 22 *For all x, $\Pr_y[g(x) = \frac{f(x+y)+f(x-y)}{2f(y)}] \geq 1 - \delta'$ where $\delta' = 4\delta + 2 \cdot \frac{1}{20}$.*

Proof:

$\Pr_{y,z}[f(y) \neq 0 \; and \; f(z) \neq 0 \; and$
$\quad 2f(z)(f(x+y)+f(x-y))$

$$\begin{aligned}
&= (f(x+y+z)+f(x+y-z)) \\
&\quad +(f(x-y-z)+f(x-y+z)) \\
&= (f(x+y+z)+f(x-y+z)) \\
&\quad +(f(x-y-z)+f(x+y-z)) \\
&= 2f(y)(f(x+z)+f(x-z))] \\
&\geq 1-4\delta-2\cdot\tfrac{1}{20}
\end{aligned}$$

$f(y) = 0$ or $f(z) = 0$ with probability at most $2 \cdot \frac{1}{20}$. The first and third equality each hold with probability $1-2\delta$ by our assumption on f and since all the references to f are uniformly distributed in $\mathcal{D}$. The second equality always holds. If $f(y), f(z)$ are both nonzero and all equalities hold, then $\frac{f(x+y)+f(x-y)}{2f(y)} = \frac{f(x+z)+f(x-z)}{2f(z)}$. The lemma now follows from the fact that the probability that the same object is drawn twice from a set in two independent trials lower bounds the probability of drawing the most likely object in one trial. □

Finally, we can prove that g satisfies d'Alembert's equation everywhere:

Lemma 23 *For all x, y, $2g(x)g(y) = g(x+y)+g(x-y)$.*

Proof:
$\Pr_z[f(z) \neq 0 \; and$
$\quad f(z) \cdot (g(x+y)+g(x-y))$

$$\begin{aligned}
&= \frac{f(x+y+z)+f(x+y-z)+f(x-y+z)+f(x-y-z)}{2} \\
&= \frac{2g(x)f(y+z)+2g(x)f(y-z)}{2} \\
&= 2g(x)\cdot g(y)\cdot f(z)] \\
&> 1-5\delta'-\tfrac{1}{20} > 0
\end{aligned}$$

$f(z) = 0$ with probability at most $\frac{1}{20}$. By Lemma 22, the first, second and third equalities hold with probability $1-2\delta'$, $1-2\delta'$ and $1-\delta'$ respectively. If $f(z) \neq 0$ and all equalities hold then $2g(x)g(y) = g(x+y)+g(x-y)$. Since the statement is independent of z and holds with positive probability, it must hold with probability 1. □ □

5 Self-testing/correcting from functional equations

We give informal definitions of self-testers and self-correctors. Formal definitions are given in [19]. An (ϵ_1, ϵ_2)-*self-tester* $(0 \leq \epsilon_1 < \epsilon_2)$ for f on D must fail any program that is not $(1-\epsilon_2)$-close to f on D, and must pass any program that is $(1-\epsilon_1)$-close to f on D (the behavior of the tester is not specified for programs that are $(1-\epsilon_2)$-close but not $(1-\epsilon_1)$-close to f). The tester should satisfy these conditions with error probability at most β, where β is a confidence parameter input by the user. For simplicity, we assume that $\epsilon_1 = 0$, and we drop that parameter from our claims. An ϵ-*self-corrector* for f on D is an algorithm C that uses program P as a black box, such that for every $x \in D$ and β, $\Pr[C^P(x) = f(x)] \geq 1-\beta$, for *every* P which is $(1-\epsilon)$-close to f on D. Furthermore, all require only a small multiplicative

overhead over the running time of P and are different, simpler and faster than any correct program for f in a precise sense defined in [18]. Checkers can be constructed by finding both a self-tester and a self-corrector for the function [19].

In this section, we give self-correctors and self-testers that are based on the class of functional equations of the form $F[f(x-y), f(x+y), f(x), f(y)] = 0$. We will use the robustness theorems proved earlier for the self-testers, but not for the self-correctors. We refer to the function computed by the program as P, and the correct function as $f : \mathcal{D} \rightarrow \mathcal{R}$. For purposes of exposition, we assume that $\mathcal{D}$ is a finite group. All results can be extended to rational domains of the form $\mathcal{D}_{p,s} = \{\frac{i}{s} || i| \leq p\}$ using known techniques from [31][37] (see Section 6). We assume that $\mathcal{R}$ is a (possibly infinite) abelian group.

