Exact and Approximate Testing/Correcting of Algebraic Functions: A Survey

Marcos Kiwi* ⋆, Frédéric Magniez† ⋆⋆, and Miklos Santha† ⋆⋆⋆

* Dept. Ing. Matemática, U. Chile & Ctr. Modelamiento Matemático, UMR 2071 UChile–CNRS, Santiago 170–3, Chile.
† CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France.
mkiwi@dim.uchile.cl {magniez,santha}@lri.fr

Abstract. In the late 80's Blum, Luby, Rubinfeld, Kannan et al. pioneered the theory of self–testing as an alternative way of dealing with the problem of software reliability. Over the last decade this theory played a crucial role in the construction of probabilistically checkable proofs and the derivation of hardness of approximation results. Applications in areas like computer vision, machine learning, and self–correcting programs were also established.
In the self–testing problem one is interested in determining (maybe probabilistically) whether a function to which one has oracle access satisfies a given property. We consider the problem of testing algebraic functions and survey over a decade of research in the area. Special emphasis is given to illustrate the scenario where the problem takes place and to the main techniques used in the analysis of tests. A novel aspect of this work is the separation it advocates between the mathematical and algorithmic issues that arise in the theory of self–testing.

1 Introduction

The issue of program (software) reliability is probably just as old as the theory of program design itself. People have spent and continue to spend considerable time on finding bugs in programs. But, the conception of a really satisfying theory for handling this problem remains a hard and elusive goal. Besides professional programmers, users would also like to dispose of tools which could enable them to efficiently address this task. Since they are usually not experts, these tools should ideally be less complicated than the ones used in the programs themselves. The fact that programs are becoming more and more involved obviously presents an

⋆ Gratefully acknowledges the support of Conicyt via Fondecyt No. 1981182 and Fondap in Applied Mathematics, 2000.

⋆⋆ Partially supported by the EC thematic network RAND-APX IST-1999-14036. The participation at the Summer School was founded by the LRI (Orsay) and the IPM (Tehran).

⋆⋆⋆ Partially supported by the EC thematic network RAND-APX IST-1999-14036. The participation at the Summer School was founded by the EGIDE (Paris) and the IPM (Tehran).

additional difficulty. Nonetheless, several approaches have been considered and are used in practice. Each of them have different pros and cons. None is totally satisfactory.

The method of program verification proceeds through mathematical claims and proofs involving the behavior of a program. In principle this method could perfectly achieve the desired task once the program has been proven to behave correctly on all possible inputs. Another advantage of this approach is that the verification takes place only once, before the program is ever executed. Unfortunately, establishing such proofs turns out to be extremely difficult, and in practice it has only been achieved for a few quite simple programs. Also, requiring that programmers express their ideas in mathematically verifiable programs is probably not a realistic expectation. Moreover, there is no protection against errors caused by hardware problems.

Traditional program testing selects a few (sometimes random) inputs, and verifies the program's correctness on these instances. The drawbacks of this approach are fairly obvious. First, there is a priori no reason that the correctness on the chosen instances would imply correctness on instances which were not tested. Second, testing the correctness on the chosen instances usually involves another program which is believed to execute perfectly its task. Clearly, there is some circularity in this reasoning: relying on the correctness of another program is using a tool which is just as powerful as the task that was set to be achieved. Finally, hardware based errors might not be detected until it is too late.

In the late eighties a significantly novel approach, the theory of *program checking* and *self–testing/correcting*, was pioneered by the work of Blum [Blu88], Blum and Kannan [BK89] and Blum, Luby, and Rubinfeld [BLR90]. This theory is meant to address different aspects of the basic problem of program correctness via formal methods by verifying carefully chosen mathematical relationships between the outputs of the program on randomly selected inputs. Specifically, consider the situation where a program P is supposed to compute some function f. A *checker* for f verifies whether the program P computes f on a particular input x; a *self–tester* for f verifies whether the program P is correct on most inputs; and a *self–corrector* for f uses a program P, which is correct on most inputs, to compute f correctly everywhere. All these tasks are supposed to be achieved algorithmically by probabilistic procedures, and the stated requirements should be obtained with high probability. Checkers and self–testers/correctors can only access the program as a black box, and should do something different and simpler than to actually compute the function f.

More than a decade after the birth of this new approach, one can state with relatively high assurance that it has met considerable success both in the theoretical level and in practice. The existence of efficient self–testers was established in the early years of the theory for a variety of mostly algebraic problems, including linear functions and polynomials. These results which were first obtained in the model of exact computations were later partly generalized to more and more complicated (and realistic) models of computations with errors. Self–testers were heavily used in structural complexity, paving the way for the fundamental results characterizing complexity classes via interactive and probabilistically checkable proofs. These results also had remarkable and surprising consequences — they played a crucial role in the derivation of strong non–approximability results for NP–hard optimization problems. In recent years, the theory of self–testing has evolved into what is called today *property testing*, where one has to establish via a few random checks whether an object possesses some given property. Among the many examples one can mention numerous graph properties such as bipartiteness or colorability; monotonicity of functions, or properties of formal languages.

On the practical side, self–testers/correctors were constructed for example for a library of programs computing standard functions in linear algebra [BLR90]. The viability of the approach was also illustrated in the study by Blum and Wassermann [BW97] of the division bug of the first Pentium processors. They showed that this problem could have been detected and corrected by the self–testing/correcting techniques already available at that time. Also, the self–tester of Ergün [Erg95] for the Discrete Fourier Transform is currently used in the software package FFTW for computing reliably fast Fourier transformations [FFT].

In this survey we review the most important results and techniques that arise in self–testing/correcting algebraic functions, but we do not address the subject of checking since the existence of a self–tester/corrector directly implies the existence of a checker. This work also contains some new results about self–correcting, but its main originality lies in the systematic separation it advocates between the purely mathematical and the algorithmic/computational aspects that arise in the theory of self–testing. Also, we do not include any specific computational restriction in our definitions of self–testers/correctors. Instead, we think it is better to give precise statements about the algorithmic performance of the self–testers/correctors constructed. The advantage of this approach is that it allows to independently address different aspects of self–testers/correctors.

This work will be divided into two main parts: the first one deals with exact, and the second with approximate computations. In both models, our basic definition will be for testing function families. In the exact model we first prove a generic theorem for constructing self–testers. This method requires that the family to be tested be characterized by a (property) test possessing two specific properties: continuity and robustness. These properties ensure that the distance of a program from the target function family is close to the probability that the program is rejected by the test, which in turn can be well approximated by standard sampling techniques. After illustrating the method on the benchmark problem of linearity, we address the questions of self–correcting linear functions, and a way to handle the so-called generator bottleneck problem which is often encountered when testing whether a program computes a specific function. Afterwards, we study self–testers for multiplication and polynomials.

The basic treatment of the approximate model will be analogous. The general notion of a computational error term will enable us to carry this out in several models of approximate computing, such as computations with absolute error, with error dependent on the input size, and finally with relative error.

We will emphasize the new concepts and techniques we employ to deal with the increasing difficulties due to the changes in the model. In particular, we will have to address a new issue that arises in approximate testing: stability. This property ensures that a program approximately satisfying a test everywhere is close to a function which exactly satisfies the test everywhere. We formalize here the notion of an approximate self–corrector. In our discussion of approximate self–testers we again address the linearity testing problem in (almost) full detail, whereas for polynomials we mostly simply state the known results. On the other hand, we address in some detail the issue of how to evaluate rapidly the test errors in the case of input size dependent errors. In the case of relative error we discuss why the standard linearity test, which works marvelously well in all previously mentioned scenarios, has to be replaced by a different one.

In the last section of this work we briefly deal with two subjects closely related to self–testing: probabilistically checkable proofs and property testing. Probabilistically checkable proofs heavily use specific self–testing techniques to verify with as few queries as possible whether a function satisfies some pre-specified properties. Property testing applies the testing paradigm to the verification of properties of combinatorial objects like graphs, languages, etc. We conclude this survey by describing the relation between self–testing and property testing and mentioning some recent developments concerning this latter framework.

2 Exact Self–testing

2.1 Introduction to the model

Throughout this section, let D and R be two sets such that D is finite, and let $\mathcal{C}$ be a family of functions from D to R. In the testing problem one is interested in determining, maybe probabilistically, how "close" an oracle function $f : D \to R$ is from an underlying family of functions of interest $\mathcal{F} \subseteq \mathcal{C}$. The function class $\mathcal{F}$ represents a property which one desires f to have. In order to formalize the notion of "closeness," the concept of distance is introduced. Informally, the distance between f and $\mathcal{F}$ is the smallest fraction of values taken by f that need to be changed in order to obtain a function in $\mathcal{F}$. For a formal definition, let $\mathbf{Pr}_{a \in_{\mathcal{D}} A}[E_a]$ denote the probability that the event E_a occurs when a is chosen at random according to the distribution $\mathcal{D}$ over A (typically, $\mathcal{D}$ is omitted when it is the uniform distribution).

Definition 1 (Distance). *Let $P, f \in \mathcal{C}$ be two functions. The* distance between P and f *is*

$$\mathsf{Dist}(P, f) = \mathop{\mathbf{Pr}}_{x \in D}[P(x) \neq f(x)].$$

If $\mathcal{F} \subseteq \mathcal{C}$, then the distance of P from $\mathcal{F}$ *is*

$$\mathsf{Dist}(P, \mathcal{F}) = \mathop{\mathrm{Inf}}_{f \in \mathcal{F}} \mathsf{Dist}(P, f).$$

A self–tester for a function class $\mathcal{F} \subseteq \mathcal{C}$ is a probabilistic oracle program T that can call as a subroutine another program $P \in \mathcal{C}$, i.e., can pass to P an input x and is returned in one step $P(x)$. The goal of T is to ascertain whether P is either close or far away from $\mathcal{F}$. This notion was first formalized in [BLR90].

Definition 2 (Self–tester). *Let $\mathcal{F} \subseteq \mathcal{C}$, and let $0 \le \eta \le \eta' < 1$. An (η, η')–* self–tester *for $\mathcal{F}$ on $\mathcal{C}$ is a probabilistic oracle Turing machine T such that for every $P \in \mathcal{C}$ and for every confidence parameter $0 < \gamma < 1$:*

- *if* $\mathsf{Dist}(P, \mathcal{F}) \le \eta$, *then* $\mathbf{Pr}\left[T^P(\gamma) = \texttt{GOOD}\right] \ge 1 - \gamma$;
- *if* $\mathsf{Dist}(P, \mathcal{F}) > \eta'$, *then* $\mathbf{Pr}\left[T^P(\gamma) = \texttt{BAD}\right] \ge 1 - \gamma$,

where the probabilities are taken over the coin tosses of T.

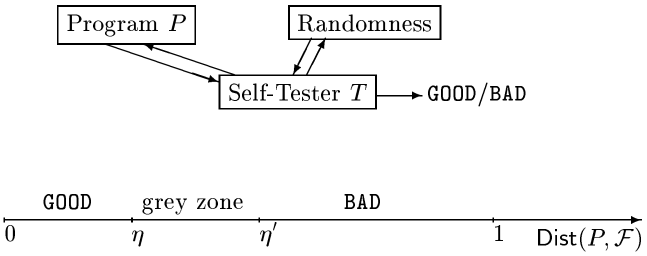

Fig. 1. Definition of an (η, η')–self–tester.

One of the motivations for building self–testers is to make it possible to gain evidence that a program correctly computes a function f on a collection of instances without trying to prove that the program is correct on all possible inputs. However, this raises the question of how to determine that the self–tester is correct. One way around this issue is to ask for the self–tester to be simpler than any correct program for f. Unfortunately simplicity is an aesthetic notion difficult to quantify. Thus, Blum suggested forcing the self–tester to be different from any program computing f in a quantifiable way. This leads to the following definition [Rub90]: *A self–tester T is quantifiably different with respect to $\mathcal{F}$, when for all programs P the incremental time taken by T^P is smaller than the fastest known program for computing a function in $\mathcal{F}$.*[1] Still, this requires the design of equally good self–testers for both efficient and inefficient programs purportedly computing the same function f. Moreover, self–testers are useful in contexts other than program verification, e.g., in the construction of

[1] Ideally, one would prefer that the incremental time be smaller than any correct program for computing functions in $\mathcal{F}$. But, this is too strong a requirement since for many problems of interest no non–trivial lower bound on a correct program's running time is known.

probabilistically checkable proofs where one is more concerned with the query complexity and randomness usage rather than the efficiency of the self–testers. Thus, we simply advocate precisely stating the incremental running time and the operations carried out by the self–testers in order to let the user judge whether the self–tester is useful.

Traditionally, the self–testing literature identifies a test with a self–tester. We do not advocate this practice. We prefer to think of a test as a purely mathematical object and keep it separate from its computational implementation, as proposed in [Kiw96]. This motivates the following:

Definition 3 (Exact test). *An* exact test $(\mathcal{T},\mathcal{C},\mathcal{D})$ *is a set of applications* $\mathcal{T}$ *from* $\mathcal{C}$ *to the set* $\{\texttt{GOOD},\texttt{BAD}\}$ *together with a distribution* $\mathcal{D}$ *over* $\mathcal{T}$*. The exact test* characterizes *the family of functions*

$$\mathsf{char}(\mathcal{T},\mathcal{C},\mathcal{D}) = \{f \in \mathcal{C} \ : \ \mathop{\mathbf{Pr}}_{t\in_{\mathcal{D}}\mathcal{T}}[t(f) = \texttt{GOOD}] = 1\}.$$

The rejection probability *of a function* $P \in \mathcal{C}$ *by the exact test is defined as*

$$\mathsf{Rej}(P,\mathcal{T}) = \mathop{\mathbf{Pr}}_{t\in_{\mathcal{D}}\mathcal{T}}[t(P) = \texttt{BAD}]\,.$$

A probabilistic oracle Turing machine M realizes *the exact test* $\mathcal{T}$ *on* $\mathcal{C}$ *if for all* $P \in \mathcal{C}$,

$$\mathbf{Pr}\left[M^P \ \textit{returns}\ \texttt{BAD}\right] = \mathsf{Rej}(P,\mathcal{T}),$$

where the probability on the left hand side is taken over the coin tosses of the machine M*.*

For the sake of clarity, we specify an exact test via the following mathematically equivalent very high level algorithm:

Exact Test$(P \in \mathcal{C}, \mathcal{T}, \mathcal{D})$
1. Choose an element $t \in \mathcal{T}$ according to $\mathcal{D}$.
2. Reject if $t(P) = \texttt{BAD}$ (otherwise accept).

This notation highlights how to realize an exact test: First, randomly sample from $\mathcal{T}$ according to $\mathcal{D}$ by using only coin tosses and then compute $t(P)$.

In order not to unnecessarily clutter the notation, when referring to an exact test $(\mathcal{T},\mathcal{C},\mathcal{D})$ we henceforth omit $\mathcal{D}$ and assume that it is the uniform distribution over $\mathcal{T}$. Also, if no reference to a particular distribution is given, by a randomly chosen element we mean an element chosen uniformly at random. In addition, when talking about several randomly chosen elements, unless said otherwise, we mean that they are randomly and independently chosen. It is a simple exercise to extend the framework presented here to the case of non–uniform distributions over $\mathcal{T}$. Note however, that if an exact test $\mathcal{T}$ on $\mathcal{C}$ is finite, then in the uniform distribution case it characterizes the family of functions $f \in \mathcal{C}$ such that $t(f) = \texttt{GOOD}$ for every $t \in \mathcal{T}$. We also omit $\mathcal{C}$ if it is clear from context. Finally, it is to be understood that a test accepts when it does not reject.

Computing the distance of a function from a family $\mathcal{F}$ is usually a hard task. On the other hand, the rejection probability of a function by an exact test $\mathcal{T}$ can be easily approximated by standard sampling techniques. Therefore, if an exact test characterizing some function family is such that for every function the rejection probability is close to the distance, then by approximating the rejection probability one can estimate the distance. This allows to probabilistically determine whether the oracle function is close or far away from the function class $\mathcal{F}$ of interest. In other words, one obtains a self–tester for $\mathcal{F}$. The two important properties of an exact test which ensure that this approach succeeds are:

Definition 4 (Continuity & robustness). *Let $\mathcal{T}$ be an exact test on $\mathcal{C}$ characterizing $\mathcal{F}$. Let $0 \leq \eta, \delta < 1$ be constants. Then $\mathcal{T}$ is (η, δ)–*continuous *if for all $P \in \mathcal{C}$,*

$$\mathsf{Dist}(P, \mathcal{F}) \leq \eta \implies \mathsf{Rej}(P, \mathcal{T}) \leq \delta,$$

*and it is (η, δ)–*robust *if for all $P \in \mathcal{C}$,*

$$\mathsf{Rej}(P, \mathcal{T}) \leq \delta \implies \mathsf{Dist}(P, \mathcal{F}) \leq \eta.$$

Thus, proving continuity of an exact test implies upper bounding the rejection probability of the exact test in terms of the relevant distance. On the contrary, to prove robustness one needs to bound the relevant distance in terms of the rejection probability of the exact test. In fact, we advocate explicitly stating these bounds as long as the clarity of the writeup is not compromised.

The importance of continuity and robustness was very early recognized in the self–testing literature. Proving continuity is usually very easy, often people do not even bother stating it explicitly. The term itself was first used by Magniez [Mag00a]. Robustness on the other hand is quite delicate to establish. The term itself was coined and formally defined by Rubinfeld and Sudan in [RS92b] and studied in [Rub94]. Typically, exact tests that are both continuous and robust give rise to self–testers. We now precisely state this claim. The construction of most of the known self–testers are based on it.

Theorem 1 (Generic self–tester). *Let $\mathcal{F} \subseteq \mathcal{C}$ be a function family and let $\mathcal{T}$ be an exact test on $\mathcal{C}$ that is realized by a probabilistic Turing machine M. Let $0 \leq \delta < \delta' < 1$ and $0 \leq \eta \leq \eta' \leq 1$. If $\mathcal{T}$ characterizes $\mathcal{F}$,*

- *is (η, δ)–continuous, and*
- *(η', δ')–robust,*

then there exists an (η, η')–self–tester T for $\mathcal{F}$ on $\mathcal{C}$. Moreover, T performs, for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)\frac{\delta+\delta'}{(\delta-\delta')^2})$ iterations of M, counter increments, comparisons, and binary shifts.

Proof. Let $\delta^* = (\delta + \delta')/2$. The self–tester T repeats N-times the computation of M with program P as oracle. After N repetitions, T computes the fraction err of runs which gave the answer `BAD`. If $err > \delta^*$, then T returns `BAD`, and `GOOD` otherwise. To make the computation of err simple, N is chosen a power of 2. Moreover, N is chosen large enough so that $\mathsf{Rej}(P,\mathcal{T}) \leq \delta$ (respectively $\mathsf{Rej}(P,\mathcal{T}) > \delta'$) implies $err \leq \delta^*$ (respectively $err > \delta^*$) with probability at least $1-\gamma$. Standard Chernoff bound arguments (see e.g. [AS92a, Appendix A]) show that it is sufficient to choose N so that $N = O(\ln(1/\gamma)/\frac{\delta+\delta'}{(\delta-\delta')^2})$. The work performed by the self–tester consists of at most N iterations of M, counter increments, comparisons, and binary shifts.

We now show that T has the claimed properties. First, assume $\mathsf{Dist}(P,\mathcal{F}) \leq \eta$. The continuity of the exact test implies that $\mathsf{Rej}(P,\mathcal{T}) \leq \delta$. Therefore, with probability at least $1-\gamma$ the machine $T^P(\gamma)$ computes $err \leq \delta^*$ and returns `GOOD`. Suppose now that $\mathsf{Dist}(P,\mathcal{F}) > \eta'$. Since the exact test is (η',δ')–robust, $\mathsf{Rej}(P,\mathcal{T}) > \delta'$. Therefore, with probability at least $1-\gamma$ the machine $T^P(\gamma)$ computes $err > \delta^*$ and returns `BAD`. □

A typical way of specifying an exact test is through a functional equation. Indeed, let $\Phi : \mathcal{C} \times \mathcal{N} \to \mathbb{R}$ be a functional where $\mathcal{N} \subseteq \bigcup_{k=1}^{|D|} D^k$. The set $\mathcal{N}$ is called the neighborhood set and each of its members is typically referred to as a neighborhood. The functional Φ *induces* the exact test $\mathcal{T}$ on $\mathcal{C}$ by defining for every $(x_1,\ldots,x_k) \in \mathcal{N}$ the mapping $t_{x_1,\ldots,x_k} : \mathcal{C} \to \{\texttt{GOOD},\texttt{BAD}\}$ as

$$t_{x_1,\ldots,x_k}(P) = \begin{cases} \texttt{GOOD}, & \text{if } \Phi(P,x_1,\ldots,x_k) = 0, \\ \texttt{BAD}, & \text{otherwise}, \end{cases}$$

and letting

$$\mathcal{T} = \{t_{x_1,\ldots,x_k} : (x_1,\ldots,x_k) \in \mathcal{N}\}.$$

The **Exact Test**(P) thus becomes:

Functional Equation Test(P,Φ)
1. Randomly choose $(x_1,\ldots,x_k) \in \mathcal{N}$.
2. Reject if $\Phi(P,x_1,\ldots,x_k) = \texttt{BAD}$.

