# ERASURE-RESILIENT PROPERTY TESTING*

KASHYAP DIXIT†, SOFYA RASKHODNIKOVA‡, ABHRADEEP THAKURTA§, AND NITHIN VARMA‡

**Abstract.** Property testers form an important class of sublinear-time algorithms. In the standard property testing model, an algorithm accesses the input function $f : \mathcal{D} \mapsto \mathcal{R}$ via an oracle. With very few exceptions, all property testers studied in this model rely on the oracle to provide function values at all queried domain points. However, in many realistic situations, the oracle may be unable to reveal the function values at some domain points due to privacy concerns, or when some of the values get erased by mistake or by an adversary. The testers do not learn anything useful about the function by querying those *erased* points. Moreover, the knowledge of a tester may enable an adversary to erase some of the values so as to increase the query complexity of the tester arbitrarily or, in some cases, make the tester entirely useless.

In this work, we initiate a study of property testers that are resilient to the presence of *adversarially erased* function values. An $\alpha$-erasure-resilient $\varepsilon$-tester is given parameters $\alpha \in [0, 1), \varepsilon \in (0, 1)$, along with oracle access to a function $f$ such that at most an $\alpha$ fraction of function values have been erased. The tester does not know whether a value is erased until it queries the corresponding domain point. The tester has to accept with high probability if there is a way to assign values to the erased points such that the resulting function satisfies the desired property $\mathcal{P}$. It has to reject with high probability if, for every assignment of values to the erased points, the resulting function has to be changed in at least an $\varepsilon$ fraction of the non-erased domain points to satisfy $\mathcal{P}$. Erasure-resilient testing generalizes the standard property testing model of Rubinfeld and Sudan (SIAM J. Comput., 1996) and Goldreich, Goldwasser and Ron (J. ACM, 1998). Compared to the tolerant testing model of Parnas, Ron and Rubinfeld (J. Comput. System Sci., 2006), our model places less stringent requirements on the tester.

We design erasure-resilient property testers for a large class of properties. For some properties, it is possible to obtain erasure-resilient testers by simply using standard testers as a black box. However, for some more challenging properties, all existing algorithms are more likely to query certain points in the domain. If these points are erased, the algorithms break. We give efficient erasure-resilient testers for several important classes of such properties of functions including monotonicity, the Lipschitz property, and convexity. Finally, we show a separation between the standard and erasure-resilient testing. Specifically, we describe a property that can be $\varepsilon$-tested with $O(1/\varepsilon)$ queries in the standard model, whereas testing it in the erasure-resilient model requires a number of queries polynomial in the input size.

**Key words.** sublinear algorithms, property testing, error correction, monotone, convex and Lipschitz functions

**AMS subject classifications.** 68W20, 68W25, 68P10, 68Q87, 68W40

## 1. Introduction.

In this paper, we revisit the question of how sublinear-time algorithms access their input. With very few exceptions, all algorithms studied in the literature on sublinear-time algorithms have *oracle* access to their input[1]. However, in

*A preliminary version of this work has appeared in the Proceedings of the International Colloquium on Automata, Languages and Programming 2016 [23].

†Senior Software Engineer, Jet.com. The work was done while the author was at the Department of Computer Science and Engineering, The Pennsylvania State University. The author was supported in part by NSF CAREER award CCF-0845701. (dixit.kashyap@gmail.com).

‡Department of Computer Science, Boston University. These two authors were supported in part by NSF award CCF-1320814, NSF CAREER award CCF-0845701, NSF award CCF-1422975, Pennsylvania State University College of Engineering Fellowship and Pennsylvania State University Graduate Fellowship. (sofya@bu.edu, nvarma@bu.edu).

§Department of Computer Science, University of California Santa Cruz. The work was done while the author was at Yahoo! Labs. (aguhatha@ucsc.edu).

[1]Sublinear-time algorithms with various distributional assumptions on the input positions that the algorithms access have been investigated, for example, in [34, 4, 37]. There is also a line of work, initiated by [5], that studies sublinear-time algorithms that access distributions, as opposed to fixed

many applications, this assumption is unrealistic. The oracle may be unable to reveal parts of the data due to privacy concerns, or when some of the values get erased by mistake or by an adversary. Motivated by these scenarios, we propose to study sublinear-time algorithms that work with partially erased data.