5.1 Self-correctors

The following self-corrector works for any function satisfying $\forall x, y \;\; f(x) = G[f(x-y), f(x+y), f(y)]$. This includes functions satisfying an addition theorem of the form $f(x+y) = G[f(x), f(y)]$, since letting $z = x+y$, $f(z) = G[f(z-y), f(y)]$. Self-correctors for functions that are not solvable for $f(x)$, but are solvable for another of $f(x-y), f(x+y), f(y)$ can be similarly constructed.

Program Self-Correct(x, β)

$N \leftarrow O(\ln(1/\beta))$
Do for $m = 1, \ldots, N$
  Pick $y \in_R \mathcal{D}$
  $answer_m \leftarrow G[P(x-y), P(x+y), P(y)]$
Output the most common answer in $\{answer_m : m = 1, \ldots, N\}$

Theorem 24 *Given $\mathcal{D}$ a finite group, and P and f functions over domain $\mathcal{D}$. If P is $1/12$-close to f over $\mathcal{D}$, then $\forall x$, Pr[Self-Correct$(x, \beta) = f(x)] \geq 1 - \beta$.*

The proof of this theorem follows the format in [19] and is based on the fact that since calls to P are made on uniformly distributed inputs in $\mathcal{D}$, at each iteration, all calls are answered correctly by P with probability at least $3/4$.

The existence of an ϵ-self-corrector for a class of functions $\mathcal{F}$ trivially implies that for any two functions $f_1, f_2 \in \mathcal{F}$, the quantity $Pr_x[f_1(x) \neq f_2(x)]$, or the *distance* between f_1 and f_2, must be large. Thus, the existence of self-correctors for $\mathcal{F}$ implies that the functions in $\mathcal{F}$ can be thought of as a collection of codewords with large distance.

5.2 Self-testers

In this section we show self-testers based on robust functional equations of the form $F[f(x-y), f(x+y), f(x), f(y)] = 0$. In all of our examples only a constant number of additions and multiplications are required to perform a test. Furthermore, only a constant number of tests need to be performed. It often happens that more than one functional equation can be used to specify a function family; the user can determine which of the robust functional equations is best to use for testing based on criteria such as efficiency and ease of programming.

When a family of functions satisfies the property, equality testing must be done to determine that the program is computing the correct function within the family. Though equality testing is often easier than the original testing task, it may still be inefficient, as in the case of multivariate polynomials [38]. For the functions considered in this paper, the problem of equality testing can be solved efficiently. We assume that the function values are given at a constant number of inputs, such that these values in conjuction with the property F are enough to completely specify the function. For example, for functions satisfying addition theorems, the function values at 0 and all generators of the group suffice to completely specify the function. In particular, if the group is cyclic and generated by 1, only $f(0)$ and $f(1)$ are required since $f(x+1) = G[f(x), f(1)]$. Similarly, over $\mathcal{D}_{p,s}$ it is enough to specify the function at $0, \frac{1}{s}, \frac{-1}{s}$. It is often the case that there are certain inputs at which the function is much easier to compute, and that these inputs can be used for the equality testing.

It may happen that the functional equation over the reals characterizes a different family of functions than same functional equation over $\mathcal{D}_{p,s}$. For example, suppose we are given $\mathcal{F}$, a set of functions that is a solution to the functional equation over the reals. The set of functions that we are interested in testing is $\mathcal{F}' = \{g \mid g : \mathcal{D}_{p,s} \rightarrow \mathcal{R}, \exists f \in \mathcal{F}$ such that $\forall x \in \mathcal{D}_{p,s}, f(x) = g(x)\}$, the set of functions that are restrictions of functions in $\mathcal{F}$ to the domain $\mathcal{D}_{p,s}$. Consider also $\mathcal{F}''$, the solutions to the functional equation over $\mathcal{D}_{p,s}$. Then, since functions in $\mathcal{F}''$ must satisfy a subset of the constraints satisfied by $\mathcal{F}'$, $\mathcal{F}' \subseteq \mathcal{F}''$. It can happen that $\mathcal{F}'$ is a proper subset of $\mathcal{F}''$. Nevertheless, to test that a program purporting to compute function $f \in \mathcal{F}'$ is correct, the property test determines whether the program agrees on most inputs with some function g in $\mathcal{F}''$, and the equality test will then determine that $g = f$ as long as it is given the correct values of f at the inputs required for the equality test.