The family of functions characterized by the induced exact test consists of those functions $f \in \mathcal{C}$ satisfying the following functional equation:

$$\forall (x_1,\ldots,x_k) \in \mathcal{N}, \qquad \Phi(f,x_1,\ldots,x_k) = 0.$$

There might be different functionals characterizing the same collection of functions, and not necessarily all of them give rise to equally appealing exact tests. Indeed, one usually desires that the largest number of values of f that one needs to know in order to compute $\Phi(f,\ldots)$, no matter what f is, be small. If the largest such number is K, the exact test is called *K–local.* For example, the exact test induced by the functional $\Phi(f,x,y) = f(x+y) - f(x) - f(y)$ is 3–local.

Through Theorem 1, functional equations that give rise to exact tests that are both continuous and robust lead to the construction of self–testers.

For the sake of concreteness, we now introduce one of the most famous self–testing problems, one that has become the benchmark throughout the self–testing literature for trying out new techniques, disproving conjectures, etc. —

the so called *linearity testing problem.* In it, one is interested in verifying whether a function P taking values from one finite abelian group G into another such group G' is a group homomorphism. In other words, whether

$$\forall g, g' \in G, \qquad P(g+g') - P(g) - P(g') = 0.$$

This functional equation gives rise to the following functional equation test:

Linearity Test(P)
1. Randomly choose $x, y \in G$.
2. Reject if $P(x+y) - P(x) - P(y) \neq 0$.

The above described exact test was introduced in [BLR90] and is also known as the BLR test. We will now illustrate the concepts introduced so far as well as discuss several important issues that arise in connection to testing by focusing our attention in the study of the **Linearity Test**.

2.2 Linearity self–testing

Let $\mathcal{C}$ denote the collection of functions from G to G', and let $\mathcal{L}$ be the subset of those functions that are homomorphisms. By Theorem 1, in order to come up with a self–tester for $\mathcal{L}$ on $\mathcal{C}$ we need only that the **Linearity Test** is both continuous and robust. As mentioned before, the continuity of an exact test is a property which is rather easy to establish. This is also the case for the **Linearity Test** as shown by the following result from which the $(\eta, 3\eta)$–continuity immediately follows.

Theorem 2. *Let G and G' be two finite abelian groups and let $P, g : G \to G'$ be such that g is a homomorphism. Then*

$$\Pr_{x,y \in G}[P(x+y) - P(x) - P(y) \neq 0] \leq 3 \Pr_{x \in G}[g(x) \neq P(x)].$$

Proof. Observe that $P(x+y) - P(x) - P(y) \neq 0$ implies that $g(x+y) \neq P(x+y)$ or $g(x) \neq P(x)$ or $g(y) \neq P(y)$ and that $x+y$ is uniformly distributed in G for x, y uniformly and independently chosen in G. To conclude, apply the union bound. □

There is nothing very special about the **Linearity Test** for the above argument to work. Indeed, suppose that $\Phi(f, \ldots)$ was a functional that gave rise to a K–local functional equation test. Then, the **Functional Equation Test** associated with Φ would be $(\eta, K\eta)$–continuous provided each evaluation of f was performed on an element chosen uniformly from a fixed subset of f's domain.

We now turn our attention to the harder task of proving robustness. In doing so we illustrate the most successful argument known for establishing this property — the so called majority function argument [BLR90, Cop89]. All proofs of robustness based on the majority function argument start by defining a function g whose value at x takes the most commonly occurring value among the members of a multiset S_x whose elements depend on x and P, i.e.,

$$g(x) = \operatorname*{Maj}_{s \in S_x}(s).$$

(Here, as well as throughout this paper, $\mathrm{Maj}_{s \in S}(s)$ denotes the most frequent among the elements of the multiset S, ties broken arbitrarily.) Moreover, there are three clearly identified stages in this type of proof argument. First, one shows that an overwhelming number of the elements of each S_x agree with the most commonly occurring value in the set, i.e., $g(x)$. Second, it is shown that g is close to P. Finally, it is shown that g has the property of interest (in the case of Theorem 3, that g is an homomorphism). The majority argument as well as its three main stages are illustrated by the following result taken from [BLR90].

Theorem 3. *Let G and G' be two finite abelian groups and let $P : G \to G'$ be an application such that for some constant $\eta < 1/6$,*

$$\operatorname*{\mathbf{Pr}}_{x,y \in G}[P(x+y) - P(x) - P(y) \neq 0] \leq \eta.$$

Then, there exists a homomorphism $g : G \to G'$ such that

$$\operatorname*{\mathbf{Pr}}_{x \in G}[g(x) \neq P(x)] \leq 2\eta.$$

Proof. We define the function $g(x) = \mathrm{Maj}_{y \in G}(P(x+y) - P(y))$. First, we show that with overwhelming probability $P(c+y) - P(y)$ agrees with $g(c)$, i.e.,

$$\operatorname*{\mathbf{Pr}}_{y \in G}[g(c) = P(c+y) - P(y)] \geq 1 - 2\eta. \tag{1}$$

By hypothesis, for randomly chosen x and y in G, we have that $P(c+x+y) - P(c+x) - P(y) \neq 0$ with probability at most η. Under the same conditions, the probability that $P(c+x+y) - P(c+y) - P(x) \neq 0$ is also upper bounded by η. Therefore,

$$\operatorname*{\mathbf{Pr}}_{x,y \in G}[P(c+x) - P(x) = P(c+y) - P(y)] \geq 1 - 2\eta.$$

Note that $\sum_{z \in G'}(\mathbf{Pr}_{y \in G}[P(c+y) - P(y) = z])^2$ equals the left hand side term of the previous inequality. By definition of $g(c)$ we know that for every $z \in G'$

$$\operatorname*{\mathbf{Pr}}_{y \in G}[P(c+y) - P(y) = z] \leq \operatorname*{\mathbf{Pr}}_{y \in G}[g(c) = P(c+y) - P(y)].$$

Since $\sum_{z \in G'} \mathbf{Pr}_{y \in G}[P(c+y) - P(y) = z] = 1$, we obtain (1).

Suppose now, for the sake of contradiction, that the distance between P and g was greater than 2η. By (1), for every x the probability that $g(x) = P(x+y) - P(y)$ is at least $1/2$ when y is randomly chosen in G. Thus,

$$\operatorname*{\mathbf{Pr}}_{x,y \in G}[P(x+y) - P(x) - P(y) \neq 0] > \eta,$$

which contradicts our hypothesis.

Finally, we prove that g is indeed a homomorphism. Fix $a, b \in G$. Applying (1) three times we get that, with probability at least $1 - 6\eta$ when y is randomly chosen in G, the following three events hold

$$\begin{aligned} g(a) &= P(a+y) - P(y), \\ g(b) &= P(a+b+y) - P(a+y), \\ g(a+b) &= P(a+b+y) - P(y). \end{aligned}$$

Therefore,

$$\mathop{\mathbf{Pr}}_{y \in G}\left[g(a+b) = g(a) + g(b)\right] > 1 - 6\eta > 0.$$

Since the event $g(a+b) = g(a) + g(b)$ is independent of y, we get that $g(a+b) = g(a) + g(b)$ must hold. □

Note that the proof of the previous result shows more than what its statement claims. In fact, the proof is constructive and it not only shows that an homomorphism g with the claimed properties exist, but that one such homomorphism is

$$g(x) = \mathop{\mathrm{Maj}}_{y \in G}\left(P(x+y) - P(y)\right).$$

Also, observe that a direct consequence of Theorem 3 is that the **Linearity Test** is $(2\eta, \eta)$–robust provided $\eta < 1/6$, or simply $(6\eta, \eta)$–robust (for every $\eta \geq 0$) if one is not so much concerned with the constants. We will use the latter statement since, rather than derive the best possible constants, in this work we strive to present ideas as clearly as possible. A similar convention is adopted throughout this survey for all the tests we discuss.

Corollary 1. *Let G and G' be two abelian groups, let $\mathcal{C}$ be the family of all functions from G to G', and let $\mathcal{L} \subseteq \mathcal{C}$ be the set of homomorphisms. Then, for every $\eta > 0$, there is an $(\eta, 19\eta)$–self–tester for $\mathcal{L}$ on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/\eta)$ calls to the oracle program, additions comparisons, counter increments, and binary shifts.*

Proof. The **Linearity Test** characterizes $\mathcal{L}$ since it is induced by the functional equation

$$\forall x, y \in G, \qquad P(x+y) - P(x) - P(y) = 0.$$

Realizing the **Linearity Test** just means randomly choosing x and y in G and verifying whether $P(x+y) - P(x) - P(y) = 0$. By Theorem 2, the test is $(\eta, 3\eta)$–continuous and by Theorem 3 it is also $(6\eta', \eta')$–robust. Letting $\eta' = (3 + 1/6)\eta = 19\eta/6$ and applying Theorem 1, the existence of the claimed self–tester is established. □

2.3 Self–correcting

We saw that for some function classes $\mathcal{F}$, a self–tester can be used to ascertain whether a program P correctly computes a function in $\mathcal{F}$. As we shall see later on, self–testing techniques can often be used to verify (probabilistically) whether a program P computes a specific function $g \in \mathcal{F}$ on some significant fraction of its domain. Sometimes in these cases, the program P itself can be used to compute g correctly with very large probability *everywhere* in its domain. This leads to the following:

Definition 5 (Self–corrector). *Let $\mathcal{F} \subseteq \mathcal{C}$ be a function family and let $\eta \geq 0$. An η–*self–corrector *for $\mathcal{F}$ on $\mathcal{C}$ is a probabilistic oracle Turing machine T such that for every $P \in \mathcal{C}$, if $\mathsf{Dist}(P, g) \leq \eta$ for some $g \in \mathcal{F}$, then for every $x \in D$ and for every confidence parameter $0 < \gamma < 1$, the output $T^P(x, \gamma)$ is $g(x)$ with probability at least $1 - \gamma$.*

Note that by definition, in order to possess an *η-self–corrector*, a family $\mathcal{F}$ has to satisfy that for each function $P \in \mathcal{C}$, there exists at most one function $g \in \mathcal{F}$ such that $\mathsf{Dist}(P, g) \leq \eta$.

Below we give an example of a self–corrector for the class of homomorphisms from one finite abelian group into another. In doing so we illustrate how the majority function argument discussed in the previous section naturally gives rise, when applicable, to self–correctors.

Theorem 4. *Let G and G' be two abelian groups, let $\mathcal{C}$ be the family of all functions from G to G', and let $\mathcal{L} \subseteq \mathcal{C}$ be the set of homomorphisms. Then, for every $0 \leq \eta < 1/4$ there is an η–self–corrector for $\mathcal{L}$ on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/(1-4\eta)^2)$ calls to the oracle program, additions, comparisons, counter increments, and binary shifts.*

Proof. For some fixed $P \in \mathcal{C}$, let g be the function defined in $x \in G$ by $g(x) = \mathrm{Maj}_{y \in G}\,(P(x+y) - P(y))$.

If $\mathsf{Dist}(P, \mathcal{L}) \leq \eta$ for some $0 \leq \eta < 1/4$, then there exists only one function $l \in \mathcal{L}$ such that $\mathsf{Dist}(P, l) \leq \eta$. This is a direct consequence of the fact that for all linear functions $l, l' \in \mathcal{L}$, since the elements $x \in G$ such that $l(x) = l'(x)$ form a subgroup H of G, either $H = G$ or $|H|/|G| \leq 1/2$. Therefore, either $l = l'$ or $\mathsf{Dist}(l, l') \geq 1/2$.

The closeness of P to l implies that $P(x+y) - P(y) = l(x)$ for at least half the elements y in G. Thus, by the definition of g, we get that $g = l$. Moreover, Chernoff bounds tell us that for every $x \in G$, the quantity $g(x)$ can be correctly determined with probability greater than $1-\gamma$ by choosing $N = O(\ln(1/\gamma)/(1-4\eta)^2)$ points $y_i \in G$ and then computing $\mathrm{Maj}_{i=1,\ldots,N}\,(P(x+y_i) - P(y_i))$. $\square$

Note that the function $g(x) = \mathrm{Maj}_{y \in G}\,(P(x+y) - P(y))$ played a key role both in the construction of a self–tester and of a self–corrector for the function class of homomorphisms. This is a distinctive feature of the majority function argument. Indeed, recall that this argument is constructive. Specifically, it proceeds by defining a function g whose value at x takes the most commonly occurring value among the members of a set S_x whose elements depend on x and P. When successful, this argument suggests that a self–corrector for the function class of interest is one that on input γ and x, for an appropriately chosen $N = N(\gamma)$, randomly chooses $g_1, \ldots, g_N$ in S_x and returns $\mathrm{Maj}_{i=1,\ldots,N}(g_i)$.

2.4 Generator test

An extreme case of self–testing is to ascertain whether a program P computes a fixed function f. This task is usually undertaken in two stages. First, self–testing techniques are used in order to determine whether P computes, in a large fraction of its inputs, a function f_P in some specific function class $\mathcal{F}$. For example, when f is a group homomorphism, one checks whether P is close to a group homomorphism. In the second stage, one ascertains whether f_P is indeed equal to f. To do so it suffices to check that f_P and f agree in a collection of inputs equality over which implies agreement everywhere. The process through which this goal is achieved is called *generator test.* This is somewhat complicated by the fact that one cannot evaluate f_P directly but only P which is merely close and not equal to f_P. Self–testing takes care of the first stage while self–correcting is useful in the second stage. We shall illustrate the technique with our benchmark linearity testing problem. First, we state a result implicit in the proof of Theorem 3.

Corollary 2. *Let G and G' be two finite abelian groups, and $(e_i)_{1\le i\le d}$ be a set of generators of G. Let $f : G \to G'$ be a homomorphism and $P : G \to G'$ be such that for some constant $\eta < 1/6$,*

$$\mathop{\mathbf{Pr}}_{x,y\in G}\left[P(x+y) - P(x) - P(y) \neq 0\right] \le \eta.$$

Suppose also that $g(e_i) = f(e_i)$ for $i = 1, \ldots, d$, where by definition $g(x) = \mathrm{Maj}_{y\in G}(P(x+y) - P(y))$. Then, $g = f$ and

$$\mathop{\mathbf{Pr}}_{x\in G}\left[f(x) \neq P(x)\right] \le 2\eta.$$

Proof. Implicit in the proof of Theorem 3 is the fact that g is an homomorphism and that $\mathbf{Pr}_{x\in G}\left[g(x) \neq P(x)\right] \le 2\eta$. Since two homomorphisms $f, g : G \to G'$ agree everywhere if and only if they agree on a set of generators of G, the desired conclusion follows. □

The preceding result tells us that the following procedure leads to a continuous and robust exact test for the linear function f.

Specific Linear Function Test(P, f)
- **Linearity Test**(P)
  1. Randomly choose $x, y \in G$.
  2. Reject if $P(x+y) - P(x) - P(y) \neq 0$.
- **Generator Test**(P)
  1. For $i = 1, \ldots, d$, reject if $P(x+e_i) - P(x) - f(e_i) \neq 0$.

The second part of this procedure consists of verifying that the closest linear function to P coincides with f on a set of generators for the group domain. Thus, it illustrates an instance of the generator test. The generator test is done on the corrected program which is computed by the self–correcting process. Indeed, instead of comparing $f(e_i)$ and $P(e_i)$ the comparison is done with respect to $P(x+e_i)-P(x)$ for a randomly chosen x. This is necessary since P although close to an homomorphism $f' \neq f$ might agree with f over all generators — but, in this case $P(x+e_i)-P(x)$ will most likely agree with $f'(e_i)$ for a randomly chosen x. Finally, observe how the **Linearity Test** simplifies the task of verifying whether P is close to the homomorphism f. Indeed, when P is known to compute on a large fraction of the inputs an homomorphism g, it is sufficient to check that g equals f on a set of generators whose size can be very small (constant) compared to the size of the whole domain.

Corollary 3. *Using the notation of Corollary 2, the* **Specific Linear Function Test** *characterizes f, is $(\eta,(3+d)\eta)$–continuous, and $(6\eta,\eta)$–robust.*

Nonetheless, when the number of generators d is large (for example grows with the size of the group), the number of calls to the program in the **Generator Test** will be large. This situation is called the *generator bottleneck*.

2.5 Generator bottleneck

In some cases, it is possible to get around the generator bottleneck using an *inductive test*. This is essentially another property test which eliminates the need of testing the self–corrected function on all the generators. We illustrate this point for the case of the Discrete Fourier Transform (DFT). The method and the results in this subsection are due to Ergün [Erg95]. For a more detailed discussion on the possibilities to circumvent the generator bottleneck by an inductive test see [ESK00].

Let p be a prime number and fix some $x=(x_0,\dots,x_n)\in\mathbb{Z}_p^{n+1}$, where $x_i\neq x_j$ for $i\neq j$. Then the linear function $\mathrm{DFT}_x:\mathbb{Z}_p^{n+1}\to\mathbb{Z}_p^{n+1}$ maps the coefficient representation $a=(a_0,\dots,a_n)\in\mathbb{Z}_p^{n+1}$ of a degree n polynomial $Q(X)=a_0+a_1X+\ldots+a_nX^n$ into its point-value representation $(Q(x_0),\dots,Q(x_n))\in\mathbb{Z}_p^{n+1}$. The group $\mathbb{Z}_p^{n+1}$ has $n+1$ generators which can be chosen for example as $e_0=(1,0,\dots0),\dots,e_n=(0,\dots,0,1)$. Applying the **Generator Test** in order to verify whether a linear function g is equal to DFT_x would require checking whether $g(e_i)=\mathrm{DFT}_x(e_i)$ for $i=0,\dots,n$, and therefore the number of calls to the program would grow linearly with the degree of the polynomial. The key observation that helps to overcome this problem is that the generators $e_0,\dots,e_n$ can be obtained from each other by a simple linear operation and that the same is true for the values of DFT_x on $e_0,\dots,e_n$. We now explain in detail how to take advantage of this fact. First, we need to introduce some notation. For $a=(a_0,\dots,a_n)\in\mathbb{Z}_p^{n+1}$ let the rotation to the right vector be $\mathrm{ROR}(a)=(a_n,a_0,\dots,a_{n-1})$ and let $x\cdot a=(x_0a_0,\dots,x_na_n)$. Note that $e_{i+1}=\mathrm{ROR}(e_i)$ for $i=0,\dots,n-1$, that DFT_x maps e_0 to $(1,1,\dots,1)$, and most importantly, that DFT_x sends $\mathrm{ROR}(a)$ to $x \cdot \mathrm{DFT}_x(a)$ for all $a = (a_0, \ldots, a_n) \in \mathbb{Z}_p^{n+1}$ with $a_n = 0$. Therefore, to verify whether a linear function g is equal to DFT_x it suffices to check whether g maps e_0 to $(1, 1, \ldots, 1)$ and that $g(\mathrm{ROR}(a)) = x \cdot g(a)$ for all a with $a_n = 0$. The robustness of this testing procedure is guaranteed by the following:

Theorem 5. *Let $x \in \mathbb{Z}_p^{n+1}$ and let $P : \mathbb{Z}_p^{n+1} \to \mathbb{Z}_p^{n+1}$ be an application such that for some constant $\eta < 1/6$,*

$$\mathop{\mathbf{Pr}}_{a,b \in \mathbb{Z}_p^{n+1}} [P(a+b) - P(a) - P(b) \neq 0] \leq \eta,$$

$$\mathop{\mathbf{Pr}}_{c \in \mathbb{Z}_p^{n+1} : c_n = 0} [g(\mathrm{ROR}(c)) \neq x \cdot g(c)] < 1/2,$$

where $g(a) = \mathrm{Maj}_{b \in \mathbb{Z}_p^{n+1}} (P(a+b) - P(b))$ for all $a \in \mathbb{Z}_p^{n+1}$ and $g(1, 0, \ldots, 0) = (1, \ldots, 1)$. Then, $g = \mathrm{DFT}_x$ and

$$\mathop{\mathbf{Pr}}_{a \in \mathbb{Z}_p^{n+1}} [\mathrm{DFT}_x(a) \neq P(a)] \leq 2\eta.$$

Proof. Theorem 3 and the comment following its proof guarantee that g is linear and that P is close to g. The linearity of g implies that for every $a, b \in \mathbb{Z}_p^{n+1}$, we have $g(\mathrm{ROR}(a+b)) = g(\mathrm{ROR}(a)) + g(\mathrm{ROR}(b))$. By linearity we also have $x \cdot (a+b) = x \cdot a + x \cdot b$ for every $a, b \in \mathbb{Z}_p^{n+1}$. Thus, the second probability bound in the hypotheses of the theorem implies that for all a with $a_n = 0$,

$$\mathop{\mathbf{Pr}}_{c \in \mathbb{Z}_p^{n+1}, c_n = 0} [g(\mathrm{ROR}(a)) = g(\mathrm{ROR}(c)) + g(\mathrm{ROR}(a-c)) = x \cdot g(a)] > 0.$$

Therefore, $g(\mathrm{ROR}(a)) = x \cdot g(a)$ always holds. To conclude the proof observe that the latter identity and the fact that $g(1, 0, \ldots, 0) = (1, \ldots, 1)$ imply that $g = \mathrm{DFT}_x$. □

The previous result suggests the following exact test in order to ascertain whether a program computes DFT_x.