Formally, we view a dataset as a function over some discrete domain $\mathcal{D}$, such as $[n] = \{1, \ldots, n\}$ or $[n]^d$. For example, the classical problem of testing whether a list of $n$ numbers is sorted in nondecreasing order can be viewed as a problem of testing whether a function $f : [n] \to \mathbb{R}$ is monotone (nondecreasing). Given a parameter $\alpha \in [0, 1)$, we say that a function is *$\alpha$-erased* if at most an $\alpha$ fraction of its domain points are marked as "erased" or protected (that is, an algorithm is denied access to these values). An algorithm that takes an $\alpha$-erased function as its input does not know which values are erased until it queries the corresponding domain points. For each queried point $x$, the algorithm either learns $f(x)$ or, if $x$ is an erased point, gets back a special symbol $\perp$. We study algorithms that work in the presence of *adversarial erasures*. In other words, the query complexity of an algorithm is the number of queries it makes in the *worst case* over all $\alpha$-erased input functions.

In this work, we initiate a systematic study of property testers that are resilient to the presence of adversarial erasures. An $\alpha$-erasure-resilient $\varepsilon$-tester for a property $\mathcal{P}$ is given parameters $\alpha \in [0, 1), \varepsilon \in (0, 1)$, along with oracle access to an $\alpha$-erased function $f$. The tester has to accept with high probability if $f$ can be completed to a function on the whole domain that satisfies the desired property $\mathcal{P}$ and reject with high probability if every completion of $f$ is $\varepsilon$-far from $\mathcal{P}$ on the nonerased part of the domain, that is, every completion of $f$ needs to be changed on at least an $\varepsilon$ fraction of the *nonerased* points in its domain. We stress that the relative distance of a function to a property is measured as a fraction of the nonerased points whose values need to be change in order to satisfy the property. This is because one can always assume that a function is "correct" on the erased points.

*Relationships with other models.* Erasure-resilient property testing model generalizes the standard property testing model of Rubinfeld and Sudan [47] and Goldreich, Goldwasser and Ron [34]. Compared to the tolerant testing model of Parnas, Ron and Rubinfeld [44], our model places less stringent requirements on the tester. We explore the relationship of erasure-resilient testing with these other testing models in Section 7.

We provide (in Section 7.1) a separation between our erasure-resilient model and the standard model, showing that erasure-resilient testing is a strict generalization of standard testing. Specifically, we prove the existence of a property that can be tested with $O(1/\varepsilon)$ queries in the standard model, but requires polynomially many queries in the length of the input in the erasure-resilient model. This result builds on the ideas of Fischer and Fortnow [30] that separate tolerant testing from standard testing.

We discuss the relationship of erasure-resilient testing and tolerant testing in Section 7.2. A tolerant tester for a property $\mathcal{P}$, given two parameters $\varepsilon_1, \varepsilon_2 \in (0, 1)$, where $\varepsilon_1 < \varepsilon_2$, is required to, with probability at least $2/3$, accept inputs that are $\varepsilon_1$-close to $\mathcal{P}$ and reject inputs that are $\varepsilon_2$-far from $\mathcal{P}$. We prove that the existence of tolerant testers implies the existence of erasure-resilient testers with related parameters. Using this implication and existing tolerant testers for sortedness [48], monotonicity [28], and convexity [27], we get erasure-resilient testers for these properties as corollaries. We note that the erasure-resilient testers obtained this way work only for a restricted range of erasures. However, we obtain erasure-resilient testers for the above properties

datasets. In this work, we focus on fixed datasets.

with much better parameters in the technical sections of this article.

Intuitively, the relationship of our erasure-resilient model to tolerant testing is akin to the relationship between error-correcting codes that withstand erasures and error-correcting codes that withstand general errors. We conjecture that erasure-resilient testing can be separated from tolerant testing in the same strong sense as in our separation of standard testing from erasure-resilient testing.