We concentrate on functions that can be tested by testers of the form given below. In the following, δ_0 is the maximum δ for which the functional equation is robust (see Theorems 13, 18, 19, 20), and the function values specifying f are given as a list $(x_i, y_i), 0 \leq i \leq c$ where $y_i = f(x_i)$.

Program Self-Test$((x_0, y_0), (x_1, y_1), \ldots, (x_c, y_c), \delta_0, \beta)$

$N \leftarrow O(\max\{\frac{1}{\delta_0}, 24\} \ln(2/\beta))$
Do for $m = 1, \ldots, N$ {*Property Test* }
  Pick $u, v \in_R \mathcal{D}$
  if $F[P(u-v), P(u+v), P(u), P(v)] \neq 0$ output FAIL and halt
Do for $i = 1, \ldots, c$ {*Equality Test* }
  If self-correct$(x_i, \beta/c) \neq y_i$ output FAIL and halt
Output PASS

Theorem 25 *Given domain $\mathcal{D}$ a finite group, range $\mathcal{R}$ an abelian group, and functions P and f mapping $\mathcal{D}$ to $\mathcal{R}$. If F is $(2\delta, \delta)$-robust over domain $\mathcal{D}$, and if P is not $(1/12)$-close to f on $\mathcal{D}$, then Pr[Self-Test(x, β) = FAIL] $\geq 1 - \beta$. If $P \equiv f$, then Self-Test outputs PASS. Thus, Self-Test is a $\frac{1}{12}$-self-testing program for f on $\mathcal{D}$.*

The proof of the theorem is based on the robustness of F, which tells us that if there is no function g such that (1) g is usually equal to P and (2) g satisfies the property everywhere,

then P is reasonably likely to fail the test. Furthermore, if there is such a function g, the equality tests are likely to fail unless $g(x_0) = f(x_0), \ldots, g(x_c) = f(x_c)$ which ensures that $g \equiv f$. Thus P fails unless it is usually equal to f. It is easy to see that by altering the choice of N, one can construct ϵ-self-testers for any $\epsilon < 1/12$.

The above self-tester is not sufficient for testing functions using d'Alembert's equation, since we have proved its robustness only under the condition that the function P is 0 on $\leq 1/20$ of the inputs. To fix this, one may use an algorithm that depends on the fraction of inputs on which f takes the value 0. The solution to d'Alembert's equation over $\Re$ is all functions of the form $0, \cos Ax, \cosh Ax$. The 0 function is trivial to test. The $\cosh Ax$ functions are never 0 so a program purporting to compute $\cosh Ax$ can be first tested to ensure that it does not output 0 on more than 1/20 of the domain, and then the above tester may be used. The $\cos Ax$ functions are 0 only at odd multiples of $\pi/(2A)$. If the odd multiples of $\pi/(2A)$ constitute at most 1/20 of the domain, then a similar procedure to the one for $\cosh Ax$ may be used. Otherwise one can use a different functional equation for which $\cos Ax$ is a solution. Alternatively, suppose the program can be modified to compute over a larger domain $\hat{\mathcal{D}}$ which contains $\mathcal{D}$ such that $f(x) = 0$ on at most 1/20 of $\hat{\mathcal{D}}$. (For example, if $f(x) = \cos x, \mathcal{D} = \{i \cdot \pi/2 | 0 \leq i \leq p-1\}$, then choose $\hat{\mathcal{D}} = \{i \cdot \pi/20 | 0 \leq i \leq 10p-1\}$.) Then one can test the program over $\hat{\mathcal{D}}$. If the program passes the test, it is possible to test the program on $\mathcal{D}$ by using the self-corrector based on D'Alembert's equation to correctly compute f at any input in $\mathcal{D} \subseteq \hat{\mathcal{D}}$.

6 Extensions to Rational Domains

In this section, we show the self-testers and self-correctors that result from extending the results in Section 5 to rational domains. We consider rational domains of the form $\mathcal{D}_{n,s} = \{\frac{i}{s} | \ |i| \leq n\}$.

The theorems follow the same outline as in the finite fields case, but certain additional technical details must be addressed. These technical details are similar to those used in [31][38].