> **$\mathbf{DFT}_x$ Test(P)**
> 1. Randomly choose $a, b \in \mathbb{Z}_p^{n+1}$.
> 2. Reject if $P(a+b) - P(a) - P(b) \neq 0$.
> 3. Reject if $P((1, 0, \ldots, 0) + a) - P(a) - (1, \ldots, 1) \neq 0$.
> 4. Randomly choose $c \in \mathbb{Z}_p^{n+1}$ such that $c_n = 0$.
> 5. Reject if $P(\mathrm{ROR}(c) + a) - P(a) - x \cdot (P(c+b) - P(b)) \neq 0$.

It follows that,

Corollary 4. *The* **$\mathbf{DFT}_x$ Test** *is such that it characterizes DFT_x, is $(\eta, 6\eta)$–continuous, and $(6\eta, \eta)$–robust.*

2.6 Beyond self–testing linearity

So far we have discussed the testing problem for collections of linear functions. This was done for ease of exposition. The arguments and concepts we have described are also useful in testing non–linear functions. We now illustrate this fact with two examples.

Multiplication over $\mathbb{Z}_n$: The **Linearity Test** and the **Generator Test** can be combined to yield various self–testers. One such example allows to ascertain whether a program computes the multiplication function mult over $\mathbb{Z}_n$, i.e., the function that associates to $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n$ the value xy (arithmetic over $\mathbb{Z}_n$). The exact test that achieves this goal is realized by the following procedure which is due to Blum, Luby, and Rubinfeld [BLR90]:

Multiplication Test(P)
1. Randomly choose $x, y, z \in \mathbb{Z}_n$.
2. Reject if $P(x, y + z) - P(x, y) - P(x, z) \neq 0$.
3. Reject if $P(x, y + 1) - P(x, y) - x \neq 0$.

Corollary 5. *The* **Multiplication Test** *is such that it characterizes* mult*, is* $(\eta, 4\eta)$*–continuous, and* $(6\eta, \eta)$*–robust.*

Proof. For a fixed $x \in \mathbb{Z}_n$, let $l_x : \mathbb{Z}_n \to \mathbb{Z}_n$ be the linear function defined by $l_x(y) = xy$. Let d_x be the distance $\mathsf{Dist}(P(x, \cdot), l_x)$ and let e_x be the rejection probability of the test for randomly chosen $y, z \in \mathbb{Z}_n$. By Corollary 3, we know that $e_x/4 \leq d_x \leq 6e_x$ for all x. Observe now that $\mathbf{E}_{x \in \mathbb{Z}_n}[d_x] = \mathsf{Dist}(P, \mathsf{mult})$ and that $\mathbf{E}_{x \in \mathbb{Z}_n}[e_x]$ is the probability that the test rejects P. The desired result follows. $\square$

Polynomials: Let $\mathbb{F}$ be a field and let $f : \mathbb{F} \to \mathbb{F}$ be a function. We adopt the standard convention of denoting the *forward difference operator* by ∇_t. Hence, by definition, $\nabla_t f(x) = f(x + t) - f(x)$ for $x, t \in \mathbb{F}$. If we let ∇_t^d denote the operator corresponding to d applications of ∇_t and for $\boldsymbol{t} \in \mathbb{F}^d$ denote by $\nabla_{\boldsymbol{t}}$ the operator corresponding to the applications of $\nabla_{t_1}, \ldots, \nabla_{t_d}$, then it is easy to check that:

1. ∇_t is linear,
2. ∇_{t_1} and ∇_{t_2} commute,
3. $\nabla_{t_1,t_2} = \nabla_{t_1+t_2} - \nabla_{t_1} - \nabla_{t_2}$, and
4. $\nabla_t^d = \sum_{k=0}^{d} (-1)^{d-k} \binom{d}{k} \nabla_{kt}$.

The usefulness of the difference operator in testing was recognized by Rubinfeld and Sudan [RS92b]. They used it to give a more efficient self–corrector for polynomials over finite fields than the one proposed by Lipton [Lip91]. Its utility is mostly based on two facts: $\nabla_t f(x)$ can be computed efficiently, and it gives rise to the following well known characterization of polynomials:

Theorem 6. *Let p be a prime number, let $f : \mathbb{Z}_p \to \mathbb{Z}_p$ be a function, and let $d < p-1$. Then f is a degree d polynomial over $\mathbb{Z}_p$ if and only if $\nabla_t^{d+1} f(x) = 0$ for all $x, t \in \mathbb{Z}_p$.*

The preceding theorem gives a functional equation characterization of degree d polynomials. Hence, it gives rise to the following functional equation test:

Degree d Polynomial Test(P)
1. Randomly pick $x, t \in \mathbb{Z}_p$.
2. Reject if $\nabla_t^{d+1} P(x) \neq 0$.

The above described exact test was proposed and analyzed in [RS92b]. Let us now discuss shortly its properties. For the sake of simplicity, consider the following particular case where $d = 1$:

Affine Test(P)
1. Randomly pick $x, t \in \mathbb{Z}_p$.
2. Reject if $P(x+2t) - 2P(x+t) + P(x) \neq 0$.

Instead of choosing t as above it is tempting to pick two values t_1 and t_2 also in $\mathbb{Z}_p$ and check whether $P(x+t_1+t_2) - P(x+t_1) - P(x+t_2) + P(x) \neq 0$. This is not an acceptable verification procedure in the self–testing context since it is essentially equivalent to affine interpolation (polynomial interpolation in the general case). Hence, it is not really computationally simpler than computing the functions of the class one wishes to test. On the contrary, the **Degree d Polynomial Test** is computationally more efficient than computing a degree d polynomial. Moreover, it requires less evaluations of the program P. This justifies the use of the ∇_t^{d+1} operator in testing degree d polynomials.

Since the **Degree d Polynomial Test** is $(d+2)$–local, the standard approach for proving continuity of such tests yield that it is $(\eta, (d+2)\eta)$–continuous. The robustness of the test is guaranteed by the following result of Rubinfeld and Sudan [RS92b]:

Theorem 7. *Let p be a prime number, let $P : \mathbb{Z}_p \to \mathbb{Z}_p$ be a function, and let $d < p-1$. If for some $\eta < 1/2(d+2)^2$,*

$$\mathop{\mathbf{Pr}}_{x,t \in \mathbb{Z}_p} \left[\nabla_t^{d+1} P(x) \neq 0 \right] \leq \eta,$$

then there exists a degree d polynomial $g : \mathbb{Z}_p \to \mathbb{Z}_p$ such that

$$\mathop{\mathbf{Pr}}_{x \in \mathbb{Z}_p} [g(x) \neq P(x)] \leq 2\eta.$$

Proof (Sketch). The proof is a standard application of the majority function argument albeit algebraically somewhat involved. We only describe the main proof steps. For $i = 0, \ldots, d+1$, let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$. Note that $\nabla_t^{d+1} P(x) = 0$ if and only if $P(x) = \sum_{i=1}^{d+1} \alpha_i P(x+it)$. This motivates the following definition:

$$g(x) = \mathop{\mathrm{Maj}}_{t \in \mathbb{Z}_p} \left(\sum_{i=1}^{d+1} \alpha_i P(x+it) \right).$$

The proof proceeds in the typical three stages that application of the majority function argument gives rise to. First, one shows that with overwhelming probability $\sum_{i=1}^{d+1} \alpha_i P(x+it)$ agrees with $g(x)$, in particular that

$$\mathop{\mathbf{Pr}}_{t\in\mathbb{Z}_p}\left[g(x)=\sum_{i=1}^{d+1}\alpha_i P(x+it)\right]\geq 1-2(d+1)\eta.$$

Second, one establishes that the distance between g and P is at most 2η. Finally, one proves that

$$\mathop{\mathbf{Pr}}_{t_1,t_2\in\mathbb{Z}_p}\left[\sum_{i=0}^{d+1}\alpha_i g(x+it)=0\right]\geq 1-2(d+2)^2\eta>0$$

Since the event $\sum_{i=0}^{d+1} \alpha_i g(x+it)=0$ is independent of t_1 and t_2, we get that $\sum_{i=0}^{d+1} \alpha_i g(x+it)=0$ must hold. The desired conclusion follows. □

3 Approximate self–testing

Initially it was assumed in the self–testing literature that programs performed exact computations and that the space of valid inputs was closed under the standard arithmetic operations, i.e., was an algebraically closed domain. However, early on it was recognized that these assumptions were too simplistic to capture the real nature of many computations, in particular the computation of real valued functions and of functions defined over finite rational domains (finite subsets of fixed point arithmetic of the form $\{i/s : |i|\leq n, i\in\mathbb{Z}\}$ for some $n,s>0$).

Self–testers/correctors for programs whose input values are from finite rational domains were first considered by Lipton [Lip91] and further developed by Rubinfeld and Sudan [RS92b]. In [Lip91] a self–corrector for multivariate polynomials over a finite rational domain is given. In the same scenario [RS92b] describes more efficient versions of this result as well as a self–tester for univariate polynomials. The study of self–testing in the context of inexact computations was started by Gemmell et al. [GLR+91] who provided approximate self–testers/correctors for linear functions, logarithmic functions, and floating point exponentiation. Nevertheless, their work was limited to the context of algebraically closed domains. Program checking in the approximate setting was first considered by Ar et al. [ABCG93] who provided, among others, approximate checkers for some trigonometric functions and matrix operations. Considering both aspects simultaneously led to the development of approximate self–testers over finite rational domains by Ergün, Kumar, and Rubinfeld [EKR96]. Among other things, they showed how to perform approximate self–testing with absolute error for linear functions, polynomials, and for functions satisfying addition theorems.

We now begin a formal discussion of the theory of approximate testing. Our exposition follows the presentation in [Mag00a] which has the advantage of encompassing all models of approximations studied in the self–testing literature.

Throughout this section, let D be a finite set and let R be a metric space. The distance in R will be denoted by $d(\cdot,\cdot)$, When R is also a normed space, its norm will be denoted by $\|\cdot\|$. As usual we denote by $\mathcal{C}$ the family of functions from D to R.

As in the case of the exact testing problem we are again interested in determining, maybe probabilistically, how "close" a program $P : D \to R$ is to an underlying family of functions of interest $\mathcal{F} \subseteq \mathcal{C}$. But now, the elements of R might be hard to represent (for example, when $\mathcal{F}$ is a family of trigonometric functions). Thus, any reasonable program P for computing $f \in \mathcal{F}$ will necessarily have to compute an approximation. In fact, P might never equal f over D but still be, for all practical purposes, a good computational realization of a program that computes f. Hence, the way in which we captured the notion of "closeness" in Section 2, that is, Definition 1, is now inadequate. Thus, to address the testing problem for R valued functions we need a different notion of incorrect computation. In fact, we need a definition of error. This leads to the following:

Definition 6 (Computational error term). *A* computational error term *for $\mathcal{C}$ is a function $\varepsilon : D \times R \to \mathbb{R}^+$. If $P, f : D \to R$ are two functions, then P* ε–computes f on $x \in D$ *if $d(P(x), f(x)) \leq \varepsilon(x, f(x))$.*

This definition encompasses several models of approximate computing that depend on the restriction placed on the computational error term ε. Indeed, it encompasses the

- exact computation case, where $\varepsilon(x, v) = 0$,
- approximate computation with absolute error, where $\varepsilon(x, v) = \varepsilon_0$ for some constant $\varepsilon_0 \in \mathbb{R}^+$,
- approximate computation with error relative to input size, where $\varepsilon(x, v) = \varepsilon_1(x)$ for some function $\varepsilon_1 : D \to \mathbb{R}^+$ depending only on x,
- approximate computation with relative error, where R is a normed space and $\varepsilon(x, v) = \theta\|v\|$ for some constant $\theta \in \mathbb{R}^+$.

Based on the definition of computational error term we can give a notion of distance, similar to that of Definition 1, which is more appropriate for the context of approximate computation.

Definition 7 (ε–Distance). *Let $P, f \in \mathcal{C}$, let $D' \subseteq D$, and let ε be a computational error term. The* ε–distance of P from f on D' *is*[2]

$$\mathsf{Dist}(P, f, D', \varepsilon) = \mathop{\mathbf{Pr}}_{x \in D'}\left[P \textit{ does not } \varepsilon\textit{–compute } f \textit{ on } x\right].$$

If $\mathcal{F} \subseteq \mathcal{C}$, then the ε–distance of P from $\mathcal{F}$ on D' *is*

$$\mathsf{Dist}(P, \mathcal{F}, D', \varepsilon) = \mathop{\mathrm{Inf}}_{f \in \mathcal{F}} \mathsf{Dist}(P, f, D', \varepsilon).$$

[2] The need for considering the values taken by P over a subset D' of f's domain is a technical one. We discuss this issue later on. In the meantime, the reader might prefer to simply assume $D' = D$.

The new notion of distance naturally gives rise to extensions of the notions introduced in Section 2. In what follows, we state these extensions.

Definition 8 (Approximate self–tester). *Let $\mathcal{F} \subseteq \mathcal{C}$, let $D' \subseteq D$, let ε and ε' be computational error terms and let $0 \leq \eta \leq \eta' < 1$ be constants. A $(D, \varepsilon, \eta; D', \varepsilon', \eta')$–*(approximate) self–tester* for $\mathcal{F}$ on $\mathcal{C}$ is a probabilistic oracle Turing machine T such that for every $P \in \mathcal{C}$ and for every confidence parameter $0 < \gamma < 1$:*

- *if $\mathsf{Dist}(P, \mathcal{F}, D, \varepsilon) \leq \eta$, then $\mathbf{Pr}\left[T^P(\gamma) = \texttt{GOOD}\right] \geq 1 - \gamma$;*
- *if $\mathsf{Dist}(P, \mathcal{F}, D', \varepsilon') > \eta'$, then $\mathbf{Pr}\left[T^P(\gamma) = \texttt{BAD}\right] \geq 1 - \gamma$,*

where the probabilities are taken over the coin tosses of T.

Definition 9 (Approximate test). *An* approximate test $(\mathcal{A}, \mathcal{C}, \mathcal{D}, \beta)$ *is a set of applications $\mathcal{A}$ from $\mathcal{C}$ to $\mathbb{R}^+$ with a distribution $\mathcal{D}$ over $\mathcal{A}$ and a* test error, *i.e., a function $\beta : \mathcal{A} \times \mathcal{C} \to \mathbb{R}^+$. The approximate test* characterizes *the family of functions*

$$\mathsf{char}(\mathcal{A}, \mathcal{C}, \mathcal{D}) = \{f \in \mathcal{C} \;:\; \mathop{\mathbf{Pr}}_{t \in_{\mathcal{D}} \mathcal{A}} [t(f) = 0] = 1\}.$$

The rejection probability *of a function $P \in \mathcal{C}$ by the approximate test is defined as*

$$\mathsf{Rej}(P, \mathcal{A}, \beta) = \mathop{\mathbf{Pr}}_{t \in_{\mathcal{D}} \mathcal{A}} [t(P) > \beta(t, P)].$$

A probabilistic oracle Turing machine M realizes *the approximate test if for all $P \in \mathcal{C}$,*

$$\mathbf{Pr}\left[M^P \text{ returns } \texttt{BAD}\right] = \mathsf{Rej}(P, \mathcal{A}, \beta),$$

where the probability on the left hand side is taken over the internal coin tosses of the machine M.

As in the case of exact testing, we specify approximate tests through the following high level description:

Approximate Test$(P \in \mathcal{C}, \mathcal{A}, \mathcal{D}, \beta)$
1. Choose an element $t \in \mathcal{A}$ according to $\mathcal{D}$.
2. Reject if $t(P) > \beta(t, P)$.

Note that one needs to compute the test error for realizing the approximate test. Also, exact tests are a particular case of approximate tests where the test error is 0 everywhere, `GOOD` is identified with 0 and `BAD` with 1.

In order not to unnecessarily clutter the notation, we again omit $\mathcal{C}$, $\mathcal{D}$, and β whenever clear from context and restrict our discussion to the case where $\mathcal{D}$ is the uniform distribution.

The robustness and continuity properties of exact tests are now generalized as follows:

Definition 10 (Continuity & robustness). *Let ε be a computational error term for $\mathcal{C}$, let $D' \subseteq D$, and let $(\mathcal{A}, \beta)$ be an approximate test on $\mathcal{C}$ which characterizes the family $\mathcal{F}$. Also, let $0 \leq \eta, \delta < 1$ be constants. Then, $(\mathcal{A}, \beta)$ is (η, δ)–*continuous *on D' with respect to ε if for all $P \in \mathcal{C}$,*

$$\mathsf{Dist}(P, \mathcal{F}, D', \varepsilon) \leq \eta \implies \mathsf{Rej}(P, \mathcal{A}, \beta) \leq \delta,$$

*and it is (η, δ)–*robust *on D' with respect to ε if for all $P \in \mathcal{C}$,*

$$\mathsf{Rej}(P, \mathcal{A}, \beta) \leq \delta \implies \mathsf{Dist}(P, \mathcal{F}, D', \varepsilon) \leq \eta.$$

The continuity and the robustness of an approximate test give rise to the construction of approximate self–testers through the following:

Theorem 8 (Approximate generic self–tester). *Let $0 \leq \delta < \delta' < 1$ and $0 \leq \eta \leq \eta' < 1$ be constants, $\mathcal{C}$ be a family of functions from a finite set D to a metric space R, ε and ε' be computational error terms for $\mathcal{C}$, and $D' \subseteq D$. Also, let $(\mathcal{A}, \beta)$ be an approximate test on $\mathcal{C}$ that is realized by the probabilistic Turing machine M. If $(\mathcal{A}, \beta)$ characterizes the family $\mathcal{F}$,*

- *is (η, δ)–continuous on D with respect to ε, and*
- *(η', δ')–robust on D' with respect to ε',*

then there exists a $(D, \varepsilon, \eta; D', \varepsilon', \eta')$–self–tester for $\mathcal{F}$ on $\mathcal{C}$ which performs for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)\frac{\delta+\delta'}{(\delta-\delta')^2})$ iterations of M, counter increments, comparisons, and binary shifts.

Proof. Similar to the proof of Theorem 1. □

As in the case of exact self–testing, realizable approximate tests are often constructed through functional equations. Specifically, for $D' \subseteq D$, let $\Phi : \mathcal{C} \times \mathcal{N} \to \mathbb{R}$ be a functional where $\mathcal{N} \subseteq \bigcup_{k=1}^{|D'|} (D')^k$ is a collection of neighborhoods. The functional Φ and a function $\beta' : \mathcal{N} \to \mathbb{R}^+$ induce an approximate test $(\mathcal{A}, \beta)$ by defining for all $(x_1, \ldots, x_k) \in \mathcal{N}$ the mapping $t_{x_1,\ldots,x_k}(f) : \mathcal{F} \to \mathbb{R}^+$ as $t_{x_1,\ldots,x_k}(f) = |\Phi(f, x_1, \ldots, x_k)|$, making $\beta(t_{x_1,\ldots,x_k}, f) = \beta'(x_1, \ldots x_k)$, and letting

$$\mathcal{A} = \{t_{x_1,\ldots,x_k} : (x_1, \ldots, x_k) \in \mathcal{N}\}.$$

By definition,

$$\mathsf{char}(\mathcal{A}) = \{f \in \mathcal{C} \ : \ \forall (x_1, \ldots, x_k) \in \mathcal{N}, \qquad \Phi(P, x_1, \ldots, x_k) = 0\}.$$

If Φ and β' are efficiently computable, then a Turing machine M realizes the induced approximate test by choosing $(x_1, \ldots x_k) \in \mathcal{N}$ and comparing the value $|\Phi(f, x_1, \ldots, x_k)|$ to $\beta'(x_1, \ldots x_k)$. When $(\mathcal{A}, \beta)$ is continuous and robust with respect to some computational error term, Theorem 8 can be applied to derive a corresponding approximate self–tester. The complexity of the self–tester will ultimately depend on the complexity of computing Φ and β'.