*Generic transformations.* Our first goal while designing erasure-resilient testers is to understand which existing algorithms in the standard property testing model can be easily made erasure-resilient. We show (in Section 2) how to obtain erasure-resilient testers for some properties by using standard testers for these properties as black box. Our transformations apply to *sample-based testers*, which are testers that query uniformly and independently sampled points[2]. More specifically, our first transformation works for *proximity oblivious testers* (POTs) [36] that are, in addition, restricted to be sample-based. Our second transformation applies to sample-based testers for a class of properties that we call *extendable*. Loosely speaking, a property is extendable if (1) a function satisfying the property on a subdomain can be extended to a function satisfying the same property on the whole domain, and (2) a function that is $\varepsilon$-far from the property on a subdomain cannot be extended to a function on the whole domain that satisfies the property without changing the values on at least an $\varepsilon$ fraction of positions on the subdomain. Extendable properties are a generalization of a class of properties defined by Jha and Raskhodnikova [40] and are formally defined in Definition 2.6. Using our second transformation, we are able to obtain erasure-resilient testers for properties such as being a low-degree univariate polynomial [47], monotonicity over general poset domains [32], convexity of black and white images [10], and Boolean functions over $[n]$ with at most $k$ runs of 0s and 1s.

*Erasure-resilient testers for more challenging properties.* One challenge in designing erasure-resilient testers by using existing algorithms in the standard model as a starting point is that many existing algorithms are more likely to query certain points in the domain. Therefore, if these points are erased, the algorithms break. Specifically, the previously known optimal algorithms for testing whether a list of numbers is sorted (and there are at least three different algorithms for this problem [26, 12, 18]) have this feature. Moreover, it is known that an algorithm that makes uniformly random queries is far from optimal: it needs $\Theta(\sqrt{n})$ queries instead of $\Theta(\log n)$ for $n$-element lists [26, 29].

There are a number of well studied properties for which all known optimal algorithms heavily rely on querying specific points. Most prominent examples include monotonicity, the Lipschitz properties and, more generally, bounded-derivative properties of real-valued functions on $[n]$ and $[n]^d$, as well as convexity of real-valued functions on $[n]$. It is especially challenging to deal with real-valued functions in our model, because there are many possibilities for erased values. We give efficient erasure-resilient testers for all aforementioned properties of real-valued functions in Sections 3-6.

**1.1. The Erasure-Resilient Testing Model.** We formalize our erasure-resilient model for the case of property testing. Erasure-resilient versions of other computational models, such as tolerant testing, can be defined analogously.

[2]Sample-based testers were first considered by Goldreich, Goldwasser and Ron [34]. A systematic study of sample-based testers was initiated by Goldreich and Ron [37] and continued by Fischer et al. [31]. Sample-based testers are also called uniform testers in some other papers on testing specific properties [10, 9, 11].

DEFINITION 1.1 ($\alpha$-erased function). *Let $\mathcal{D}$ be a domain, $\mathcal{R}$ be a range, and $\alpha \in [0,1)$. A function*[3] *$f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$ is* $\alpha$-erased *if $f$ evaluates to $\perp$ on at most an $\alpha$ fraction of domain points. The points on which $f$ evaluates to $\perp$ are called* erased. *The set of remaining (nonerased) points is denoted by $\mathcal{N}$.*

Note that an $\alpha$-erased function has *at most* an $\alpha$ fraction of its values erased. In particular, a function with no erasures is also an $\alpha$-erased function for all $\alpha \in [0,1)$.

A function $f' : \mathcal{D} \to \mathcal{R}$ that differs from a function $f$ only on points erased in $f$ is called a *completion* of $f$. The (Hamming) distance of an $\alpha$-erased function $f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$ from a property (set) $\mathcal{P}$ is the minimum number of points in $\mathcal{N}$ on which every completion of $f$ needs to be changed to satisfy $\mathcal{P}$. The relative Hamming distance of $f$ from $\mathcal{P}$ is the aforementioned quantity normalized by $|\mathcal{N}|$. An $\alpha$-erased function $f$ is $\varepsilon$-far from a property $\mathcal{P}$ if the relative Hamming distance of $f$ from $\mathcal{P}$ is at least $\varepsilon$.