6.1 Self-correctors.

The following self-corrector works for any function satisfying $\forall x, y \ \ f(x) = G[f(x-y), f(x+y), f(y)]$. Self-correctors for functions that are not solvable for $f(x)$, but are solvable for another of $f(x-y), f(x+y), f(y)$ can be similarly constructed.

As in [31], we assume that the program has been tested over a larger domain $\mathcal{D}_{m,s}$, in order to self-correct over the domain $\mathcal{D}_{n,s}$ (this requires the more general definitions of self-correcting given in [31]). It suffices that $m > 12n$.

Program Self-Correct(x, β)

$N \leftarrow O(\ln(1/\beta))$
Do for $i = 1, \ldots, N$
  Pick $y \in_R \mathcal{D}_{m,s}$
  $answer_i \leftarrow G[P(x-y), P(x+y), P(y)]$

Output the most common answer in $\{answer_i : i = 1, \ldots, N\}$

Theorem 26 *Let m, n be such that $m > 12n$. If P is $(1/24)$-close to f over $\mathcal{D}_{m,s}$, then $\forall x \in \mathcal{D}_{n,s}$, $Pr[\text{Self-Correct}(x, \beta) = f(x)] \geq 1 - \beta$.*

The proof of this theorem follows the format in [31].

Proof: [of Theorem 26] By the assumption on P, $P(y)$ is correct with probability at least $1 - \frac{1}{24}$. Two bad events can happen when picking $x + y$: either $x + y$ is not in $\mathcal{D}_{m,s}$, in which case we know nothing about the probability that $P(x + y)$ is correct, or $x + y$ is in $\mathcal{D}_{m,s}$, but happens to be one of the inputs for which P is incorrect. By our choice of m, the first bad situation happens with probability $\leq 1/24$. The second bad situation happens with probability $\leq \frac{1}{24}$. If neither of these happens, then $P(x+y) = f(x+y)$. The same argument can be made for $P(x - y)$. Thus, at each iteration, $answer_i = f(x)$ with probability at least $1 - 2\frac{1}{24} - 3\frac{1}{24} > 3/4$. □

6.2 Self-testers.

The following is a self-tester for addition theorems over the rational domain $\mathcal{D}_{m,s}$. We test the program over a larger domain $\mathcal{D}_{p,s}$ in order to certify that it is usually correct over $\mathcal{D}_{m,s}$. It suffices that $p > 11m$. As in Section 5.2, we assume the function values specifying f are given as a list of pairs $(x_i, y_i), 0 \leq i \leq c$ where $y_i = f(x_i)$. In addition we assume that $x_i \in \mathcal{D}_{n,s}$ for all i.

The self-tester is based on finding a neighborhood $\mathcal{N}^{\text{add}'}{}_{\mathcal{D}_{p,s}}$ such that $(\mathcal{D}_{m,s}, F_{\mathcal{D}_{p,s},\Re,\Re}, \mathcal{N}^{\text{add}'}{}_{\mathcal{D}_{p,s}})$ is an $(\epsilon, \epsilon/2)$-robust characterization.

Program Self-Test$((x_0, y_0), (x_1, y_1), \ldots, (x_c, y_c), n, m, \delta_0, \beta)$

$N \leftarrow O(\max\{\frac{4}{\delta_0}, 48\} \ln(2/\beta))$
{ *Property Test* }
Do for $m = 1, \ldots, N$
  Choose $i \in \{1, 2, 3, 4\}$
  If $i = 1$ then $\{x_1 \leftarrow x, x_2 \leftarrow x - y, x_3 \leftarrow y\}$
   Pick $x \in_R \mathcal{D}_{m,s}$ and $y \in_R \mathcal{D}_{p,s}$
     if $P(x) \neq G[P(x - y), P(y)]$ output FAIL and halt
  Else if $i = 2$ then $\{x_1 \leftarrow x, x_2 \leftarrow x - y, x_3 \leftarrow y\}$
   Pick $x, y \in_R \mathcal{D}_{p,s}$
     if $P(x) \neq G[P(x - y), P(y)]$ output FAIL and halt
  Else if $i = 3$ then $\{x_1 \leftarrow x + y, x_2 \leftarrow x, x_3 \leftarrow y\}$
   Pick $x, y \in_R \mathcal{D}_{p,s}$
     if $P(x + y) \neq G[P(x), P(y)]$ output FAIL and halt
  Else $\{i = 4\}$ $\{x_1 \leftarrow x, x_2 \leftarrow y, x_3 \leftarrow x - y\}$
   Pick $x, y \in_R \mathcal{D}_{p,s}$
     if $P(x) \neq G[P(y), P(x - y)]$ output FAIL and halt
{ *Equality Test* }
Do for $i = 1, \ldots, c$
  If self-correct$(x_i, \beta/c) \neq y_i$ output FAIL and halt
Output PASS