The approximate testing problem is technically more challenging and involved than the exact testing problem. We shall try to smoothly introduce the new aspects that one encounters in the approximate testing scenario. Thus, the discussion that follows is divided into three parts: the case of absolute error, the case of error relative to the input size, and finally the case of relative error. The discussion becomes progressively more involved. We shall try to stress the common arguments used in the different cases, but will discuss each one in a separate section.

Before proceeding, it is worth pointing out two common aspects of all known analyses of approximate tests. Specifically, in their proofs of robustness. First, they are considerably more involved than in the case of exact testing. Second, there are two clearly identifiable stages in such proofs. In each stage, it is shown that the approximate test exhibits one of the properties captured by the following two notions:

Definition 11 (Approximate robustness). *Let ε be a computational error term for $\mathcal{C}$ and $D' \subseteq D$. Let $(\mathcal{A}, \beta)$ and $(\mathcal{A}', \beta')$ be approximate tests on $\mathcal{C}$, both characterizing the family $\mathcal{F}$. Let $0 \leq \eta, \delta < 1$ be constants. Then, $(\mathcal{A}, \beta)$ is* (η, δ)–approximately robust *for $(\mathcal{A}', \beta')$ on D' with respect to ε if for all $P \in \mathcal{C}$,*

$$\mathsf{Rej}(P, \mathcal{A}, \beta) \leq \delta \implies \exists g \in \mathcal{C},\ \mathsf{Dist}(P, g, D', \varepsilon) \leq \eta,\ \mathsf{Rej}(g, \mathcal{A}', \beta') = 0.$$

Definition 12 (Stability). *Let ε be a computational error term for $\mathcal{C}$ and $D' \subseteq D$. Let $(\mathcal{A}, \beta)$ be an approximate test on $\mathcal{C}$ which characterizes the family $\mathcal{F}$. Then, $\mathcal{A}$ is* stable *on D' with respect to ε if for all $g \in \mathcal{C}$,*

$$\mathsf{Rej}(g, \mathcal{A}, \beta) = 0 \implies \mathsf{Dist}(g, \mathcal{F}, D', \varepsilon) = 0.$$

Note that stability is nothing else than $(0,0)$–robustness. A direct consequence of these definitions is that if $(\mathcal{A}, \beta)$ is approximately robust for $(\mathcal{A}', \beta')$ with respect to ε and $(\mathcal{A}', \beta')$ is stable with respect to ε', then $(\mathcal{A}, \beta)$ is also robust with respect to $\varepsilon + \varepsilon'$.

We henceforth restrict our discussion to real valued functions whose domain is $D_n = \{i \in \mathbb{Z} \ :\ |i| \leq n\}$ for some $n > 0$. Our results can be directly extended to finite rational domains. We conclude this section by stating some general facts that play a key role in the design and analysis of all approximate tests.

3.1 Basic tools

Here we state two simple lemmas which will be repeatedly applied in the forthcoming sections.

Definition 13 (Median). *For $f : X \to \mathbb{R}$ denote by $\mathrm{Med}_{x \in X}(f(x))$ the median of the values taken by f when x varies in X, i.e.,*

$$\operatorname*{Med}_{x \in X}(f(x)) = \mathrm{Inf}\{a \in \mathbb{R} : \operatorname*{\mathbf{Pr}}_{x \in X}[f(x) > a] \leq 1/2\}.$$

Lemma 1 (Median principle). *Let D, D' be two finite sets. Let $\varepsilon \geq 0$ and $F : D \times D' \to \mathbb{R}$. Then,*

$$\mathop{\mathbf{Pr}}_{x\in D}\left[\left|\mathop{\mathrm{Med}}_{y\in D'}(F(x,y))\right| > \varepsilon\right] \leq 2 \mathop{\mathbf{Pr}}_{(x,y)\in D\times D'}\left[|F(x,y)| > \varepsilon\right].$$

Proof. Observe that

$$\mathop{\mathbf{Pr}}_{x\in D}\left[\left|\mathop{\mathrm{Med}}_{y\in D'}(F(x,y))\right| > \varepsilon\right] \leq \mathop{\mathbf{Pr}}_{x\in D}\left[\mathop{\mathbf{Pr}}_{y\in D'}\left[|F(x,y)| > \varepsilon\right] > 1/2\right],$$

and apply Markov's inequality. □

Lemma 2 (Halving principle). *Let Ω and S denote finite sets such that $S \subseteq \Omega$, and let ψ be a boolean function defined over Ω. Then,*

$$\mathop{\mathbf{Pr}}_{x\in S}\left[\psi(x)\right] \leq \frac{|\Omega|}{|S|} \mathop{\mathbf{Pr}}_{x\in \Omega}\left[\psi(x)\right].$$

Proof.

$$\mathop{\mathbf{Pr}}_{x\in \Omega}\left[\psi(x)\right] \geq \mathop{\mathbf{Pr}}_{x\in \Omega}\left[\psi(x) | x \in S\right] \mathop{\mathbf{Pr}}_{x\in \Omega}\left[x \in S\right] = \frac{|S|}{|\Omega|} \mathop{\mathbf{Pr}}_{x\in \Omega}\left[\psi(x)\right].$$

□

If Ω is twice the size of S, then $\mathbf{Pr}_{x\in\Omega}\left[\psi(x)\right]$ is at least one half of $\mathbf{Pr}_{x\in S}\left[\psi(x)\right]$. This motivates the choice of name for Lemma 2.

We will soon see the importance that the median function has in the context of approximate self–testing. This was recognized by Ergün, Kumar, and Rubinfeld in [EKR96] where the median principle was also introduced. The fact that the Halving principle can substantially simplify the standard proof arguments one encounters in the approximate testing scenario was observed in [KMS99].

4 Testing with absolute error

Throughout this section we follow the notation introduced in the previous one. Moreover, we restrict our discussion to the case of absolute error, i.e., to the case where $\varepsilon(x, v)$ is some non–negative real constant ε. Again, for the purpose of illustration we consider the linearity testing problem over a rational domain D, say $D = D_{8n}$ for concreteness. Hence, taking $D' = D_{4n}$, the functional equation

$$\forall x, y \in D_{4n}, \qquad P(x+y) - P(x) - P(y) = 0,$$

gives rise to the following approximate absolute error test:

Absolute error Linearity Test(P, ε)
1. Randomly choose $x, y \in D_{4n}$.
2. Reject if $\lvert P(x+y) - P(x) - P(y)\rvert > \varepsilon$.

The preceding approximate test was proposed and analyzed by Ergün, Kumar, and Rubinfeld [EKR96]. We illustrate the crucial issues related to testing under absolute error by fully analyzing this approximate test. Our discussion is based on [EKR96] and simplifications proposed in [KMS99].

4.1 Continuity

As in the case of exact testing, continuity is a property which is usually much easier to establish than robustness. Although proofs of continuity in the approximate case follow the same argument than in the exact case, there is a subtlety involved. It concerns the use of the Halving principle as shown by the following result from which $(\eta, 6\eta)$–continuity of the **Absolute error Linearity Test** immediately follows.

Lemma 3. *Let $\varepsilon \geq 0$. Let P, l be real valued functions over D_{8n} such that l is linear. Then,*

$$\underset{x,y\in D_{4n}}{\mathbf{Pr}}\left[|P(x+y) - P(x) - P(y)| > 3\varepsilon\right] \leq 6 \underset{x\in D_{8n}}{\mathbf{Pr}}\left[|P(x) - l(x)| > \varepsilon\right].$$

Proof. Simply observe that $|P(x+y)-P(x)-P(y)| > 3\varepsilon$ implies $|P(x+y)-l(x+y)| > \varepsilon$ or $|P(x) - l(x)| > \varepsilon$ or $|P(y) - l(y)| > \varepsilon$. By the Halving principle, the probability that each of these three events occur when x and y are independently and uniformly chosen in D_{4n} is at most $2\,\mathbf{Pr}_{x\in D_{8n}}\left[|P(x) - l(x)| > \varepsilon\right]$. Thus, the union bound yields the desired conclusion. □

4.2 Approximate robustness

We now describe how robustness is typically established. Our discussion is based on [EKR96]. The majority argument will again be useful, but it needs to be modified. To see why, recall that the argument begins by defining a function g whose value at x takes the most commonly occurring value among the members of a multiset S_x whose elements depend on x and P, i.e.,

$$g(x) = \underset{s\in S_x}{\mathrm{Maj}}\,(s)\,.$$

Each value in S_x is seen as an estimation of the correct value of P on x. But now, P is not restricted to taking a finite number of values. There might not be any clear majority, or even worse, all but one pair of values in every set S_x might be distinct while very different from all other values in the set — the latter of these values might even be very similar among themselves. Thus, the $\mathrm{Maj}\,(\cdot)$ is not a good estimator in the context of testing programs that only approximately compute the desired value. A more robust estimator is needed. This explains why $\mathrm{Med}\,(\cdot)$ is used instead of $\mathrm{Maj}\,(\cdot)$. This gives rise to what we shall call the median function argument. The robustness proofs based on it will also exhibit three stages. The first two are similar to those encountered in the majority function argument. Indeed, first one shows that an overwhelming number of the elements of S_x are good approximations of $g(x) = \mathrm{Med}_{s\in S_x}(s)$, then one shows that g is close to P. The major difference is in the third stage — it falls short of establishing that g has the property one is interested in. For the sake of concreteness, we now illustrate what happens in the case of linearity testing.

Theorem 9. *Let $\varepsilon \geq 0$ and $0 \leq \eta < 1/96$ be constants and let $P : D_{8n} \to \mathbb{R}$ be an application such that*

$$\mathop{\mathbf{Pr}}_{x,y \in D_{4n}} \left[|P(x+y) - P(x) - P(y)| > \varepsilon \right] \leq \eta.$$

Then, there exists a function $g : D_{2n} \to \mathbb{R}$ such that

$$\mathop{\mathbf{Pr}}_{x \in D_n} \left[|g(x) - P(x)| > \varepsilon \right] \leq 16\eta,$$

and for all $a, b \in D_n$,

$$|g(a+b) - g(a) - g(b)| \leq 6\varepsilon.$$

Proof. Let $P_{x,y} = P(x+y) - P(x) - P(y)$. Define the function $g : D_{2n} \to \mathbb{R}$ by $g(x) = \mathrm{Med}_{y \in D_{2n}} \left(P(x+y) - P(y) \right)$. First, we show that with overwhelming probability $P(x+y) - P(y)$ is a good approximation to $g(x)$, specifically, that for all $c \in D_{2n}$ and $I \subseteq D_{2n}$ such that $|I| = |D_n|$,

$$\mathop{\mathbf{Pr}}_{y \in I} \left[|g(c) - (P(c+y) - P(y))| > 2\varepsilon \right] < 32\eta. \tag{2}$$

The Median principle implies that

$$\mathop{\mathbf{Pr}}_{y \in I} \left[|g(c) - (P(c+y) - P(y))| > 2\varepsilon \right] \leq 2 \mathop{\mathbf{Pr}}_{y \in I, z \in D_{2n}} \left[|P_{c+y,z} - P_{c+z,y}| > 2\varepsilon \right].$$

Observe that if y and z are randomly chosen in I and D_{2n} respectively, then the union bound yields

$$\mathop{\mathbf{Pr}}_{y,z} \left[|P_{c+y,z} - P_{c+z,y}| > 2\varepsilon \right] \leq \mathop{\mathbf{Pr}}_{y,z} \left[|P_{c+z,y}| > \varepsilon \right] + \mathop{\mathbf{Pr}}_{y,z} \left[|P_{c+y,z}| > \varepsilon \right].$$

To obtain (2), note that the Halving principle implies that the latter sum is at most

$$2 \frac{|D_{4n}|^2}{|D_n||D_{2n}|} \mathop{\mathbf{Pr}}_{x,y \in D_{4n}} \left[|P_{x,y}| > \varepsilon \right].$$

To see that g is close to P, observe that the Halving principle implies that

$$\mathop{\mathbf{Pr}}_{x \in D_n} \left[|g(x) - P(x)| > \varepsilon \right] \leq 4 \mathop{\mathbf{Pr}}_{x \in D_{4n}} \left[|g(x) - P(x)| > \varepsilon \right].$$

By definition of g we get that $g(x) - P(x) = \mathrm{Med}_{y \in D_{2n}} \left(P_{x,y} \right)$. Hence, the Median principle and the Halving principle yield

$$\begin{aligned} \mathop{\mathbf{Pr}}_{x \in D_n} \left[|g(x) - P(x)| > \varepsilon \right] &\leq 8 \mathop{\mathbf{Pr}}_{x \in D_{4n}, y \in D_{2n}} \left[|P_{x,y}| > \varepsilon \right] \\ &\leq 8 \frac{|D_{4n}|}{|D_{2n}|} \mathop{\mathbf{Pr}}_{x,y \in D_{4n}} \left[|P_{x,y}| > \varepsilon \right]. \end{aligned}$$

Elementary calculations and the hypothesis imply that the last expression is upper bounded by 16η.

Finally, let $a, b \in D_n$. Three applications of (2) imply that for some $y \in D_n$

$$|g(a) - (P(a+y) - P(y))| \leq 2\varepsilon,$$
$$|g(b) - (P(a+b+y) - P(a+y))| \leq 2\varepsilon,$$
$$|g(a+b) - (P(a+b+y) - P(y))| \leq 2\varepsilon.$$

It follows that $|g(a+b) - g(a) - g(b)| \leq 6\varepsilon$. □

The previous result falls short of what one desires. Indeed, it does not show that a low rejection probability for the **Absolute error Linearity Test** guarantees closeness to linearity. Instead, it establishes that if

$$|P(x+y) - P(x) - P(y)| > \varepsilon$$

holds for most x's and y's in a large domain, then P must be close to a function g which is approximately linear, i.e., for all a's and b's in a small domain,

$$|g(a+b) - g(a) - g(b)| \leq 6\varepsilon.$$

A conclusion stating that $g(a+b) = g(a)+g(b)$ would have been preferable. This will follow by showing that g is close to a linear function, thus implying the closeness of P to a linear function. By Definition 12, these results whereby it is shown that a function that approximately satisfies a functional equation everywhere must be close to a function that exactly satisfies the functional equation, are called stability proofs. Also, by Definition 11, results as those we have shown so far (i.e., whereby it is proved that a function that approximately satisfies a functional equation for most inputs must be close to a function that approximately satisfies the functional equation everywhere) are called approximate robustness proofs. As mentioned earlier, approximate robustness and stability imply robustness. In the following section we discuss a technique for proving stability results.

4.3 Stability

The main result of this section, i.e., the statement concerning stability of the **Absolute error Linearity Test** is from [EKR96]. However, the proof presented here is from [KMS99] and is based on an argument due to Skopf [Sko83]. The proof technique is also useful for obtaining stability results in the context of approximate testing over finite rational domains. It relies on two ideas developed in the context of stability theory. The first consists in associating to a function g approximately satisfying a functional equation a function h approximately satisfying the same functional equation but over an algebraically closed domain, e.g., a group. The function h is carefully chosen so that h agrees with g over a given subset of g's domain. In other words, h will be an *extension* of g. Thus, showing that h can be well approximated by a function with a given property is sufficient to establish that the function g can also be well approximated by a function with the same property. This task is easier to address due to the fact that h's domain has a richer algebraic structure. In fact, there is a whole community that for over half a century has been dedicated to the study of these type of problems. Indeed, in 1941, Hyers [Hye41] addressed one such problem for functions whose domain have a semi–group structure. The work of Hyers was motivated by a question posed by Ulam. Coincidentally, Ulam's question concerned linear functions. Specifically, Ulam asked whether a function f that satisfies the functional equation $f(x+y) = f(x) + f(y)$ only approximately could always be approximated by a linear function. Hyers showed that f could be approximated within a constant error term by a linear function when the equality was correct also within a constant term. To be precise, Hyers proved the following:

Theorem 10 (Hyers). *Let E_1 be a normed semi–group, let E_2 be a Banach space, and let $h : E_1 \to E_2$ be a mapping such that for all $x, y \in E_1$,*

$$\|h(x+y) - h(x) - h(y)\| \leq \varepsilon.$$

Then, the function $l : E_1 \to E_2$ defined by $l(x) = \lim_{m\to\infty} h(2^m x)/2^m$ is a well defined linear mapping such that for all $x \in E_1$,

$$\|h(x) - T(x)\| \leq 2\varepsilon.$$

Remark 1. We have stated Theorem 10 in its full generality in order to highlight the properties required of the domain and range of the functions we deal with. Also for this purpose, as long as we discuss stability issues, we keep the exposition at this level of generality. Nevertheless, we will apply Theorem 10 only in cases where $E_1 = \mathbb{Z}$ and $E_2 = \mathbb{R}$. □

Many other Ulam type questions have been posed and satisfactorily answered. For surveys of such results see [HR92, For95]. But, these results cannot directly be applied in the context of approximate testing. To explain this, recall that we are concerned with functions g such that $|g(x+y) - g(x) - g(y)| \leq \varepsilon$ only for x's and y's in D_n — which is not a semi–group. To address this issue and exploit results like those of Hyers one associates to g a function h that extends it over a larger domain which is typically a group. Moreover, the extension is done in such a way that one can apply a Hyers's type theorem.

Although the approach described in the previous paragraph is a rather natural one, it requires more work than necessary, at least for our purposes. Indeed, when deriving a stability type result for the approximate testing problem over D_n one considers the extension h of g given by $h(x) = g(r_x) + q_x g(n)$, where $q_x \in \mathbb{Z}$ and $r_x \in D_n$ are the unique numbers such that $x = q_x n + r_x$ and $|q_x n| < |x|$ if $x \in \mathbb{Z} \setminus \{0\}$, and $q_0 = r_0 = 0$. (See Fig. 2.) Thus, the limit of $h(2^m x)/2^m$ when m goes to ∞ is $xg(n)/n$. Hence, there is no need to prove that $l(x) = \lim_{m\to\infty} h(2^m x)/2^m$ is well defined and determines a linear mapping. Thus, when Hyers's theorem is applied to a function like h the only new thing we get is that l is close to h. As shown in the next lemma, to obtain this same conclusion a weaker hypothesis than that of Theorem 10 suffices. This fact significantly simplifies the proof of the stability results needed in the context of approximate testing.

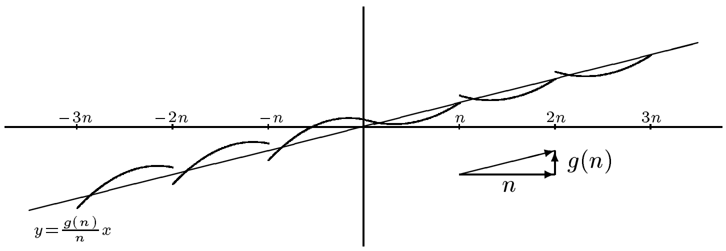

Fig. 2. Extension of g.

Lemma 4. *Let E_1 be a normed semi–group and E_2 be a Banach space. Let $\varepsilon \geq 0$ and let $h : E_1 \to E_2$ be such that for all $x \in E_1$,*

$$\|h(2x) - 2h(x)\| \leq \varepsilon.$$

Then, the function $T : E_1 \to E_2$ defined by $T(x) = \lim_{m\to\infty} h(2^m x)/2^m$ is a well defined mapping such that for all $x \in E_1$,

$$\|h(x) - T(x)\| \leq \varepsilon.$$

Proof. We follow the argument used by Hyers [Hye41] to prove Lemma 4. First, we show by induction on m, that

$$\left\| \frac{h(2^m x)}{2^m} - h(x) \right\| \leq \varepsilon \sum_{t=1}^{m} 2^{-t}. \tag{3}$$

The case $m = 1$ holds due to the hypothesis. Assume the claim is true for m. To prove the claim for $(m+1)$, note that

$$\begin{aligned} \left\| \frac{h(2^{m+1}x)}{2^{m+1}} - h(x) \right\| &\leq \left\| \frac{h(2x)}{2} - h(x) \right\| + \frac{1}{2} \left\| \frac{h(2^m \cdot 2x)}{2^m} - h(2x) \right\| \\ &\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \sum_{t=1}^{m} 2^{-t} \\ &= \varepsilon \sum_{t=1}^{m+1} 2^{-t}. \end{aligned}$$

Fix $x = 2^k y$ in (3). Then, the sequence $(h(2^k y)/2^k)_k$ satisfies a Cauchy criterion for every y. Therefore, T is well defined. Letting $m \to \infty$ in (3) one obtains the desired conclusion. $\square$

Thus, to establish the stability type result we are seeking for the linearity testing problem one needs to show that an appropriate extension $h : \mathbb{Z} \to \mathbb{R}$ of a function $g : D_{2n} \to \mathbb{R}$ such that $|g(x+y) - g(x) - g(y)| \leq \varepsilon$ for all $x, y \in D_n$ satisfies the hypothesis of Lemma 4. The following lemma achieves this goal.