DEFINITION 1.2 (Erasure-resilient tester). *An* $\alpha$-erasure-resilient $\varepsilon$-tester *of property $\mathcal{P}$ gets input parameters $\alpha \in [0,1)$, $\varepsilon \in (0,1)$ and oracle access to an $\alpha$-erased function $f : \mathcal{D} \to \mathcal{R} \cup \{\perp\}$. It outputs, with probability*[4] *at least 2/3,*

- **accept** *if there is a completion $f' : \mathcal{D} \to \mathcal{R}$ of $f$ that satisfies $\mathcal{P}$;*
- **reject** *if every completion $f' : \mathcal{D} \to \mathcal{R}$ of $f$ needs to be changed on at least an $\varepsilon$ fraction of $\mathcal{N}$, the nonerased portion of $f$'s domain, to satisfy $\mathcal{P}$ (that is, $f'$ is $\varepsilon \cdot \frac{|\mathcal{N}|}{|\mathcal{D}|}$-far from $\mathcal{P}$).*

*The tester has* 1-sided error *if the first item holds with probability 1. The tester is* nonadaptive *if the queries made by the tester do not depend on the answers to the previous queries, and* adaptive *otherwise.*

Let $f_{|\mathcal{N}}$ denote the function $f$ restricted to the set $\mathcal{N}$ of nonerased points. We show (in Section 2) that if property $\mathcal{P}$ is extendable (Definition 2.6), we can define a property $\mathcal{P}_{\mathcal{N}}$ such that the erasure-resilient tester for $\mathcal{P}$ is simply required to distinguish the case that $f_{|\mathcal{N}}$ satisfies $\mathcal{P}_{\mathcal{N}}$ from the case that $f_{|\mathcal{N}}$ is $\varepsilon$-far from satisfying $\mathcal{P}_{\mathcal{N}}$. For example, if $\mathcal{P}$ is monotonicity of functions on a partially-ordered domain $\mathcal{D}$ then $\mathcal{P}_{\mathcal{N}}$ is monotonicity of functions on $\mathcal{N}$. (Most of the properties we consider in this article, including monotonicity, Lipschitz properties and convexity, are extendable properties.) Note that, even for the case of extendable properties, our problem is different from the standard property testing problem because the tester does not know in advance which points are erased.

**1.2. Relationships With Other Testing Models.** In this section, we state our results that place erasure-resilient testing in between standard testing [47, 34] and tolerant testing [44].

*Connections to standard testing.* Erasure-resilient testing is a generalization of standard property testing [47, 34]. For functions with no erasures, we can set $\alpha = 0$ in Definition 1.2 and obtain the definition of a standard tester. In Section 7.1, we prove the following theorem which shows that erasure-resilient testing is a strict generalization of standard testing even for properties of Boolean strings.

[3] Any object can be viewed as a function. E.g., an $n$-element array of real numbers can be viewed as a function $f : [n] \to \mathbb{R}$, an image—as a map from the plane to the set of colors, and a graph—as a map from the set of vertex pairs to $\{0,1\}$.

[4] In general, the error probability can be any $\delta \in (0,1)$. For simplicity, we formulate our model and the results with $\delta = 1/3$. To get results for general $\delta$, by standard arguments, it is enough to multiply the complexity of an algorithm by $\log 1/\delta$.

THEOREM 1.3. *There exists a property $R$ on Boolean strings of length $n$ such that for all large enough $n$, the property $R$ can be $\varepsilon$-tested in the standard model using $O(1/\varepsilon)$ queries. However, there exists some $c > 0$ such that for large enough $n$ and for all constant $\alpha \in (0,1)$, every $\alpha$-erasure-resilient $\frac{1}{4}$-tester for $R$ has to make at least $n^c$ queries.*

The techniques used to prove Theorem 1.3 is nearly identical to the techniques used by Fischer and Fortnow [30] to separate tolerant testing from standard testing. However, our result is stronger than the result in [30], as erasure-resilient testing is at least as easy as tolerant testing.

*Connections to tolerant testing and distance approximation.* Tolerant testers were defined and studied by Parnas, Ron and Rubinfeld [44]. An algorithm is an $(\varepsilon_1, \varepsilon_2)$-*tolerant tester* for a property $\mathcal{P}$ if, when given oracle access to a function $f$, the algorithm (i) accepts with probability at least $2/3$ if $f$ is $\varepsilon_1$-close to $\mathcal{P}$ and (ii) rejects with probability at least $2/3$ if $f$ is $\varepsilon_2$-far from $\mathcal{P}$, where $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$. The algorithm is *fully tolerant* if it works as above for all $\varepsilon_1 < \varepsilon_2$, which are given as the inputs. We show (in Section 7.2) that the existence of a fully tolerant tolerant tester for a property implies the existence of an $\alpha$-erasure-resilient $\varepsilon$-tester for that property for some settings of $\varepsilon$ and $\alpha$.