We have the following theorem:

Theorem 27 *Let p, m, n be such that $p > 11m$ and $m > 12n$. Let the function values (x_i, y_i) specifying f be such that $x_i \in \mathcal{D}_{n,s}$. If P is not $1/24$-close to f on $\mathcal{D}_{m,s}$, then $Pr[\text{Self-Test}(x, \beta) = \text{FAIL}] \geq 1 - \beta$. If $P \equiv f$, then Self-Test outputs PASS. Thus, Self-Test is an $1/24$-self-testing program for f on $\mathcal{D}_{m,s}$.*

In order to show the above theorem, we need the following to show that the addition theorems are robust properties over rational domains.

Lemma 28 *If (1) $Pr_{x \in \mathcal{D}_{m,s}, y \in \mathcal{D}_{p,s}}[P(x+y) = G[P(x), P(y)]] \geq 1 - \delta$, (2) $Pr_{x,y \in \mathcal{D}_{p,s}}[P(x) = G[P(x-y), P(y)]] \geq 1-\delta$, (3) $Pr_{x,y \in \mathcal{D}_{p,s}}[P(x+y) = G[P(x), P(y)]] \geq 1-\delta$, (4) $Pr_{x,y \in \mathcal{D}_{p,s}}[P(x) = G[P(y), P(x-y)]] \geq 1 - \delta$, for $\delta < \frac{1}{48}$, then there exists a function g such that*

1. $Pr_{x \in \mathcal{D}_{m,s}}[P(x) = g(x)] \geq 1 - 2\delta = 1 - \frac{1}{24}$
2. $\forall x, y \in \mathcal{D}_{m,s} \quad g(x+y) = G[g(x), g(y)]$.

Let $\mathcal{N}^{\text{add}'}$ be the multiset such that picking random $(x_1, x_2, x_3) \in \mathcal{N}^{\text{add}'}$ is the same as picking inputs from the above distribution. If the functional equation is satisfied with probability at least $1-\delta/4$ when neighborhoods are chosen from $\mathcal{N}^{\text{add}'}$, then each of the four conditions of the theorem are met. Thus we have the the following theorem:

Theorem 29 *Let $\mathcal{R} = \mathcal{T}$ be a group. Let G be such that G satisfies $\forall a, b, c \in \mathcal{R} \;\; G[a, G[b, c]] = G[G[a, b], c]$. Let $F(x, y) = P(x+y) - G[P(x), P(y)]$. Then for all $\delta < 1/48$, $(\mathcal{D}_{m,s}, F_{\mathcal{D}_{p,s}, \mathcal{R}, \mathcal{T}}, \mathcal{N}^{\text{add}'}{}_{\mathcal{D}_{p,s}})$ is $(2\delta, \delta/4)$-robust.*

Proof: [of Lemma 28] Define $g(x)$ to be $\text{maj}_{z \in \mathcal{D}_{p,s}}\{G(P(x-z), P(z))\}$.

The following lemma follows from the first condition on P and a counting argument.

Lemma 30 *g and P agree on more than $1 - 2\delta$ fraction of the inputs from $\mathcal{D}_{m,s}$.*

For the following lemmas, set $\gamma = \frac{m}{2p}$.

Lemma 31 *For all $x \in \mathcal{D}_{2m,s}$, $\text{Pr}_{z \in \mathcal{D}_{p,s}}[g(x) = G[P(x-z), P(z)]] \geq 1-\delta'$ where $\delta' = 2\delta + 2\gamma$.*

Proof: For $x \in \mathcal{D}_{2m,s}$, $Pr_{y \in \mathcal{D}_{p,s}}[x + y \in \mathcal{D}_{p,s}] \geq 1 - \gamma$. Thus,
$\text{Pr}_{y,w \in \mathcal{D}_{p,s}}[G[P(x-y), P(y)]$

$$\begin{aligned} &= G[G[P(x-w), P(w-y)], P(y)] \\ &= G[P(x-w), G[P(w-y), P(y)]] \\ &= G[P(x-w), P(w)]] \\ &\geq 1 - 2\delta - 2\gamma \end{aligned}$$

By the fourth condition on P, the first equality holds with probability $1 - \delta - 2\gamma$. By the second condition on P, the third equality holds with probability $1 - \delta$. The second equality always holds.