Lemma 5. *Let $\varepsilon \geq 0$ and let $g : D_{2n} \to \mathbb{R}$ be such that for all $x, y \in D_n$,*

$$|g(x+y) - g(x) - g(y)| \leq \varepsilon.$$

Then, the function $h : \mathbb{Z} \to \mathbb{R}$ such that $h(x) = g(r_x) + q_x g(n)$ satisfies that for all $x \in \mathbb{Z}$,

$$|h(2x) - 2h(x)| \leq 2\varepsilon.$$

Proof. Let $x, y \in \mathbb{Z}$. By definition of h and since $r_{2x} = 2r_x - n(q_{2x} - 2q_x)$,

$$|h(2x) - 2h(x)| = |g(2r_x - n(q_{2x} - 2q_x)) - 2g(r_x) + (q_{2x} - 2q_x)g(n)|.$$

We will show that the right hand side of this equality is upper bounded by 2ε. Note that $q_{2x} - 2q_x \in \{-1, 0, 1\}$. We consider three cases depending on the value that this latter quantity takes.

Case 1: Assume $q_{2x} - 2q_x = 0$. Then, since $r_x \in D_n$, the hypothesis implies that $|h(2x) - 2h(x)| = |g(2r_x) - 2g(r_x)| \leq \varepsilon$.

Case 2: Assume now that $q_{2x} - 2q_x = 1$. Hence, $r_{2x} = 2r_x - n$ and

$$\begin{aligned}
|h(2x) - 2h(x)| &= |g(2r_x - n) - 2g(r_x) + g(n)| \\
&\leq |g(2r_x) - 2g(r_x)| + |g(2r_x - n) + g(n) - g(2r_x)| \\
&\leq 2\varepsilon,
\end{aligned}$$

where the first inequality is due to the triangle inequality and the second inequality follows from the hypothesis since $r_x, r_{2x} = 2r_x - n, n \in D_n$.

Case 3: Assume $q_{2x} - 2q_x = -1$. Hence, $r_{2x} = 2r_x + n$ which is at most n. Thus, r_x cannot be positive. This implies that $r_x + n \in D_n$ and

$$\begin{aligned}
|h(2x) - 2h(x)| &= |g(2r_x + n) - 2g(r_x) - g(n)| \\
&\leq |g(2r_x + n) - g(r_x + n) - g(r_x)| + |g(r_x + n) - g(r_x) - g(n)| \\
&\leq 2\varepsilon,
\end{aligned}$$

where the first inequality is due to the triangle inequality and the second one follows from the hypothesis since $r_x + n, r_x, n \in D_n$. □

An immediate consequence of the two previous results is the following:

Theorem 11. *Let $g : D_{2n} \to \mathbb{R}$ be a function such that for all $x, y \in D_n$,*

$$|g(x+y) - g(x) - g(y)| \leq \varepsilon.$$

Then, the linear function $l : D_{2n} \to \mathbb{R}$ defined by $l(n) = g(n)$ satisfies, for all $x \in D_n$,

$$|g(x) - l(x)| \leq 2\varepsilon.$$

4.4 Robustness

The results presented in the two previous sections yield the following:

Theorem 12. *Let $\varepsilon \geq 0$ and $0 \leq \eta < 1/96$ be constants, and let $P : D_{8n} \to \mathbb{R}$ be an application such that*

$$\mathbf{Pr}_{x,y \in D_{4n}} [|P(x+y) - P(x) - P(y)| > \varepsilon] \leq \eta.$$

Then, there exists a linear function $l : D_n \to \mathbb{R}$ such that

$$\mathbf{Pr}_{x \in D_n} [|l(x) - P(x)| > 13\varepsilon] \leq 16\eta.$$

Proof. Direct consequence of Theorem 9 and Theorem 11. □

This last result gives us the analog of Theorem 3 that we need in order to establish the robustness of the **Absolute error Linearity Test**.

4.5 Self–testing with absolute error

We now put together all the different pieces of the analyses of previous sections and establish the existence of an approximate self–tester for linearity.

Corollary 6. *Let $\mathcal{C}$ be the set of real valued functions over D_{8n} and let $\mathcal{L} \subseteq \mathcal{C}$ be the set of linear functions. Let $\eta > 0$ and $\varepsilon \geq 0$ be two constants. Then, there exists a $(D_{8n}, \varepsilon, \eta; D_n, 39\varepsilon, 577\eta)$–self–tester for $\mathcal{L}$ on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/\eta)$ calls to the oracle program, additions, comparisons, counter increments, and binary shifts.*

Proof. Consider the approximate test induced by the functional $\Phi(P, x, y) = P(x+y) - P(x) - P(y)$ where x and y are in D_{4n} and where the test error is 3ε. This approximate test clearly characterizes the family of linear functions, in fact, it gives rise to the **Absolute error Linearity Test**. Hence, by Lemma 3, it is $(\eta, 6\eta)$–continuous on D_{8n} with respect to the computational error term ε. Moreover, by Theorem 12, it is also $(96\eta', \eta')$–robust on D_n with respect to the computational error term $13(3\varepsilon)$. Therefore, Theorem 8 implies the desired result by fixing $\eta' = (6 + 1/96)\eta = 577\eta/96$. □

4.6 Self–correcting with absolute error

An obvious generalization of Definition 5 for any computational error term is:

Definition 14 (Approximate self–corrector). *Let $\mathcal{F} \subseteq \mathcal{C}$ be a function family from D to R and $D' \subseteq D$. Let $0 \leq \eta < 1$ and let ε, ε' be computational error terms for $\mathcal{C}$. An $(\eta, D, \varepsilon, D', \varepsilon')$–*(approximate) self–corrector *for $\mathcal{F}$ on $\mathcal{C}$ is a probabilistic oracle Turing machine T such that for every $P \in \mathcal{C}$, if $\mathsf{Dist}(P, f, D, \varepsilon) \leq \eta$ for some $f \in \mathcal{F}$, then for every $x \in D'$ and for every confidence parameter $0 < \gamma < 1$,*

$$\mathbf{Pr}\left[|T^P(x, \gamma) - f(x)| < \varepsilon'\right] > 1 - \gamma,$$

where the probability is taken over the internal coin tosses of T.

Of course, the above definition would be vacuous if we could not exhibit an example that satisfies it. We believe that in the same way that the majority argument gave rise to self–correctors, the median argument gives rise to approximate self–correctors. Below, we exhibit some supporting evidence for this claim by analyzing the problem of approximate self–correction of, the benchmark, class of linear functions.

Theorem 13. *Let $\mathcal{C}$ be the family of all real valued functions over D_{2n} and let $\mathcal{L} \subseteq \mathcal{C}$ be the set of linear functions. Then, for every $0 \leq \eta < 1/4$ and $\varepsilon \geq 0$, there is an $(\eta, D_{2n}, \varepsilon, D_n, 2\varepsilon)$–self–corrector for $\mathcal{L}$ on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/(1-4\eta)^2)$ calls to the oracle program, additions, comparisons, and counter increments.*

Proof. For some fixed $P \in \mathcal{C}$ assume $l : D_{2n} \to \mathbb{R}$ is a linear function such that $\mathsf{Dist}(P, l, D_{2n}, \varepsilon) \leq \eta$. Let N be a positive integer whose value will be determined later. Let T be the probabilistic Turing machine that on input $x \in D_n$, a constant $0 < \gamma < 1$, and oracle access to P, randomly chooses $y_1, \ldots, y_N \in D_n$ and then outputs $\mathrm{Med}_{i=1,\ldots,N}\,(P(x+y_i) - P(y_i))$. Note that,

$$
\begin{aligned}
&\mathbf{Pr}\left[|T^P(x,\gamma) - l(x)| > 2\varepsilon\right] \\
&= \mathbf{Pr}_{y_1,\ldots,y_N \in D_n}\left[|\operatorname*{Med}_{i=1,\ldots,N}(P(x+y_i) - P(y_i)) - l(x)| > 2\varepsilon\right] \\
&\leq \mathbf{Pr}_{y_1,\ldots,y_N \in D_n}\left[\operatorname*{Med}_{i=1,\ldots,N}(|P(y_i) - l(y_i)|) > \varepsilon\right] \\
&\quad + \mathbf{Pr}_{y_1,\ldots,y_N \in D_n}\left[\operatorname*{Med}_{i=1,\ldots,N}(|P(x+y_i) - l(x+y_i)|) > \varepsilon\right] \\
&\leq \mathbf{Pr}_{y_1,\ldots,y_N \in D_n}\left[|\{y_i : |P(y_i) - l(y_i)| > \varepsilon\}| \geq \frac{N}{2}\right] \\
&\quad + \mathbf{Pr}_{y_1,\ldots,y_N \in D_n}\left[|\{y_i : |P(x+y_i) - l(x+y_i)| > \varepsilon\}| \geq \frac{N}{2}\right].
\end{aligned}
$$

The Halving principle implies that both $\mathbf{Pr}_{y\in D_n}[|P(x+y) - l(x+y)| > \varepsilon]$ and $\mathbf{Pr}_{y\in D_n}[|P(y) - l(y)| > \varepsilon]$ are at most $2\,\mathsf{Dist}(P, l, D_{2n}, \varepsilon)$. Hence, when $N = \Omega(\ln(1/\gamma)(1-4\eta)^2)$, a standard Chernoff bound yields the desired result. □

4.7 Self–testing a specific function with absolute error

One might expect that self–testing whether a program computes a specific linear function over $D_n \subseteq \mathbb{Z}$ would be achieved by replacing in the **Specific Linear Function Test** the **Linearity Test** by the **Absolute error Linearity Test**, equalities by approximate equalities, and a generator of $\mathbb{Z}$ (i.e., -1 or 1) by any non–zero $x \in D_n$. We shall see that one has to be more careful in the choice of the latter element. In particular one has to perform the following:

Absolute error Specific Linear Function Test(P, ε)
- **Absolute error Linearity Test**(P, ε)
  1. Randomly choose $x, y \in D_{4n}$.
  2. Reject if $|P(x+y) - P(x) - P(y)| > \varepsilon$.
- **Generator Test**(P)
  1. Randomly choose $x \in D_n$.
  2. Reject if $|P(x+n) - P(x) - f(n)| > \varepsilon$.

We now explain why in the second stage of the previous test the comparison between f and the self–corrected value of P is performed at n. First, recall that when the probability η of rejection by the **Absolute error Linearity Test** is low we have a guarantee that P is close to a linear function. A careful inspection of the proof of Theorem 12 elicits that when l is the real valued linear function defined over D_n that at n takes the value $\mathrm{Med}_{y \in D_{2n}}(P(n+y) - P(y))$,

$$\mathop{\mathbf{Pr}}_{x \in D_n}[|l(x) - P(x)| > 13\varepsilon] \leq 16\eta.$$

This justifies the comparison that is performed, in the second part of the above described approximate test, between $f(n)$ and the estimation $P(x+n) - P(n)$ of $l(n)$'s value. Lemma 3 and the following result yield the $(\eta, 22\eta)$–continuity of the **Absolute error Specific Linear Function Test** (when the test error is 5ε) on D_{8n} with respect to the computational error term ε.

Lemma 6. *Let $\varepsilon \geq 0$. Let P, f be real valued functions over D_{8n} such that f is linear. Then,*

$$\mathop{\mathbf{Pr}}_{x \in D_n}[|P(x+n) - P(x) - f(n)| > 2\varepsilon] \leq 16 \mathop{\mathbf{Pr}}_{x \in D_{8n}}[|P(x) - f(x)| > \varepsilon].$$

Proof. The linearity of f, the union bound, and the Halving principle imply that

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{x \in D_n}[|P(x+n) - P(x) - f(n)| > 2\varepsilon] \\
&\quad \leq \mathop{\mathbf{Pr}}_{x \in D_n}[|P(x+n) - f(x+n)| > \varepsilon] + \mathop{\mathbf{Pr}}_{x \in D_n}[|P(x) - f(x)| > \varepsilon] \\
&\quad \leq 16 \mathop{\mathbf{Pr}}_{x \in D_{8n}}[|P(x) - f(x)| > \varepsilon].
\end{aligned}$$

□

The following result implies the $(49\eta, \eta)$–robustness of the **Absolute error Specific Linear Function Test** (when the test error is ε) on D_n with respect to the computational error term 16ε.

Lemma 7. *Let $\varepsilon \geq 0$ and $0 \leq \eta < 1/96$ be constants and let $P, f : D_{8n} \to \mathbb{R}$ be mappings such that f is linear and the probability that the* **Absolute error Specific Linear Function Test** *rejects is at most η. Then,*

$$\mathop{\mathbf{Pr}}_{x \in D_n}[|f(x) - P(x)| > 16\varepsilon] \leq 49\eta.$$

Proof. Let $l : D_n \to \mathbb{R}$ be such that $l(n) = \mathrm{Med}_{y \in D_{2n}}\left(P(n+y) - P(y)\right)$ and linear. Implicit in the proof of Theorem 12 is that $\mathbf{Pr}_{x \in D_n}\left[|l(x) - P(x)| > 13\varepsilon\right] \leq 16\eta$ and that (see (2))

$$\mathop{\mathbf{Pr}}_{y \in D_n}\left[|g(n) - (P(n+y) - P(y))| > 2\varepsilon\right] < 32\eta.$$

Thus, $\mathbf{Pr}_{x \in D_n}\left[|f(x) - P(x)| > 16\varepsilon\right]$ is at most

$$\mathop{\mathbf{Pr}}_{x \in D_n}\left[|l(x) - P(x)| > 13\varepsilon\right] + \mathop{\mathbf{Pr}}_{x \in D_n}\left[\frac{|x|}{n}|P(x+n) - P(n) - l(n)| > 2\varepsilon\right]$$
$$+ \mathop{\mathbf{Pr}}_{x \in D_n}\left[\frac{|x|}{n}|f(n) - P(x+n) + P(n)| > \varepsilon\right].$$

Since $|x|/n \leq 1$, the observation made at the beginning of this proof shows that the first and second term of the previous summation are bounded by 16η and 32η respectively. By the hypothesis and since $|x|/n \leq 1$, the last term of the summation is bounded by η. □

Corollary 7. *Let $\mathcal{C}$ be the collection of all real valued functions over D_{8n} and let $f \in \mathcal{C}$ be linear. Then, there exists a $(D_{8n}, \varepsilon, \eta; D_n, 80\varepsilon, 2113\eta)$–self–tester for f on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/\eta)$ calls to the oracle program, additions, comparisons, counter increments, and binary shifts.*

4.8 Beyond self–testing approximate linearity

So far we have discussed the approximate testing problem only for linear functions. The arguments we have used are also useful in testing non–linear functions. Nevertheless, a couple of new issues arise. To describe them we consider the problem of approximate testing whether a real valued function behaves like a degree d polynomial.

The characterization of degree d polynomials of Theorem 6, i.e., $\nabla_t^{d+1} f(x) = 0$ for all $x, t \in \mathbb{Z}_p$, still holds when $\mathbb{Z}_p$ is replaced by $\mathbb{Z}$. Hence, our discussion concerning approximate tests for class functions defined through functional equations suggest performing the following approximate test:

Absolute error Degree d Polynomial Test(P, ε)
1. Randomly choose $x \in D_m$ and $t \in D_n$.
2. Reject if $\lvert\nabla_t^{d+1} P(x)\rvert > \varepsilon$.

The above described approximate test was proposed and analyzed by Ergün, Kumar and Rubinfeld in [EKR96]. Since the approximate test with test error $2^{d+1}\varepsilon$ is $(d+2)$–local it is easily seen to be $(\eta, 2(d+2)\eta)$–continuous with respect to the computational error term ε, specifically:

Lemma 8. *Let $\varepsilon \geq 0$. Let P, Q be real valued functions over $D_{2(2d+3)n}$ such that Q is a polynomial of degree d. Then,*

$$\mathop{\mathbf{Pr}}_{x \in D_{(2d+3)n}, t \in D_n}\left[|\nabla_t^{d+1} P(x)| > 2^{d+1}\varepsilon\right] \leq 2(d+2) \mathop{\mathbf{Pr}}_{x \in D_{2(2d+3)n}}\left[|P(x) - Q(x)| > \varepsilon\right].$$

Proof. Simply observe that since Q is a degree d polynomial, $\nabla_t^{d+1} Q(x) = 0$ for all $x \in D_{(2d+3)n}$ and $t \in D_n$. Moreover, $\nabla_t^{d+1} = \sum_{k=0}^{d+1} (-1)^{d+1-k} \binom{d+1}{k} \nabla_{kt}$, $\sum_{k=0}^{d+1} \binom{d+1}{k} = 2^{d+1}$, and $|\nabla_t^{d+1} P(x)| > 2^{d+1}\varepsilon$ imply that $|P(x+it) - Q(x+it)| > \varepsilon$ for some $i \in \{0, \ldots, d+1\}$. By the Halving principle, the probability that any of the latter events occur when $x \in D_{(2d+3)n}$ and $t \in D_n$ are randomly chosen is at most $2\,\mathbf{Pr}_{x \in D_{2(2d+3)n}} [|P(x) - Q(x)| > \varepsilon]$. $\square$

The approximate robustness of the **Absolute error Degree d Polynomial Test** is a consequence of the following:

Lemma 9. *Let $\varepsilon \geq 0$ and $0 \leq \eta < 1/(16(d+1)(d+2)^2)$ be constants, and let P be a real valued function defined over $D_{2(2d+3)n}$ such that*

$$\mathop{\mathbf{Pr}}_{x \in D_{(2d+3)n}, t \in D_n} \left[|\nabla_t^{d+1} P(x)| > \varepsilon \right] \leq \eta.$$

Then, there exists a function $g : D'_n \subseteq D_{(d+2)n} \to \mathbb{R}$ such that

$$\mathop{\mathbf{Pr}}_{x \in D'_n} [|g(x) - P(x)| > \varepsilon] \leq 2 \frac{|D_{(2d+3)n}|}{|D'_n|} \eta,$$

and for all $x, t \in D_n$,

$$|\nabla_t^{d+1} g(x)| \leq 4(2^{d+1} - 1)^2 \varepsilon.$$

Proof (Sketch). The proof is based on the median argument and follows the proof idea of Theorem 9. The choice of g is the same as in Theorem 7 but now $\mathrm{Maj}\,(\cdot)$ is replaced by $\mathrm{Med}\,(\cdot)$ and $\mathbb{Z}_p$ by D_n, i.e.,

$$g(x) = \mathop{\mathrm{Med}}_{t \in D_n} \left(\sum_{i=1}^{d+1} (-1)^{i+1} \binom{d+1}{i} P(x + it) \right).$$

$\square$

As usual, approximate robustness leaves us with the need of a stability type result in order to establish robustness, in this case of the **Absolute error Degree d Polynomial Test**. We now undertake this endeavor.

For $\boldsymbol{t} \in \mathbb{Z}^d$ denote by $\nabla_{\boldsymbol{t}}$ the operator corresponding to the applications of $\nabla_{t_1}, \ldots, \nabla_{t_d}$. To avoid getting bogged down in technicalities and focus on the new issues that arise in approximate testing of non–linear function, we henceforth state the results for general d and restrict the proofs to the $d = 1$ case, i.e., the case of testing affine functions.