THEOREM 1.4. *If $A$ is a fully tolerant tester for a property $\mathcal{P}$ of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$, then there exists an $\alpha$-erasure-resilient $\varepsilon$-tester for $\mathcal{P}$ that has the same query complexity as $A$ and works for all $\alpha \in [0,1), \varepsilon \in (0,1)$ such that $\alpha < \frac{\varepsilon}{1+\varepsilon}$.*

In Section 7.2, we apply a different version of Theorem 1.4 to algorithms that have oracle access to a function and approximate its distances to sortedness [48], monotonicity [28], and convexity [27], and obtain erasure-resilient testers for these properties.

**1.3. Properties That We Study.** Next we define properties of real-valued functions considered in this article and summarize previous work on testing them. Most properties of real-valued functions studied in the property testing framework are for functions over the *line* domain $[n]$ and, more generally, the *hypergrid* domain $[n]^d$.

DEFINITION 1.5 (Hypergrid, line). *Given $n, d \in \mathbb{N}$, the hypergrid of size $n$ and dimension $d$ is the set $[n]^d$ associated with an order relation $\preceq$, such that $x \preceq y$ for all $x, y \in [n]^d$ iff $x_i \leq y_i$ for all $i \in [d]$, where $x_i$ (respectively $y_i$) denotes the $i^{th}$ coordinate of $x$ (respectively, $y$). The special cases $[n]$ and $[2]^d$ are called the* line *and* hypercube*, respectively.*

We consider domains that are subsets of $[n]^d$ to be able to handle arbitrary erasures on $[n]^d$.

*Monotonicity.* Monotonicity of functions, first studied in the context of property testing in [33], is one of the most widely investigated properties in this model [26, 24, 42, 32, 1, 29, 39, 6, 44, 2, 12, 15, 13, 18, 19, 14, 17]. A function $f : \mathcal{D} \mapsto \mathbb{R}$, defined on a partially ordered domain $\mathcal{D}$ with order $\preceq$, is monotone if $x \preceq y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{D}$. The query complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [26, 29]; for functions $f : [n]^d \mapsto \mathbb{R}$, it is $\Theta(d \log n/\varepsilon)$ [18, 19], and for functions over arbitrary partially ordered domains $\mathcal{D}$, it is $O(\sqrt{|\mathcal{D}|/\varepsilon})$ [32].

*Lipschitz properties.* Lipschitz continuity is defined for functions between arbitrary metric spaces, but was specifically studied for real-valued functions on hypergrid domains [40, 3, 18, 22, 14, 17] because of applications to privacy [40, 22]. For $\mathcal{D} \subseteq [n]^d$

and $c \in \mathbb{R}$, a function $f : \mathcal{D} \mapsto \mathbb{R}$ is $c$-Lipschitz if $|f(x) - f(y)| \leq c \cdot ||x - y||_1$ for all $x, y \in \mathcal{D}$, where $||x - y||_1$ is the $L_1$ distance between $x$ and $y$. More generally, $f$ is $(\alpha, \beta)$-Lipschitz, where $\alpha < \beta$, if $\alpha \cdot ||x - y||_1 \leq |f(x) - f(y)| \leq \beta \cdot ||x - y||_1$ for all $x, y \in [n]^d$. All $(\alpha, \beta)$-Lipschitz properties can be tested with $O(d \log n/\varepsilon)$ queries [18].

*Bounded derivative properties (BDPs).* The class of BDPs, defined by Chakrabarty et al. [17], is a natural generalization of monotonicity and the $(\alpha, \beta)$-Lipschitz properties. An ordered set $\mathbf{B}$ of $2d$ functions $l_1, u_1, l_2, u_2, \ldots, l_d, u_d : [n-1] \mapsto \mathbb{R} \cup \{\pm\infty\}$ is a *bounding family* if for all $r \in [d]$ and $y \in [n-1]$, $l_r(y) < u_r(y)$. Let $\mathbf{B}$ be a bounding family of functions and let $\mathbf{e}_r$ be the unit vector along dimension $r$. The property $\mathcal{P}(\mathbf{B})$ of being $\mathbf{B}$-*derivative bounded* is the set of functions $f : [n]^d \mapsto \mathbb{R}$ such that $l_r(x_r) \leq f(x + \mathbf{e}_r) - f(x) \leq u_r(x_r)$ for all $r \in [d]$ and $x \in [n]^d$ with $x_r \neq n$, where $x_r$ is the $r^{\text{th}}$ coordinate of $x$.