The lemma now follows from the observation that the probability that the same object is drawn twice from a set in two independent trials lower bounds the probability of drawing the most likely object in one trial. □

Finally, we prove that g satisfies the addition theorems everywhere:

Lemma 32 *For all* $x, y \in \mathcal{D}_{m,s}$, $g(x+y) = G[g(x), g(y)]$.

Proof:
$\Pr_{u,v \in \mathcal{D}_{p,s}}[G[g(x), g(y)]$

$$\begin{aligned}
&= G[G[P(u), P(x-u)], G[P(v), P(y-v)]] \\
&= G[P(u), G[P(x-u), G[P(v), P(y-v)]]] \\
&= G[P(u), G[G[P(x-u), P(v)], P(y-v)]] \\
&= G[P(u), G[P(x-u+v), P(y-v)]] \\
&= G[P(u), P(x+y-u)] \\
&= g(x+y)] \\
&\geq 1 - 3\delta' - 2\delta - 3\gamma = 1 - 8\delta - 9\gamma > 0
\end{aligned}$$

By Lemma 31, the first equality holds with probability $1-2\delta'$ and the last equality holds with probability $1-\delta'$ (since $x+y \in \mathcal{D}_{2m,s}$). By the third assumption on P, the fourth equality holds with probability $1-\delta-\gamma$. By the second assumption on P, the fifth equality holds with probability $1-\delta-2\gamma$. The other equalities always hold, due to the structure of G.

Since the statement is independent of u, v and holds with positive probability, it must hold with probability 1. □

□[Lemma 28]

6.3 An example: Testing the cosh function

In this subsection we will illustrate how to apply the above techniques to construct a self-tester and self-corrector for a particular function, namely the cosh function, over a given domain. Suppose that one would like to reliably use a program that purports to compute the cosh function over the domain $\mathcal{D}_{2^k,2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^k\}$ (the numbers between -1 and 1 with k bits of precision). Assume that the correct values of $\cosh 0, \cosh \frac{-1}{2^k}, \cosh \frac{1}{2^k}$ are given and that the program purports to compute cosh over the larger domain $\mathcal{D}_{2^{k+8},2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^{k+8}\}$. Recall that cosh is one of the solutions to the functional equation $\forall x, y \ f(x+y) = f(x)f(y) + \sqrt{f(x)^2-1}\sqrt{f(y)^2-1}$. Furthermore, $\cosh(x)$ is the only solution to the functional equation that agrees with the given values of $\cosh 0, \cosh \frac{-1}{2^k}, \cosh \frac{1}{2^k}$, since $f(0), f(1/2^k), f(-1/2^k)$ determine the values of f over the whole domain $\mathcal{D}_{2^{k+8},2^k}$ via $f((i+1)/2^k) = f(1/2^k)f(i/2^k) + \sqrt{f(1/2^k)^2-1}\sqrt{f(i/2^k)^2-1}$ and $f((i-1)/2^k) = f(-1/2^k)f(i/2^k) + \sqrt{f(-1/2^k)^2-1}\sqrt{f(i/2^k)^2-1}$.[8]

One should first test the program over the domain $\mathcal{D}_{2^{k+4},2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^{k+4}\}$, using the tester given in Subsection 6.2 (with $\delta_0 = 1/48, s = 2^k, m = 2^{k+4}, p = 2^{k+8}, n = 2^k$ so that $p > 11m$ and $m > 12n$). By Theorem 27 (which in turn uses Theorem 29), if the program is always correct on domain $\mathcal{D}_{2^{k+8},2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^{k+8}\}$ the tester will output PASS, and if the program is incorrect on greater than $1/24$ fraction of the domain $\mathcal{D}_{2^{k+4},2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^{k+4}\}$, then the tester will output FAIL. If the program is incorrect on less than

[8] Self-testers and self-correctors for $\cosh x$ can be constructed via other functional equations that $\cosh x$ satisfies as long as they are shown to be robust and equality testing is possible.