Lemma 10. *Let $\varepsilon \geq 0$. Let $f : D_{(d+1)n} \to \mathbb{R}$ be such for all $\boldsymbol{t} \in (D_n)^{d+1}$,*

$$|\nabla_{\boldsymbol{t}} f(0)| \leq \varepsilon,$$

Then, there exists a polynomial $h_d : D_n \to \mathbb{R}$ of degree at most d such that for all $x \in D_n$,

$$|f(x) - h_d(x)| \leq 2^d \left(\prod_{i=1}^{d} (2i - 1) \right) \varepsilon \leq 2^{2d} d! \varepsilon.$$

Proof (Sketch). We consider only the case of affine functions, i.e., $d = 1$. Let $G(t) = \nabla_t f(0) = f(t) - f(0)$. Then, for all $t_1, t_2 \in D_n$,

$$|G(t_1 + t_2) - G(t_1) - G(t_2)| = |\nabla_{t_1,t_2} f(0)| \leq \varepsilon.$$

Therefore, Theorem 11 implies that there exists a real valued linear function H over D_n such that $|G(t) - H(t)| \leq 2\varepsilon$ for all $t \in D_n$. Extending H linearly to all of $\mathbb{Z}$, defining f' over D_n by $f'(x) = f(x) - H(x)$, and observing that $H(0) = 0$ since H is linear, we get that for all $t \in D_n$,

$$|\nabla_t f'(0)| = |G(t) - H(t)| \leq 2\varepsilon.$$

To conclude, let $h(x) = f(0) + H(x)$ for all $x \in D_n$, and observe that h is an affine function such that $|f(x) - h(x)| = |\nabla_x f'(0)| \leq 2\varepsilon$. □

Remark 2. For the case of general d, the proof of Lemma 10 has to be modified. First, G is defined for every $t \in \mathbb{Z}^d$ where it makes sense as $G(\boldsymbol{t}) = \nabla_{\boldsymbol{t}} f(0)$. Instead of Theorem 11, one needs a stability type result asserting the existence of a multi–linear function H on d variables which is close to G. Instead of a linear extension of H one relies on a multi–linear extension of H to $\mathbb{Z}^d$. The rest of the proof follows the same argument and exploits the fact that if $H'(x) = H(x, \ldots, x)$, then $\nabla_{\boldsymbol{t}} H'(0) = d! H(\boldsymbol{t})$ for all $\boldsymbol{t} \in \mathbb{Z}^d$. □

We are not yet done proving the stability result we seek. Indeed, the conclusion of Lemma 9 is that $|\nabla_t^{d+1} g(x)|$ is bounded when $x, t \in D_n$. In contrast, the hypothesis of Lemma 10 requires a bound on $|\nabla_{\boldsymbol{t}} g(0)|$ when $\boldsymbol{t} \in (D_n)^{d+1}$. The following result links both bounds. But, the linkage is achieved at a cost. Indeed, although our assumption will be that $|\nabla_t^{d+1} g(x)|$ is bounded for a very large range of values of $x, t \in \mathbb{Z}$, our conclusion will be that $|\nabla_{\boldsymbol{t}} g(0)|$ is bounded for a coarse range of values of $\boldsymbol{t} \in \mathbb{Z}^{d+1}$.

Lemma 11. *Let $\varepsilon \geq 0$, $\mu_{d+1} = \mathrm{lcm}\{1, \ldots, d+1\}$, $m = \mu_{d+1}(d+1)n$, and g be a real valued function over $D_{(d+2)m}$. Let $f : D_{(d+1)n} \to \mathbb{R}$ be such that $f(x) = g(\mu_{d+1} \cdot x)$. If for all $x, t \in D_m$,*

$$|\nabla_t^{d+1} g(x)| \leq \frac{\varepsilon}{2^{d+1}},$$

then for all $\boldsymbol{t} \in (D_n)^{d+1}$,

$$|\nabla_{\boldsymbol{t}} f(0)| \leq \varepsilon.$$

Proof (Sketch). We consider only the case of affine functions, i.e., $d = 1$. Observe that

$$\begin{aligned}\nabla_{t_1,t_2} f(0) &= \nabla_0^2 f(0) - \nabla_{-t_1}^2 f(t_1) - \nabla_{-t_2/2}^2 f(t_2) + \nabla_{-t_1-t_2/2}^2 f(t_1 + t_2) \\ &= \nabla_0^2 g(0) - \nabla_{-2t_1}^2 g(t_1) - \nabla_{-t_2}^2 g(t_2) + \nabla_{-2t_1-t_2}^2 g(t_1 + t_2).\end{aligned}$$

By hypothesis, each of the four terms in the last summation is upper bounded (in absolute value) by $\varepsilon/4$. The desired conclusion follows by triangle inequality. □

Putting together Lemma 9, Lemma 10, and Lemma 11 one obtains the following result from which the $(2^{O(d \log d)}\eta, \eta)$–robustness with respect to the computational error term $2^{O(d \log d)}\varepsilon$ of the **Absolute error Degree d Polynomial Test** with test error ε immediately follows:

Theorem 14. *Let $\varepsilon \geq 0$, $\eta \geq 0$, $\mu_{d+1} = \mathrm{lcm}\{1, \ldots, d+1\}$, $m = \mu_{d+1}(d+1)n$, and let $kD_n = \{kx \in \mathbb{Z} : x \in D_n\}$ for any positive integer k. Let $P : D_{2(2d+3)m} \to \mathbb{R}$ be such that*

$$\mathop{\mathbf{Pr}}_{x \in D_{(2d+3)m}, t \in D_m} \left[|\nabla_t^{d+1} P(x)| > \varepsilon \right] \leq \eta.$$

Then, there exists a polynomial $h_d : \mu_{d+1} D_n \to \mathbb{R}$ of degree at most d such that

$$\mathop{\mathbf{Pr}}_{x \in \mu_{d+1} D_n} \left[|P(x) - h_d(x)| > 32^{d+1} d! \varepsilon \right] \leq 4(d+2)^2 \mu_{d+1} \eta.$$

Corollary 8. *Let $\varepsilon \geq 0$ and $\eta > 0$ be constants, $\mu_{d+1} = \mathrm{lcm}\{1, \ldots, d+1\}$, and $m = \mu_{d+1}(d+1)n$. Let $\mathcal{C}$ be the set of real valued functions over $D_{2(2d+3)m}$, and let $\mathcal{P}_d \subseteq \mathcal{C}$ be the set of degree d polynomials. Then, there exists a $(D_{2(2d+3)m}, \varepsilon, \eta; \mu_{d+1} D_n, 2^{O(d \log d)}\varepsilon, 2^{O(d \log d)}\eta)$–self–tester for $\mathcal{P}_d$ on $\mathcal{C}$ which uses for every confidence parameter $0 < \gamma < 1$, $O(\ln(1/\gamma)/\eta)$ calls to the oracle program, additions, comparisons, counter increments, and binary shifts.*

Proof (Sketch). Similar to the proof of Corollary 6 but now based on Lemma 8 and Theorem 14. □

Note how the probability bounds in the statement of Theorem 14 depend exponentially in d. It is not clear that there has to be a dependency in d at all. A similar result without any dependency on d would be interesting. Even a polynomial in d dependency would be progress.

5 Testing with error depending on input

In the preceding section we built self–testers for different function classes and domains for the case of absolute error. These self–testers exhibit the following characteristic: when the computational error term is a small constant they reject good programs, e.g, those in which the error in the computation of $P(x)$ grows with the size of x. If on the contrary, the computational error term is a large constant, they might pass programs that make incorrectly large errors in the computation of $P(x)$ for small values of x. In the next section we address the problem of self–testing when the computational error term can be proportional to the function value to be computed. In this section, we consider the intermediate case where the computational error terms are measured relative to some pre-specified function of the input x to the program P being tested. In particular, they do not depend on the function f purportedly being computed. The results presented here appeared in [KMS99].

In order to achieve the above stated goal, we generalize the arguments discussed in the preceding sections. We begin by pointing out that a careful inspection of the proofs of Theorem 9 and Theorem 11 yield that they still hold as long as the test error satisfies a collection of properties captured by the following:

Definition 15 (Valid error terms of degree $p \in \mathbb{R}$). *These are nonnegative functions $\beta : \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}^+$ which are, in each of their coordinates, even and nondecreasing for nonnegative integers, and such that $\beta(2s, 2t) \leq 2^p \beta(s,t)$ for all integers s, t.*

Examples of valid error terms of degree p are $\beta(s,t) = |s|^p + |t|^p$ and $\beta(s,t) = \mathrm{Max}\{c, |s|^p, |t|^p\}$ for some nonnegative real constant c. Whenever it is clear from context, we abuse notation and interpret a degree p error $\beta(\cdot,\cdot)$ as the function of one variable, denoted $\beta(z)$, that evaluates to $\beta(z,z)$ at z. Also, for $0 \leq p < 1$, we set $C_p = (1 + 2^p)/(2 - 2^p)$ and henceforth throughout this section use this notation.

When it is clear from the context, speaking about valid error terms β will both refer to test errors and computational error terms of the form $\varepsilon(x, v) = \beta(x)$.

5.1 Stability

By our choice of definition for test error depending on input size, with some effort but no new ideas, one can generalize the proof arguments of Lemma 4 and Lemma 5 and derive the following analog of Theorem 11:

Theorem 15. *Let $\beta(\cdot,\cdot)$ be a valid error term of degree p where $0 \leq p < 1$. Let $g : D_{2n} \to \mathbb{R}$ be such that for all $x, y \in D_n$,*

$$|g(x + y) - g(x) - g(y)| \leq \beta(x, y).$$

Then, the linear mapping $T : \mathbb{Z} \to \mathbb{R}$ defined by $T(n) = g(n)$ is such that for all $x \in D_n$,

$$|g(x) - T(x)| \leq C_p \beta(x).$$

This last theorem is the stability type result we need to establish robustness once we prove the approximate robustness of the analog of the **Absolute error Linearity Test** where instead of comparing $|P(x + y) - P(x) - P(y)|$ to a fixed constant ε the comparison is made against $\beta(x, y)$.

5.2 Approximate Robustness

We again rely on the median argument, but there is a crucial twist that needs to be introduced in order to address the cases of non–constant test errors we are concerned with. To explain the new twist, recall that in the median argument one begins by defining a function g whose value at x is the median of a multiset S_x whose elements depend on x and P, i.e.,

$$g(x) = \operatorname*{Med}_{s \in S_x}(s).$$

Each value s in S_x is seen as an estimation of the correct value that P takes on x. We would like $g(x)$ to be a very good estimation of the correct value taken by P on x. But now, how good an estimation is depends on the size of x. The smaller the size of x, the more accurate we want the estimation to be. This forces a new definition for $g(x)$, specially when x is small. The following result illustrates this point for the case of linearity testing with valid error terms.

Theorem 16. *For* $0 \leq \delta \leq 1$ *and a valid error term* $\beta(\cdot,\cdot)$ *of degree* $0 \leq p < 1$ *define* $\widetilde{\beta}(z) = \beta(\mathrm{Max}\{n\sqrt{\delta}, |z|\})$. *Let* $P : D_{8n} \to \mathbb{R}$ *be a mapping such that*

$$\mathop{\mathbf{Pr}}_{x,y \in D_{4n}} [|P(x+y) - P(x) - P(y)| > \beta(x,y)] \leq \delta/384.$$

Then, there exists a function $g : D_{2n} \to \mathbb{R}$ *such that*

$$\mathop{\mathbf{Pr}}_{x \in D_n} \left[|g(x) - P(x)| > \widetilde{\beta}(x)\right] \leq \delta/6,$$

and for all $a, b \in D_n$*,*

$$|g(a+b) - g(a) - g(b)| \leq 16\,\mathrm{Max}\{\widetilde{\beta}(a), \widetilde{\beta}(b)\}.$$

Proof (Sketch). The key point is the choice of g, i.e., for $x \in D_n$ define

$$g(x) = \begin{cases} \mathop{\mathrm{Med}}_{y \in D_{|x|}} (P(x+y) - P(y)), & \text{if } |x| \geq n\sqrt{\delta}, \\ \\ \mathop{\mathrm{Med}}_{y \in D_{n\sqrt{\delta}}} (P(x+y) - P(y)), & \text{otherwise.} \end{cases}$$

Then, following the proof argument of Theorem 9, one obtains the desired conclusion (although not without effort). □

Note how in the above definition of g, the median is taken over sets of different sizes. We henceforth refer to this variation of the median argument as the variable size median argument.

5.3 Robustness

The main goal of the two previous sections was to help establish the following:

Theorem 17. *Let* $0 \leq \delta \leq 1$ *and* $\beta(\cdot,\cdot)$ *be a valid error term of degree* $0 \leq p < 1$. *If* $P : D_{8n} \to \mathbb{R}$ *is such that*

$$\mathop{\mathbf{Pr}}_{x,y \in D_{4n}} [|P(x+y) - P(x) - P(y)| > \beta(x,y)] \leq \delta/384,$$

then there exists a linear function $T : \mathbb{Z} \to \mathbb{R}$ *such that*

$$\mathop{\mathbf{Pr}}_{x \in D_n} [|P(x) - T(x)| > 17C_p\beta(x)] \leq 7\sqrt{\delta}/6.$$

Proof. Let $\widetilde{\beta}(z) = \beta(\mathrm{Max}\{n\sqrt{\delta}, |z|\})$ and $\beta'(x,y) = 16\,\mathrm{Max}\{\widetilde{\beta}(x), \widetilde{\beta}(y)\}$. Since $\beta'(\cdot,\cdot)$ is a valid error term of degree p, Theorem 15 and Theorem 16 imply that,

– there is a function $g : D_{2n} \to \mathbb{R}$ such that when $x \in D_n$ is randomly chosen, $|g(x) - P(x)| > \widetilde{\beta}(x)$ with probability at most $\delta/6$, and
– there is a linear map $T : \mathbb{Z} \to \mathbb{R}$ such that $|g(x) - T(x)| \le 16C_p\widetilde{\beta}(x)$ for all $x \in D_n$.

Since $1 \le C_p$, if $|P(x) - T(x)| > 17C_p\widetilde{\beta}(x)$, then $|g(x) - P(x)| > \widetilde{\beta}(x)$ when $x \in D_n$. Hence, $\mathbf{Pr}_{x\in D_n}\left[|g(x) - P(x)| > 17C_p\widetilde{\beta}(x)\right]$ is at most $\delta/6 \le \sqrt{\delta}/6$. To conclude the proof observe that $\widetilde{\beta}(x) = \beta(x)$ with probability at least $1 - \sqrt{\delta}$ when x is randomly chosen in D_n. □

The previous result is the analog of Theorem 12 one needs to construct an approximate self–tester for linear functions provided the valid error term $\beta(\cdot,\cdot)$ is easily computable. Indeed, given oracle access to the program P and a valid error term $\beta(\cdot,\cdot)$, one can perform the following procedure:

1. Randomly choose $x, y \in D_{4n}$.
2. Reject if $|P(x{+}y) - P(x) - P(y)| > \beta(x,y)$.

We are now faced with a crucial difference between testing in the absolute error case and the case where the test errors depend on the size of the inputs. The point is that the above defined approximate test can be implemented provided one has a way of computing efficiently the valid error term, i.e., $\beta(\cdot,\cdot)$. Moreover, we would certainly like that computing the valid error term is simpler than computing whatever function P purportedly computes. In the case of linearity testing this is not always the case if the valid error term is a non–linear function, say $\beta(x,y) = \sqrt{|x|} + \sqrt{|y|}$. It is interesting to note that in most of the testing literature it is implicitly assumed that the test error is efficiently computable (always 0 in the case of exact testing and a fixed constant hardwired into the testing programs in the case of testing with absolute error). Fortunately, a good approximation of the test error suffices for self–testing. More precisely, provided the valid error term $\beta(\cdot,\cdot)$ is such that for some positive constants λ and λ' there is a function $\varphi(\cdot,\cdot)$ that is (λ,λ')–*equivalent* to $\beta(\cdot,\cdot)$, i.e., $\lambda\varphi(s,t) \ge \beta(s,t) \ge \lambda'\varphi(s,t)$ for all integers s, t. In addition, one desires that evaluating φ is asymptotically faster than executing the program being tested, say it only requires additions, comparisons, counter increments, and binary shifts. Surprisingly, this is feasible. For example, let k and k' be positive integers and let $\lg(n)$ denote the length of an integer n in binary. (Note that $\lg(n) = \lceil \log_2(|n|+1)\rceil$ or equivalently $\lg(0) = 0$ and $\lg(n) = \lfloor \log_2(|n|)\rfloor + 1$ if $n \ne 0$.) Then, $\beta(s,t) = 2^{k'}(|s|^{1/2^k} + |t|^{1/2^k})$ or $\beta(s,t) = 2^{k'}\,\mathrm{Max}\{|s|^{1/2^k}, |t|^{1/2^k}\}$ are valid error terms of degree $1/2^k$ which are $(1, 1/2)$–equivalent to $\varphi(s,t) = 2^{k'}(2^{\lceil \lg(s)/2^k\rceil} + 2^{\lceil \lg(t)/2^k\rceil})$ and $\varphi(s,t) = 2^{k' + \mathrm{Max}\{\lceil \lg(s)/2^k\rceil, \lceil \lg(t)/2^k\rceil\}}$ respectively. The computation of these latter functions requires only counter increments and shifting bits.

We have finally arrived at a point where we can propose an approximate test for linearity in the case of valid error terms $\beta(\cdot,\cdot)$ of degree $0 \leq p < 1$ for which there exists an equivalent function $\varphi(\cdot,\cdot)$, i.e.,

Input Size Relative error Linearity Test(P, φ)
1. Randomly choose $x, y \in D_{4n}$.
2. Reject if $\lvert P(x+y) - P(x) - P(y)\rvert > \varphi(x,y)$.

5.4 Continuity

As usual, establishing continuity, in this case of the **Input Size Relative error Linearity Test** is simple. We had not done so before simply because we had no candidate test to analyze. Below we establish the $(\eta, 6\eta)$–continuity with respect to the computational error term β of the mentioned approximate test with test error $\beta/4$, but to succeed we need and additional condition on the valid error term. We say that a valid error term $\beta(\cdot,\cdot)$ is *c–testable*, where c is a constant, if $\beta(s)+\beta(t)+\beta(s+t) \leq c\beta(s,t)$ for all s and t. For example, for k and k' integers, k positive, $\beta(s,t) = 2^{k'}(|s|^{1/2^k} + |t|^{1/2^k})$ and $\beta(s,t) = 2^{k'} \operatorname{Max}\{|s|^{1/2^k}, |t|^{1/2^k}\}$ are 4–testable valid error terms.

Lemma 12. *Let $\beta(\cdot,\cdot)$ be a 4–testable valid error term. Let P, l be real valued functions over D_{8n} such that l is linear. Then,*

$$\mathop{\mathbf{Pr}}_{x,y\in D_{4n}}\left[|P(x+y) - P(x) - P(y)| > \beta(x,y)\right] \leq 6 \mathop{\mathbf{Pr}}_{x\in D_{8n}}\left[|P(x) - l(x)| > \frac{\beta(x)}{4}\right].$$

Proof. Let $\beta' = \beta/4$. By the Halving principle,

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{x\in D_{4n}}\left[|P(x) - l(x)| > \beta'(x)\right] &\leq 2 \mathop{\mathbf{Pr}}_{z\in D_{8n}}\left[|P(z) - l(z)| > \beta'(z)\right], \\
\mathop{\mathbf{Pr}}_{y\in D_{4n}}\left[|P(y) - l(y)| > \beta'(y)\right] &\leq 2 \mathop{\mathbf{Pr}}_{z\in D_{8n}}\left[|P(z) - l(z)| > \beta'(z)\right], \\
\mathop{\mathbf{Pr}}_{x,y\in D_{4n}}\left[|P(x+y) - l(x+y)| > \beta'(x+y)\right] &\leq 2 \mathop{\mathbf{Pr}}_{z\in D_{8n}}\left[|P(z) - l(z)| > \beta'(z)\right].
\end{aligned}$$

Hence, since $\beta'(s) + \beta'(t) + \beta'(s+t) \leq \beta(s,t)$, the union bound implies the desired result. □

5.5 Self–testing with error relative to input size

We now piece together the results and concepts introduced in previous sections and establish the existence of realizable approximate self–testers for the case when the computational error term is allowed to depend on the size of the input. We stress that the existence of computationally efficient self–testers of this type is not a priori obvious since the test error might not be efficiently computable.

Theorem 18. *Let $0 < \eta \leq 1$ and $\beta(\cdot,\cdot)$ be a 4–testable valid error term of degree $0 < p < 1$ such that $\varphi(\cdot,\cdot)$ is (λ, λ')–equivalent to $\beta(\cdot,\cdot)$. Then, there* *is a* $(D_{8n}, \beta/(4\lambda), \eta/384; D_n, 17C_p\beta/\lambda', 7\sqrt{\eta}/6)$*–self–tester for the class of real valued linear functions over* D_{8n}*. Moreover, the self–tester uses for every confidence parameter* $0 < \gamma < 1$*,* $O(\ln(1/\gamma)/\eta)$ *calls to the oracle program, additions comparisons, counter increments, and binary shifts.*

Proof. First, assume that $\beta(\cdot,\cdot)$ is efficiently computable and consider the approximate test induced by the functional $\Phi(P,x,y) = P(x+y) - P(x) - P(y)$ where x and y are in D_{4n} and the test error is $\beta(\cdot,\cdot)$. This approximate test clearly characterizes the family of linear functions. In fact, it gives rise to the **Input Size Relative error Linearity Test**(P, β). Hence, by Lemma 12, it is $(\eta, 6\eta)$–continuous on D_{8n} with respect to the computational error term $\beta/4$. Moreover, by Theorem 17, it is also $(7\sqrt{\delta}/6, \delta/384)$–robust on D_n with respect to the computational error term $17C_p\beta$. Therefore, Theorem 8 implies the desired result by fixing $6\eta < \delta/384$.

To conclude the proof, we need to remove the assumption that the valid error term $\beta(\cdot,\cdot)$ is efficiently computable. To do so, consider the self–tester that performs sufficiently many independent rounds of the **Input Size Relative error Linearity Test**(P, φ). An analysis almost identical to the one described above applied to the new self–tester yields the desired result. □

Remark 3. The $\sqrt{\eta}$ dependency in the previous theorem, which is inherited from Theorem 17 is not the type of probability bound one usually sees in the context of exact and absolute error self–testing. Nevertheless, as the example below shows, this dependency seems to be unavoidable in the case of testing with errors that depend on the size of the input.

Let n be a positive integer, $0 < p < 1$, $0 < \delta < 1/4$, $\theta, c > 0$, $\beta(x,y) = \theta \operatorname{Max}\{|x|^p, |y|^p\}$, and consider the function $P : \mathbb{Z} \to \mathbb{R}$ such that (see Fig. 3)

$$P(x) = \begin{cases} -\theta(n\sqrt{\delta})^p, & \text{if } -n\sqrt{\delta} \le x < 0, \\ \theta(n\sqrt{\delta})^p, & \text{if } 0 < x \le n\sqrt{\delta}, \\ 0, & \text{otherwise.} \end{cases}$$

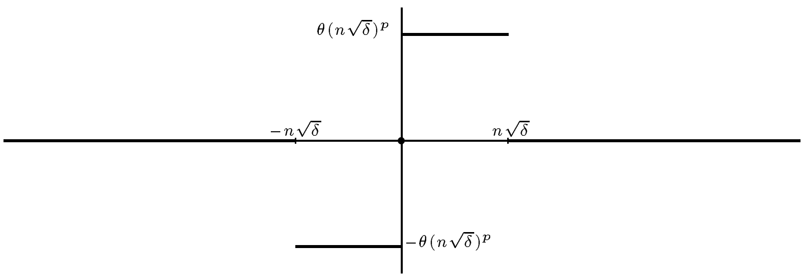

Fig. 3. The function P.

Observe that if $|x|$ or $|y|$ is greater than $n\sqrt{\delta}$ then $|P(x+y)-P(x)-P(y)| \leq 2\beta(x,y)$. Hence, if $n' \geq n$, with probability at most δ it holds that $|P(x+y) - P(x) - P(y)| > 2\beta(x,y)$ when x and y are randomly chosen in $D_{n'}$.

One can show that for every linear function T, when $x \in D_n$ is randomly chosen, $|P(x)-T(x)| > c\beta(x)$ with probability greater than $\sqrt{\delta}/(2(\mathrm{Max}\{1, 2c\})^{1/p})$. $\square$

Similar results as those stated above for linear functions hold for the class of polynomials. Their derivation is based on the arguments discussed in the previous as well as this section. Unfortunately, these arguments give rise to technically complicated proofs (for details see [KMS99]). Simplifications are certainly desirable.

6 Testing with relative error

In this section we consider the testing problem in the case where the allowed computational test error is proportional to the (absolute value of) the correct output one wishes to compute, i.e., the so called case of relative error. Again, we have oracle access to a program P purportedly computing a function belonging to a class of functions $\mathcal{F}$. The specific function f which P purportedly computes is unknown if there is more than one element in $\mathcal{F}$. The accuracy we wish P to compute f on x depends on the unknown value of $f(x)$. Thus, it is not a priori clear than one can self–test in the context of relative error. The discussion we now undertake will establish the plausibility of this task.

The forthcoming presentation is based on [Mag00a]. It shows how to build a self–tester for the class of linear functions in the case of relative error. The construction proceeds in two stages. First, one builds a linearity self–tester for the linear error case, i.e., the case where the test error is a linear function of the (absolute vale of) the input. This self–tester is then modified so as to successfully handle the case of relative error. The linear and relative error case, although related, exhibit a crucial difference. In the former case, since the error is just a known constant times the (absolute value of) the input, one knows (and thus can compute) the test error. In the latter case, one can not directly evaluate the test error since the function purportedly being computed by the oracle program is unknown and the test error depends on this value.

Note that in the context of linearity testing over rational domains, relative errors are functions that map x to $\theta|x|$ where θ is some unknown positive constant. Even if θ was known, Theorem 17 would not be applicable since it does not hold when $p = 1$. To see this, consider the real valued function over $\mathbb{Z}$ defined by $f(x) = \theta x \log_2(1 + |x|)$ for some $\theta > 0$. In [RS92a], it is shown that $|f(x+y) - f(x) - f(y)| \leq 2\theta\,\mathrm{Max}\{|x|, |y|\}$ for all $x, y \in \mathbb{Z}$. Clearly, no linear function is close to f. Hence, in the case of linear error, the **Input Size Relative error Linearity Test** is not a good self–tester for linearity. In order to overcome this situation it is natural to either use a different test error or modify the test. Based on the former option, in previous sections, approximate self–

testers were derived from exact self–testers. In contrast, to derive approximate self–testers in the case of linear error the latter path is taken.

When x is large, say $|x| \geq n/2$, a linear error term is essentially an absolute error term. When x is small, say $|x| < n/2$, we would like to efficiently amplify the linear error term to an absolute one. This can be done by multiplying x by the smallest power of 2 such that the absolute value of the result is at least $n/2$. This procedure can be efficiently implemented by means of binary shifts. Formally, each x is multiplied by 2^{k_x} where

$$k_x = \mathrm{Min}\{k \in \mathbb{N} : 2^k|x| \geq n/2\}.$$

(See Fig. 4 for an example where $n/8 < x < n/4$.)

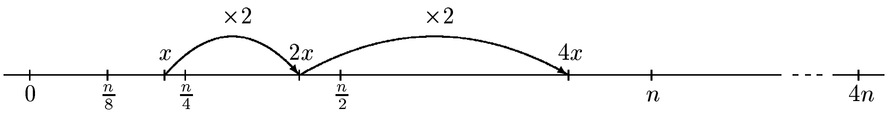

Fig. 4. Amplification procedure

The amplification procedures described above leads to the following new functional equation characterization of the class of linear functions (whose domain is D_{8n}):

$$\forall x, y \in D_{4n}, \qquad f(2^{k_x}x + y) - 2^{k_x}f(x) - f(y) = 0.$$

Note how this new characterization of linear functions relies not only on the additive properties of linear functions, but also on their homothetic properties.

6.1 Linear error

The previous section's functional equation characterization of linear functions leads in the standard way to a functional equation test. Specifically, for $\theta \geq 0$ it yields the following:

Linear error Linearity Test(P, θ)
1. Randomly choose $x, y \in D_{4n}$.
2. Reject if $\lvert P(2^{k_x}x + y) - 2^{k_x}P(x) - P(y)\rvert > \theta 2^{k_x}\lvert x\rvert$.

We henceforth denote by $\mathsf{Rej}(P, \theta)$ the rejection probability of P by the **Linear error Linearity Test**. The following claim establishes the continuity of this approximate test.

Lemma 13. *Let $\theta \geq 0$ and $\mathcal{L}$ be the set of linear functions over $\mathbb{Z}$. Then, for every $P : D_{8n} \to \mathbb{R}$,*

$$\mathsf{Rej}(P, \theta) \leq 6\,\mathsf{Dist}(P, \mathcal{L}, D_{8n}, \theta|x|/18).$$

The proof of robustness for the **Linear error Linearity Test** follows the usual two step approach where both the approximate robustness and the stability of the test are established. The first of these properties is guaranteed by the following:

Theorem 19. *Let $0 \leq \eta < 1/512$ and $\theta \geq 0$. Let $P : D_{8n} \to \mathbb{R}$ be such that*

$$\mathop{\mathbf{Pr}}_{x,y\in D_{4n}}\left[|P(2^{k_x}x+y) - 2^{k_x}P(x) - P(y)| > \theta 2^{k_x}|x|\right] \leq \eta.$$

Then, the function $g : D_{2n} \to \mathbb{R}$ defined by

$$g(x) = \frac{1}{2^{k_x}} \mathop{\mathrm{Med}}_{y\in D_{2n}:xy\geq 0} \left(P(2^{k_x}x+y) - P(y)\right),$$

is such that

$$\mathop{\mathbf{Pr}}_{x\in D_n}\left[|P(x) - g(x)| > \theta|x|\right] \leq 32\eta.$$

Moreover, $g(x) = g(2^{k_x}x)/2^{k_x}$ for all $x \in D_{2n}$, $|g(n) + g(-n)| \leq 16\theta n$, and for all x and y in $\{n/2, \ldots, n\}$ (respectively $\{-n/2, \ldots, -n\}$)

$$|g(x+y) - g(x) - g(y)| \leq 24\theta n.$$

Proof (Sketch). The proof follows the median function argument. The main difference is that we now have to cope with amplification terms. The closeness of g to P follows from the definition of g, the median principle, and the bound on rejection probability of the approximate test. The homethetic property of g under the amplification procedure follows directly from g's definition. It only remains to prove the approximate additivity of g in x and y (that is $g(x+y)$ is close to $g(x) + g(y)$) when the amplification terms of x, y and $x+y$ are all the same. More precisely, when both x and y belong to either $\{n/2, \ldots, n\}$ or $\{-n/2, \ldots, -n\}$ and when $\{x, y\} = \{-n, n\}$. This partly justifies the restriction on the set of elements of D_{2n} over which the median is taken (when $xy \geq 0$ one knows that the absolute values of $2^{k_x}x$, y, and $2^{k_x}x + y$ are all at least $n/2$). Therefore, they have no amplification factors associated to them. □

Note that the approximate additivity of g over $\{n/2, \ldots, n\}$ and $\{-n/2, \ldots, -n\}$ established by the previous result guarantees, due to g's homothetic property, its approximate additivity over small elements of g's domain.

The stability of the **Linear error Linearity Test** is established by the following:

Theorem 20. *Let $\theta_1, \theta_2 \geq 0$. Let $g : D_{2n} \to \mathbb{R}$ be such that $g(x) = g(2^{k_x}x)/2^{k_x}$ for all $x \in D_{2n}$, $|g(n) + g(-n)| \leq \theta_1 n$, and for all x and y in $\{n/2, \ldots, n\}$ (respectively $\{-n/2, \ldots, -n\}$)*

$$|g(x+y) - g(x) - g(y)| \leq \theta_2 n.$$

Then, the linear function $l : D_n \to \mathbb{R}$ defined by $l(n) = g(n)$ satisfies, for all $x \in D_n$,

$$|g(x) - l(x)| \leq (\theta_1 + 5\theta_2)|x|.$$

Proof (Sketch). The idea is to prove first that g is close to some linear function l (respectively l') on $\{n/2, \ldots, n\}$ (respectively $\{-n/2, \ldots, -n\}$), but in the absolute error sense. This can be achieved by an argument similar to the one used in the proof of Theorem 15. It follows that l and l' are necessarily close since $g(n)$ and $g(-n)$ are close to each other. Then, the homothetic property of g is used to transform absolute error bounds on the distance between g and l over $\{n/2, \ldots, n\}$ and $\{-n/2, \ldots, -n\}$, into linear error bounds over all of D_n. □

Theorem 19 and Theorem 20 immediately yield:

Theorem 21. *Let $\theta \geq 0$, $0 \leq \eta \leq 1/512$, $P : D_{8n} \to \mathbb{R}$, and $l : D_n \to \mathbb{R}$ be the linear function such that*

$$l(n) = \operatorname*{Med}_{y \in D_{2n} : y \geq 0} \left(P(n+y) - P(y) \right).$$

Then,

$$\mathsf{Rej}(P, \theta) \leq \eta \implies \mathsf{Dist}(P, l, D_n, 137\theta|x|) \leq 32\eta.$$

6.2 From linear error to relative error

We now undertake the second stage of the construction of the self–tester for the class of linear functions in the case of relative error. Specifically, we modify the **Linear error Linearity Test** so it can handle relative test errors. In order to explain this modification, consider a program P that approximately computes (with respect to relative errors) a linear function l. Then, one could allow a test error proportional to $l(n)$ in the **Linear error Linearity Test**. Since l is unknown, we need to estimate its value at n. Although P is close to l the value $P(n)$ might be very far from $l(n)$. Thus, $P(n)$ is not necessarily a good estimation of $l(n)$. We encountered a similar situation when self–testing a specific function. We addressed it via self–correction. The same approach succeeds here. This leads to the **Relative error Linearity Test** described below. To state it we first need to define (over $\mathbb{Z}$) the real valued function $ext(P, G)$ by:

$$ext(P,G)(x) = \begin{cases} P(x), & \text{if } x \in D_n, \\ ext(P,G)(x-n) + G, & \text{if } x > n, \\ ext(P,G)(x+n) - G, & \text{if } x < -n. \end{cases}$$

Then, the modified **Linear error Linearity Test** becomes the

> **Relative error Linearity Test**(P, θ)
> 1. Randomly choose $y \in \{0, \ldots, n\}$.
> 2. Compute $G_y = P(n-y) + P(y)$.
> 3. Compute $\tilde{\theta} = \theta |G_y| / n$.
> 4. Call **Linear error Linearity Test**$(ext(P, G_y), \tilde{\theta})$.

We henceforth denote by $\mathsf{Rej}^r(P, \theta)$ the rejection probability of P by the **Relative error Linearity Test** and let $\mathsf{Dist}^r(\cdot, \cdot, \cdot, \theta)$ denote $\mathsf{Dist}(\cdot, \cdot, \cdot, \varepsilon)$ when the computational error term ε is $\varepsilon(x, v) = \theta|v|$. The following results establish both the continuity and the robustness of the **Relative error Linearity Test**.

Lemma 14. *Let $0 \leq \theta \leq 18$, $\mathcal{L}$ be the set of linear functions over $\mathbb{Z}$, and $P : D_n \to \mathbb{R}$. Then,*

$$\mathsf{Rej}^r(P, \theta) \leq 10\,\mathsf{Dist}^r(P, \mathcal{L}, D_n, \theta/72).$$

Proof. Let $l : D_n \to \mathbb{R}$ be a linear function such that $\mathsf{Dist}^r(P, l, D_n, \theta/72) = \mathsf{Dist}^r(P, \mathcal{L}, D_n, \theta/72) = \eta$. For $y \in \{0, \ldots, n\}$ let $G_y = P(n-y) + P(y)$, $\tilde{\theta} = \theta|G_y|/n$, and $\tilde{P}_y = ext(P, G_y)$. By Lemma 2, $|G_y - l(n)| \leq \theta|l(n)|/36$ with probability greater than $1 - 4\eta$ when y is randomly chosen in $\{0, \ldots, n\}$. If this latter inequality is satisfied, then $\mathsf{Dist}^r(\tilde{P}_y, l, D_{8n}, \theta/36) \leq \eta$. Since $\theta/36 \leq 1/2$, the assumed inequality also implies that $|l(n)| \leq 2|G_y|$. Therefore, it follows that $\mathsf{Dist}(\tilde{P}_y, l, D_{8n}, \theta|x|/18) \leq \eta$. Lemma 13 implies that the rejection probability of the **Linear error Linearity Test**$(ext(P, G_y), \tilde{\theta})$ is at most 6η. It immediately follows that $\mathsf{Rej}^r(P, \theta) \leq (6+4)\eta$. ☐

Theorem 22. *Let $\theta \geq 0$, $0 \leq \eta \leq 1/512$, $\mathcal{L}$ be the set of linear functions over $\mathbb{Z}$, and $P : D_n \to \mathbb{R}$. Then,*

$$\mathsf{Rej}^r(P, \theta) \leq \eta \implies \mathsf{Dist}^r(P, \mathcal{L}, D_n, 137\theta) \leq 32\eta.$$

Proof. Assume $\mathsf{Rej}^r(P, \theta) \leq \eta$. Then, there exists a $y \in D_n$ such that for $G_y = P(n-y) + P(y)$ and $\tilde{\theta} = \theta|G_y|/n$, the rejection probability of **Linear error Linearity Test**$(ext(P, G_y), \tilde{\theta})$ is at most η. Thus, by Theorem 21, the linear function $l : D_n \to \mathbb{R}$ defined by

$$l(n) = \operatorname*{Med}_{y \in D_{2n} : y \geq 0} \left(P(n+y) - P(y)\right),$$

is such that $\mathsf{Dist}(P, l, D_n, 137\tilde{\theta}|x|) \leq 32\eta$. Then, it must be that $l(n) = G_y$ and therefore

$$\mathsf{Dist}^r(P, l, D_n, 137\theta) = \mathsf{Dist}(P, l, D_n, 137\tilde{\theta}|x|) \leq 32\eta.$$

☐

Similar results also hold for the class of multi–linear functions (see [Mag00b] for details).

7 Beyond testing algebraic functions

Since the pioneering work of Blum et al. [BLR90] appeared the concepts and paradigms discussed so far in this survey have been extended and studied under different contexts. This has been done in order to widen the scope of applicability of the concepts and results obtained in the self–testing literature. Below we discuss some of these extensions and new scenarios.

7.1 Testing and probabilistically checkable proofs

The results of [BFLS91, AS92b, ALM+92] concerning probabilistically checkable proofs (PCPs) enable the encoding of mathematical proofs so as to allow very efficient probabilistic verification. The latter consists of a simple randomized test that looks at a few bits of the proof and decides to accept or reject the proof's validity by performing a simple computation on those bits. Valid proofs are always accepted. Incorrect proofs are rejected with a non–negligible probability.

PCPs are built by recursion [AS92b]. Each level of the recursion uses a distinct form of error-correcting code. Correct encodings are viewed as representations of functions that satisfy a pre-specified property. Thus, a central problem in the construction of PCPs is to probabilistically check (test) whether a function satisfies a given property with as few queries as possible. Among the typical properties that come up in the PCP context are linearity [BGLR93, BS94, BCH+95, Tre98], multi–linearity [BFL90, FGL+91], low–individual degree [BFLS91, AS92b, PS94], low–total degree [ALM+92, AS97], and membership in the so called "long code" [Hås96, Hås97, Tre98]. Testing that a given function satisfies one of these properties is referred to as *low–degree testing*.

In the context of PCPs, thus in low–degree testing, the main concern is to minimize the number of oracle queries and the randomness used. It is not essential that the verification procedure be computationally more efficient than computing the functions of the class one wants to test. Also, continuity of the test is not so much of an issue, typically it only matters that the test never rejects an oracle that represents a function with the desired property. Robustness is the real issue.

Low–degree testing takes place in an adversarial scenario where the verification procedure is thought of as a verifier that has access to an oracle written down by a prover. The prover wishes to foul the verifier into believing the oracle is the table of a function satisfying a specific property. The verifier wants to determine whether this is true using as few probes and random bits as possible. To simplify his task the verifier may force the prover to add structure to the oracle. Moreover, he may choose a scenario where performing the verification is easier. For the sake of illustration and concreteness we shall consider below our benchmark linearity testing problem but in the PCP context.

The discussion that follows is taken from [KR97]. Assume G and H are finite abelian groups and $P : G \to H$. We want to verify whether P is linear, i.e., $P(x+y) = P(x) + P(y)$ for all $x, y \in G$. To simplify the verification procedure we choose G and H so they have a rich structure. Specifically, for a prime field $\mathbb{Z}_p$, we fix $G = \mathbb{Z}_p^n$ and $H = \mathbb{Z}_p$. Since $\mathbb{Z}_p$ is a prime field, P is linear if and only if $P(x) = \sum_{i=1}^{n} \alpha_i x_i$ for some $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p$. For $x \in \mathbb{Z}_p^n \setminus \{0\}$, denote by L_x the line in $\mathbb{Z}_p^n$ passing through x and 0, i.e., $L_x = \{tx : t \in \mathbb{Z}_p\}$. Observe that if $L_x \neq L_y$, then $L_x \cap L_y = \{0\}$. Note also that every linear function l over $\mathbb{Z}_p^n$ is such that $l(tx) = tl(x)$ for all $t \in \mathbb{Z}_p$ and $x \in \mathbb{Z}_p^n$. Hence, knowing the value of l at any non–zero element of a line completely determines the value of l over that line. We take advantage of this fact to facilitate the verification task. Indeed, we ask the prover to write down, for each line $L \subseteq \mathbb{Z}_p^n$, the value of P at one representative element of L (say the first $x \in L \setminus \{0\}$ according to the coordinate wise order induced by an identification of $\mathbb{Z}_p$ with the set $\{0, \ldots, p-1\}$). If we ever need to query the value of P at $x \neq 0$ we can determine it by querying the value of P at the representative of L_x and from this value compute $P(x)$ as if P was linear over L. This way, we are certain that $P(tx) = tP(x)$ for all $x \in \mathbb{Z}_p^n$ and $t \in \mathbb{Z}_p$. Equivalently, we can assume that the oracle function P has this property. We henceforth adopt this convention. Note in particular that this implies that $P(0) = 0$. Taking all the previously introduced conventions into account we perform the following:

Prime Field Linearity Test(P)
1. Randomly choose $x, y, z \in \mathbb{Z}_p^n$ such that $x+y+z = 0$.
2. Reject if $P(x) + P(y) + P(z) \neq 0$.