1/24 fraction of the domain $\mathcal{D}_{2^{k+4},2^k}$, one can use the self-corrector in Subsection 6.1 (with $s = 2^k, m = 2^{k+4}, n = 2^k$ so that $m > 12n$) in order to compute the correct value of $\cosh x$ for all $x \in \mathcal{D}_{2^k,2^k} = \{\frac{i}{2^k} | \ |i| \leq 2^k\}$. The correctness of the self-corrector is guaranteed by Theorem 26.

7 Conclusions and directions for further research

We have studied the question of when functions characterized by functional equations of the form $\forall x, y \quad F[f(x-y), f(x+y), f(x), f(y)] = 0$ are robust. However, we still do not have a complete answer to this question. Even for addition theorems $\forall x, y \quad f(x+y) = G[f(x), f(y)]$, we do not know what happens when G does not satisfy $G[a, G[b,c]] = G[G[a,b],c]\ \forall a, b, c$. More generally, many other general types of functional equations have been identified, including those on multivariate functions and systems of functional equations, but we do not know which ones are robust. Given a functional equation, is there an (efficient) algorithm to determine whether or not it is robust? Is it the case that any property that leads to a self-corrector is robust?

Systems of functional equations can be used to define more than one unknown function by their joint properties. For example, Pexider's equations are $f(x+y) = g(x)+h(y)$, $f(x+y) = g(x)h(y)$, $f(xy) = g(x)+h(y)$, and $f(xy) = g(x)h(y)$. which are generalizations of Cauchy's original functional equations. These equations have applications to the library setting [19], where programs for several functions can be used to self-test and self-correct each other, as long as none of the answers are *a priori* assumed to be correct. The library setting has been used to find checkers that are significantly more efficient for functions such as determinant and rank. Are there any other examples of functions where their mutual properties lead to more efficient testers?

It is important to find methods to extend all robustness results to the case of real valued computation as in [31], [5] [27]. One point of difficulty is that in real valued computation, none of the functional equations will be satisfied exactly, even when the program is giving very good approximations to the correct answers. Thus, the area of functional inequalities, which is the investigation of which families of functions satisfy inequalities such as $|f(x+y) - f(x) - f(y)| \leq \epsilon$, directly applies to this setting. Much of the work in [31] [5] [27] has been in relating the class of functions that are solutions to functional inequalities to the class of functions that are solutions of the corresponding functional equations. Several functional inequalities have been shown to be robust in [31], [5], [27]. Other related results used for testing matrix multiplication, linear system solution, matrix inversion and determinant computation are in [5].

Acknowledgements We wish to thank Mike Luby for initial conversations that eventually led to the idea behind this work, and Nati Linial for directing us to the area of functional equations as well as to many of the references. We thank Madhu Sudan for many interesting conversations, comments and insights on this work. We also thank Richard Zippel for his technical advice on the zero testing of algebraic functions, and for guiding us through the literature regarding the scope of the theorems in this paper. We thank S Ravi Kumar and Funda Ergün for discussions regarding the definitions in this paper and comments on the

writeup. We thank S Ravi Kumar and D. Sivakumar for pointing out an error. We thank Lucian Bebchuk, Ran Canetti, Oded Goldreich, Diane Hernek, Sandy Irani, Mike Luby, Dana Ron and especially the anonymous referee for greatly improving the presentation of this paper.

References

[1] Abadi, M., Feigenbaum, J., Kilian, J., "On Hiding Information from an Oracle", *Journal of Computer and System Sciences*, Vol. 39, No. 1, August 1989, pp. 29-50.

[2] L. Adleman, M. Huang, and K. Kompella. Efficient checkers for number-theoretic computations. To appear in *Information and Computation*.

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

[4] J. Aczél, and J. Dhombres. *Functional Equations in Several Variables*, Cambridge University Press, 1989.

[5] S. Ar, M. Blum, B. Codenotti, P. Gemmell. Checking approximate computations over the reals. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 786-795, 1993.

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

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

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

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

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

[11] Bellare, M., Coppersmith, D., Hastad, J., Kiwi, M., Sudan, M., "Linearity testing in characteristic two", *Proc. 36nd Annual Symposium on Foundations of Computer Science,*, pp.434–441.