Henceforth, let $\mathcal{T}$ denote the previous test and $\mathbb{Z}_p^*$ the set $\mathbb{Z}_p \setminus \{0\}$. Also, let ω denote a p–th root of unity. Observe that for $\phi \in \mathbb{Z}_p$, $\phi = 0$ if and only if $\left(\sum_{t\in\mathbb{Z}_p} \omega^{t\phi}\right)/|\mathbb{Z}_p| = 1$. Moreover, $\phi \neq 0$ if and only if $\left(\sum_{t\in\mathbb{Z}_p} \omega^{t\phi}\right)/|\mathbb{Z}_p| = 0$. Hence, for $l \in \mathcal{L}$,

$$\mathsf{Rej}(P, \mathcal{T}) = 1 - \frac{1}{|\mathbb{Z}_p|^{2n}} \sum_{\substack{x,y,z\in\mathbb{Z}_p^n,\\ x+y+z=0}} \left(\frac{1}{|\mathbb{Z}_p|} \sum_{t\in\mathbb{Z}_p} \omega^{t(P(x)+P(y)+P(z))} \right). \tag{4}$$

Now, for two $\mathbb{Z}_p$ valued functions f and g over $\mathbb{Z}_p^n$ denote by ω^f the function that evaluates to $\omega^{f(x)}$ at x and define

$$\chi_g(f) = \frac{1}{|\mathbb{Z}_p|^n} \sum_{x\in\mathbb{Z}_p^n} \omega^{f(x)-g(x)}.$$

Observe that

$$\mathsf{Dist}(f, g) = 1 - \frac{1}{|\mathbb{Z}_p|^n} \sum_{x\in\mathbb{Z}_p^n} \left(\frac{1}{|\mathbb{Z}_p|} \sum_{t\in\mathbb{Z}_p} \omega^{t(f(x)-g(x))} \right) = 1 - \frac{1}{|\mathbb{Z}_p|} \sum_{t\in\mathbb{Z}_p} \chi_{tg}(tf). \tag{5}$$

Lemma 15. *For all $P, l : \mathbb{Z}_p^n \to \mathbb{Z}_p$ such that l is linear and $P(tx) = tP(x)$ for all $x \in \mathbb{Z}_p^n$, $t \in \mathbb{Z}_p$,*

$$\mathsf{Dist}(P, l) = \frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|} \left(1 - \chi_l(P)\right).$$

Proof. Note that $tP(x) = P(tx)$ and $tl(x) = l(tx)$ for all $t \in \mathbb{Z}_p$ and $x \in \mathbb{Z}_p^n$. Hence, since multiplication by $t \in \mathbb{Z}_p^*$ induces a permutation of $\mathbb{Z}_p^n$, one has that $\chi_{tl}(tP) = \chi_l(P)$ for all $t \in \mathbb{Z}_p^*$. The conclusion follows from (5) and noting that $\chi_0(\cdot) = 1$. □

The following result establishes the (η,η)–robustness of the **Prime Field Linearity Test**.

Lemma 16. *Let $\mathcal{L}$ be the set of linear functions from $\mathbb{Z}_p^n$ to $\mathbb{Z}_p$ and let $P:\mathbb{Z}_p^n\to\mathbb{Z}_p$ be such that $P(tx)=tP(x)$ for all $x\in\mathbb{Z}_p^n$, $t\in\mathbb{Z}_p$. Then,*

$$\mathsf{Rej}(P,\mathcal{T})=\frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|}\left(1-\sum_{l\in\mathcal{L}}\left(1-\frac{|\mathbb{Z}_p|}{|\mathbb{Z}_p^*|}\,\mathsf{Dist}(P,l)\right)^3\right)\geq\mathsf{Dist}(P,\mathcal{L}).$$

Proof. Note that $\omega^P=\sum_{l\in\mathcal{L}}\chi_l(P)\omega^l$ and that $t(P(x)+P(y)+P(z))=P(tx)+P(ty)+P(tz)$. Hence,

$$\omega^{t(P(x)+P(y)+P(z))}=\sum_{l,l',l''\in\mathcal{L}}\chi_l(P)\chi_{l'}(P)\chi_{l''}(P)\omega^{l(tx)+l'(ty)+l''(tz)}.$$

Furthermore, $z=-(x+y)$ so the linearity of l'' implies that $l''(tz)=-(l''(tx)+l''(ty))$. Thus, (4) yields that

$$\mathsf{Rej}(P,\mathcal{T})=\frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|}-\sum_{l,l',l''\in\mathcal{L}}\chi_l(P)\chi_{l'}(P)\chi_{l''}(P)\left(\frac{1}{|\mathbb{Z}_p|}\sum_{t\in\mathbb{Z}_p^*}\chi_{tl''}(tl)\chi_{tl''}(tl')\right).$$

Moreover, when $t\in\mathbb{Z}_p^*$, $\chi_{tl''}(tl)$ and $\chi_{tl''}(tl')$ equal 1 provided $l=l'=l''$, and $\chi_{tl''}(tl)$ or $\chi_{tl''}(tl')$ equal 0 otherwise. Hence,

$$\mathsf{Rej}(P,\mathcal{T})=\frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|}\left(1-\sum_{l\in\mathcal{L}}\left(\chi_l(P)\right)^3\right).$$

Since $\mathsf{Dist}(P,l)$ is a real number, Lemma 15 implies that so is $\chi_l(P)$. Thus, since $\omega^P=\sum_{l\in\mathcal{L}}\chi_l(P)\omega^l$ and $l(0)=0$ for every $l\in\mathcal{L}$, we know that $1=\omega^{P(0)}=\sum_{l\in\mathcal{L}}\chi_l(P)$. Therefore, there is some $l\in\mathcal{L}$ for which $\chi_l(P)$ is non–negative. It follows that,

$$\mathsf{Rej}(P,\mathcal{T})=\frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|}\left(1-\sum_{l\in\mathcal{L}}\left(\chi_l(P)\right)^3\right)\geq\frac{|\mathbb{Z}_p^*|}{|\mathbb{Z}_p|}\left(1-\operatorname*{Max}_{l\in\mathcal{L}}\chi_l(P)\sum_{l\in\mathcal{L}}\left(\chi_l(P)\right)^2\right).$$

By Lemma 15, $\mathrm{Max}_{l\in\mathcal{L}}\chi_l(P)=1-(|\mathbb{Z}_p|/|\mathbb{Z}_p^*|)\,\mathsf{Dist}(P,\mathcal{L})$. The desired conclusion follows by observing that $\sum_{l\in\mathcal{L}}(\chi_l(P))^2=1$. $\square$

Clearly, the **Prime Field Linearity Test** never rejects a linear function. As far as continuity goes, this is all that usually matters in the PCP context. Note how the verification procedure is simplified both by choosing a prime field structure in which to carry out the work and by forcing structure on the oracle function P, specifically imposing that $P(tx)=tP(x)$ for all $x\in\mathbb{Z}_p^n$ and $t\in\mathbb{Z}_p$. Observe also that the (η,η)–robustness of the test is guaranteed whatever the value of η. Other robustness results discussed in other sections of this work do not exhibit this characteristic. In fact, they typically mean something non–obvious only when η

is small. In the PCP context one prefers test analyses that establish that the probability of rejection increases as the distance between the oracle function and the family of functions of interest grows. The majority and median arguments fail to achieve these type of results. The technique on which the proof of Lemma 16 relies was introduced in [BCH+95] and is based on discrete Fourier analysis. This proof technique, in contrast to the majority and median arguments, does not construct a function which is both close to the oracle function and satisfies the property of interest. Hence, when applying the discrete Fourier analysis technique one does not need to assume that the rejection probability of the test is small as is always the case when applying the majority and median argument.

7.2 Property testing

In the context of testing algebraic functions one is mainly concerned with the problem of determining whether some function to which one has oracle access belongs to some specific class. In the context of property testing one focuses in the case where one has some kind of oracle access to an object, not necessarily a function. Informally, there is an object of which one can ask questions about. The goal is to infer whether or not the object has a specific property. For concreteness, lets consider the following example given by Goldreich [Gol00]: there is a book of which one knows it contains n words and one is allowed to query what its i-th word is — the goal is to determine whether the book is writing in a specific language, say Spanish. As is often the case when testing algebraic functions, if one wants to be completely certain that the book is writing in a specific language one has to query every word. In property testing, as in self–testing, one relaxes the certainty requirement and simply tries to determine whether the object is close or far away from having the property of interest. The notion of distance depends on the problem, e.g., in Goldreich's example, a reasonable choice would be the fraction of non–Spanish words. Thus, suppose that upon seeing one randomly chosen word of the book one decides whether it is writing in Spanish depending on whether the chosen word is a word in such language. Then, a book fully written in Spanish will always be accepted and those books that are at distance δ from being fully written in that language will be discarded with probability δ.

In summary, in property testing one is interested in deciding whether an object has a global property by performing random local checks. One is satisfied if one can distinguish with sufficient confidence between those objects that are close from those that are far from having the global property. In this sense, property testing is a notion of approximation for the aforementioned decision problem.

There are several motivations for the property testing paradigm. When the oracle objects are too large to examine (e.g., the table of a boolean function on a large number of variables) there is no other feasible alternative for deciding whether the object exhibits a given property. Even if the object's size is not large it might be that deciding whether it satisfies the global property is computationally infeasible. In this latter case, property testing provides a reasonable alternative for handling the problem. Finally, when both the oracle object is not too large and the global property can be decided efficiently, property testing might still yield a much faster way of making the correct decision with a high degree of confidence. Moreover, many of the property testers that have been built also allow, at the cost of some additional computational effort, to construct a witness showing the relevant object has the property of interest. These testers could be used to detect instances which are far away from having the property of interest. More expensive computational procedures can thus be run only on instances that have a better chance of having the desired property.

Certainly, exact testing as described earlier in this work can be cast as a property testing problem. Thus, it could be that property testing is a more general paradigm. This is not the case, the two models are mathematically equivalent. One can view property testing as a case of classical testing. However, there are advantages of not doing so, and in recent years the general trend has been to cast new results in the property testing scenario. Indeed, we could have written this whole survey that way. The reasons for not doing so are twofold. The first one is historical: most of the results about algebraic testing were stated in the self–testing context. The second one is specific to this survey. By distinguishing between self–testers, which are algorithms, and (property) tests, which are mathematical objects, we hope that we did clearly point out the difference between the computational and the purely mathematical aspects of the theory. We think that this difference was not adequately dealt with in the previous literature. Had we spoken about property testers and property tests, the difference could have been easily lost for the reader because of the similarity of the terms.

Goldreich, Goldwasser, and Ron [GGR96] were the first to advocate to use of the property testing scenario. In particular they considered the case of testing graph properties. Here, the oracle objects are graphs over a known node set. In [GGR96] the notion of distance between two n–vertex graphs with equal node set is the fraction of edges on which the graphs disagree over n^2. Among the properties considered in [GGR96] were: whether the graph was k–colorable, had a clique containing a ρ fraction of its nodes, had an (edge) cut of size at least ρ fraction of the edges of the complete graph in the same node set, etc. The distance between a graph property is defined in the obvious way, i.e., as the smallest distance between the graph and any graph over the same node set that satisfies the property. In [GR97] a notion of distance better suited to the study of properties of bounded degree graphs was proposed. Specifically, the proposed notion of distance between two n–vertex maximum degree d graphs with equal node set is the fraction of edges on which the graphs disagree over dn. Among the properties studied in [GR97] were: whether the graph was connected, k–vertex–connected, k–edge–connected, planar, etc. Other recent developments in testing graph properties can be found in [GR98, AFKS99, PR99, BR00, GR00].

The works of Goldreich, Goldwasser, and Ron [GGR96, GR97] were influential in shifting the focus from testing algebraic properties of functions to testing non–algebraic properties of different type of objects. Indeed, among other properties/objects that have received attention are: monotonicity of functions [GGLR98, DGL+99], properties of formal languages [AKNS99, New00], geometric properties like clustering [ADPR00, MOP00], and specific properties of quantum gates in quantum circuits [DMMS00].

For surveys on property testing see Goldreich [Gol98] and Ron [Ron00].

References

[ABCG93] S. Ar, M. Blum, B. Codenotti, and P. Gemmell. Checking approximate computations over the reals. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 786–795, San Diego, California, May 1993. ACM.

[ADPR00] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing clustering. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000. (To appear).

[AFKS99] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 656–666, New York City, New York, October 1999. IEEE.

[AKNS99] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 645–655, New York City, New York, October 1999. IEEE.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Pittsburgh, Pennsylvania, October 1992. IEEE. Final version in [ALM+98].

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *J. of the Association for Computing Machinery*, 45(3):505–555, 1998.

[AS92a] N. Alon and J. H. Spencer. *The probabilistic method.* Wiley–Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., first edition, 1992.

[AS92b] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 2–13, Pittsburgh, Pennsylvania, October 1992. IEEE.

[AS97] S. Arora and M. Sudan. Improved low–degree testing and its applications. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 485–495, El Paso, Texas, May 1997. ACM.

[BCH+95] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 432–441, Milwaukee, Wisconsin, October 1995. IEEE.

[BFL90] L. Babai, L. Fortnow, and C. Lund. Non–deterministic exponential time has two–prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 16–25, St. Louis, Missouri, October 1990. IEEE. Final version in [BFL91].

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

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

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

[BK89] M. Blum and S. Kannan. Designing programs that check their work. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 86–97, Seattle, Washington, May 1989. ACM. Final version in [BK95].

[BK95] M. Blum and S. Kannan. Designing programs that check their work. *J. of the Association for Computing Machinery*, 42(1):269–291, 1995.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self–testing/correcting with applications to numerical problems. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, Baltimore, Maryland, May 1990. ACM. Final version in [BLR93].

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self–testing/correcting with applications to numerical problems. *J. of Computer and System Sciences*, 47(3):549–595, 1993.

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

[BR00] M. Bender and D. Ron. Testing acyclicity of directed graphs in sublinear time. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, volume 1853 of *LNCS*, pages 809–820. Springer–Verlag, 2000.

[BS94] M. Bellare and M. Sudan. Improved non–approximability results. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 184–193, Montréal, Québec, Canada, May 1994. ACM.

[BW97] M. Blum and H. Wasserman. Reflections on the Pentium division bug. *IEEE Trans. Comp.*, 26(5):1411–1473, April 1997.

[Cop89] D. Coppersmith. Manuscript. Result described in [BLR90], December 1989.

[DGL+99] Y. Dodis, O. Goldreich, E. Lehman, S. Rsakhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. In *Proceedings of RANDOM'99*, volume 1671 of *LNCS*, pages 97–108. Springer–Verlag, 1999.

[DMMS00] W. van Dam, F. Magniez, M. Mosca, and M. Santha. Self–testing of universal and fault–tolerant sets of quantum gates. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 688–696, Portland, Oregon, May 2000. ACM.

[EKR96] F. Ergün, S. Ravi Kumar, and R. Rubinfeld. Approximate checking of polynomials and functional equations. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 592–601, Burlington, Vermont, October 1996. IEEE.

[Erg95] F. Ergün. Testing multivariate linear functions: Overcoming the generator bottleneck. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 407–416, Las Vegas, Nevada, May 1995. ACM.

[ESK00] F. Ergün, S. Sivakumar, and S. Ravi Kumar. Self–testing without the generator bottleneck. *SIAM J. on Computing*, 29(5):1630–1651, 2000.

[FFT] FFTW is a free collection of fast C routines for computing the Discrete Fourier Transform in one or more dimensions. For more details see `www.fftw.org`.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP–complete. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 2–12, San Juan, Puerto Rico, October 1991. IEEE.

[For95] G. L. Forti. Hyers–Ulam stability of functional equations in several variables. *Aequationes Mathematicae*, 50:143–190, 1995.

[GGLR98] O. Goldreich, S. Goldwasser, E. Lehman, and D. Ron. Testing monotonicity. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 426–435, Palo Alto, California, November 1998. IEEE.

[GGR96] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 339–348, Burlington, Vermont, October 1996. IEEE.

[GLR+91] 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, New Orleans, Louisiana, May 1991. ACM.

[Gol98] O. Goldreich. *Combinatorial property testing — A survey*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 45–60. ACM/AMS, 1998.

[Gol00] O. Goldreich. Talk given at the DIMACS Workshop on Sublinear Algorithms, September 2000.

[GR97] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 406–415, El Paso, Texas, May 1997. ACM.

[GR98] O. Goldreich and D. Ron. A sublinear bipartiteness tester for bounded degree graphs. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 289–298, Dallas, Texas, May 1998. ACM.

[GR00] O. Goldreich and D. Ron. On testing expansion in bounded–degree graphs. Technical Report ECCC TR00–020, Electronic Colloquium on Computational Complexity, 2000. (Available at `www.eccc.uni-trier.de/eccc/`).

[Hås96] J. Håstad. Testing of the long code and hardness of clique. In *Proceedings of the 37nd Annual IEEE Symposium on Foundations of Computer Science*, pages 11–19, Burlington, Vermont, October 1996. IEEE.

[Hås97] J. Håstad. Getting optimal in–approximability results. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, El Paso, Texas, May 1997. ACM.

[HR92] D. H. Hyers and T. M. Rassias. Approximate homomorphisms. *Aequationes Mathematicae*, 44:125–153, 1992.

[Hye41] D. H. Hyers. On the stability of the linear functional equation. *Proceedings of the National Academy of Science, U.S.A.*, 27:222–224, 1941.

[Kiw96] M. Kiwi. *Probabilistically Checkable Proofs and the Testing of Hadamard–like Codes.* PhD thesis, Massachusetts Institute of Technology, February 1996.

[KMS99] M. Kiwi, F. Magniez, and M. Santha. Approximate testing with relative error. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 51–60, Atlanta, Georgia, May 1999. ACM.

[KR97] M. Kiwi and A. Russell. Linearity testing over prime fields. Unpublished manuscript, 1997.

[Lip91] R. J. Lipton. *New directions in testing*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 191–202. ACM/AMS, 1991.

[Mag00a] F. Magniez. *Auto–test pour les calculs approché et quantique*. PhD thesis, Université Paris–Sud, France, 2000.

[Mag00b] F. Magniez. Multi–linearity self–testing with relative error. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *LNCS*, pages 302–313. Springer–Verlag, 2000.

[MOP00] M. Mishra, D. Oblinger, and L. Pirtt. Way–sublinear time approximate (PAC) clustering. Unpublished, 2000.

[New00] I. Newman. Testing of functions that have small width branching programs. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000. (To appear).

[PR99] M. Parnas and D. Ron. Testing the diameter of graphs. In *Proceedings of RANDOM'99*, volume 1671 of *LNCS*, pages 85–96. Springer–Verlag, 1999.

[PS94] A. Polishchuk and D. Spielman. Nearly–linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 194–203, Montréal, Québec, Canada, May 1994. ACM.

[Ron00] D. Ron. Property testing (A tutorial), 2000. (Available at `www.eng.tau.ac.il/~danar/papers.html`). To appear in *Handbook on Randomization*.

[Rub90] R. Rubinfeld. *A mathematical theory of self–checking, self–testing and self–correcting programs.* PhD thesis, University of California, Berkeley, 1990.

[Rub94] R. Rubinfeld. On the robustness of functional equations. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 288–299, Santa Fe, New Mexico, November 1994. IEEE. Final version in [Rub99].

[Rub99] R. Rubinfeld. On the robustness of functional equations. *SIAM J. on Computing*, 28(6):1972–1997, 1999.

[RS92a] T. M. Rassias and P. Šemrl. On the behaviour of mappings which do not satisfy Hyers–Ulam stability. *Proceedings of the American Mathematical Society*, 114(4):989–993, April 1992.

[RS92b] 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 23–32, Orlando, Florida, January 1992. ACM/SIAM. Final version in [RS96].

[RS96] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing*, 25(2):252–271, April 1996.

[Sko83] F. Skopf. Sull'approssimazione delle applicazioni localmente δ–additive. *Atti della Accademia delle Sciencze di Torino*, 117:377–389, 1983. (In Italian.).

[Tre98] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 299–308, Dallas, Texas, May 1998. ACM.