[12] Bellare, M., Goldreich, O., Sudan, M., "Free bits and non-approximability", *Proc. 36nd Annual Symposium on Foundations of Computer Science,*.

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

[14] Bellare, M., Sudan, M., "Improved non-approximability results", *Proc. of the ACM Symposium on the Theory of Computing*, pp. 184-193, 1994.

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

[16] M. Blum, B. Codenotti, P. Gemmell, T. Shahoumian. Self-correcting for function fields of finite transcendental degree. Manuscript.

[17] Blum, M., Evans, W., Gemmell, P., Kannan, S., Naor, M., "Checking the Correctness of Memories" *Proc. 32nd Annual Symposium on Foundations of Computer Science*, pp. 90-99, 1991.

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

[19] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Sys. Sci.* Vol. 47, No. 3, December 1993. Preliminary version appears in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, 1990.

[20] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM J. on Computing*, Vol. 13, 1984, pp. 850-864, extended abstract in *FOCS 1982.*

[21] M. Blum and H. Wasserman. Program result-checking: A theory of testing meets a test of theory. In *Proc. 35th FOCS*, pp. 382–392, 1994.

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

[23] R. Cleve, M. Luby. A Note on Self-Testing/Correcting Methods for Trigonometric Functions, International Computer Science Institute Technical Report TR-90-032, July, 1990.

[24] Cody, W.J., "Performance evaluation of programs related to the real gamma function", *ACM Transactions on Mathematical Software*, Vol. 17, No. 1, March 1991, pp. 46-54.

[25] Cody, W.J., Stoltz, L., "The use of Taylor series to test accuracy of function programs", *ACM Transactions on Mathematical Software*, Vol. 17, No. 1, March 1991, pp. 55-63.

[26] D. Coppersmith. Manuscript, December 1989. Result described in [19].

[27] F. Ergün, S R. Kumar, R. Rubinfeld. Approximate Checking of Polynomials and Functional Equations. *Proceedings of the 37nd IEEE Symposium on Foundations of Computer Science*, 1996.

[28] Funda Ergün, S Ravi Kumar, and D Sivakumar, "Self-testing without the generator bottleneck." Submitted to SIAM Journal on Computing.

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

[30] J. Feigenbaum, L. Fortnow. On the Random-Self-Reducibility of Complete Sets. SIAM J. Computing, 1993.

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

[32] M. Klawe. Limitations on Explicit Constructions of Expanding Graphs. in *SIAM J. Computing*, Vol. 13, No. 1, pp. 156-166, February 1984.

[33] S Ravi Kumar and D Sivakumar, "Efficient self-testing/self-correction of linear recurrences," *Proc. 37th IEEE Foundations of Computer Science*, pages 602–611, October 1996.

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

[35] Micali, S. Computationally-Sound Proofs. *Proc. 35nd Annual Symposium on Foundations of Computer Science.*

[36] Polischuk, A., Spielman, D., "Nearly linear size holographic proofs", *Proc. 26th ACM Symposium on Theory of Computing*, pp. 194-203, 1994.

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

[38] R. Rubinfeld and M. Sudan. Robust Characterizations of Polynomials and their Applications to Program Testing. *SIAM J. of Computing*, 25(2):252–271, 1996.

[39] Schriver, N., Personal communication. February 1990.

[40] M. Sudan. Efficient checking of polynomials and proofs and the hardness of approximation problems. PhD Thesis. U.C. Berkeley, Oct. 1992.

[41] M. Sudan. Personal communications, Summer 1994.

[42] F. Vainstein, Error Detection and Correction in Numerical Computations by Algebraic Methods, Proceedings 9th International Symposium, AAECC-9, New Orleans, LA, 1991, LNCS 539, Springer Verlag.

[43] F. Vainstein, Algebraic Methods in Hardware/Software Testing. PhD Thesis, Boston University, 1993.

[44] F. Vainstein, Low Redundancy Polynomial Checks for Numerical Computation. Applicable Algebra in Engineering, Communication and Complexity, 1995.

[45] F. Vainstein, Self Checking Design Technique for Numerical Computations. Journal of VLSI Design, 1995.

[46] R. Zippel. Zero Testing of Algebraic Functions. *Information Processing Letters*, to appear.

[47] R. Zippel. Effective Polynomial Computation. Kluwer Academic Publishers, 1993.