Consider a graph $\mathcal{H}$ with vertex set $[n]^d$ and edges in both directions between every pair of points in $[n]^d$ that differ in exactly one coordinate. Then the value $u_r(x_r)$ is the upper bound on the increase in function value along the edge $(x, x + e_r)$ and $-l_r(x_r)$ is the upper bound on the increase in function value along the edge $(x + e_r, x)$. A bounding family $\mathbf{B} = \{l_1, u_1, \ldots, l_d, u_d\}$ defines a quasi-metric

$$\mathfrak{m}_{\mathbf{B}}(x, y) := \sum_{r : x_r > y_r} \sum_{t=y_r}^{x_r - 1} u_r(t) - \sum_{r : x_r < y_r} \sum_{t=x_r}^{y_r - 1} l_r(t)$$

over points $x, y \in [n]^d$. In [17], the authors observe that for $\mathcal{D} = [n]^d$, a function $f : \mathcal{D} \mapsto \mathbb{R}$ satisfies $\mathcal{P}(\mathbf{B})$, the bounded derivative property defined by $\mathbf{B}$, iff $\forall x, y \in \mathcal{D}$, $f(x) - f(y) \leq \mathfrak{m}_{\mathbf{B}}(x, y)$. To get an intuition about this observation, note that $\mathfrak{m}_{\mathbf{B}}(x, y)$ is the upper bound dictated by the functions in $\mathbf{B}$ on the amount by which the value of $f$ can increase along a shortest path from $y$ to $x$ in the graph $\mathcal{H}$. We use this characterization as our definition of BDPs for functions over arbitrary $\mathcal{D} \subseteq [n]^d$.

The bounding family for monotonicity is obtained by setting $l_r(y) = 0$ and $u_r(y) = \infty$ for all $r \in [d]$, and for the $(\alpha, \beta)$-Lipschitz property, by setting $l_r(y) = \alpha$ and $u_r(y) = \beta$ for all $r \in [d]$. In general, different bounding families allow a function to be monotone in one dimension, $(\alpha, \beta)$-Lipschitz in another dimension and so on. Chakrabarty et al. [17] showed that for every BDP $\mathcal{P}$, the complexity of testing $\mathcal{P}$ for functions $f : [n]^d \mapsto \mathbb{R}$ is $\Theta(d \log n/\varepsilon)$.

*Convexity of functions.* A function $f : \mathcal{D} \mapsto \mathbb{R}$ is convex if $f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ and $t \in [0, 1]$. If $\mathcal{D} \subseteq [n]$, equivalently, $f$ is convex if $\frac{f(y)-f(x)}{y-x} \leq \frac{f(z)-f(y)}{z-y}$ for all $x < y < z$. Parnas, Ron and Rubinfeld [43] gave a convexity tester for functions $f : [n] \mapsto \mathbb{R}$ with query complexity $O(\log n/\varepsilon)$. Blais, Raskhodnikova and Yaroslavtsev [14] gave an $\Omega(\log n)$ bound for nonadaptive testers for this problem.

**1.4. Our Results.** We give efficient erasure-resilient testers for all properties discussed in Section 1.3. All our testers have optimal complexity for the case with no erasures and have an additional benefit of not relying too heavily on the value of the input function at any specific point.

*Monotonicity on the line.* We start by giving (in Section 3) an erasure-resilient monotonicity tester on $[n]$.

THEOREM 1.6 (Monotonicity tester on the line). *There exists a one-sided error $\alpha$-erasure-resilient $\varepsilon$-tester for monotonicity of real-valued functions on the line $[n]$ that works for all $\alpha \in [0, 1), \varepsilon \in (0, 1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Without erasure resilience, the complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [26, 29]. Thus, the query complexity of our erasure-resilient tester has optimal dependence on the domain size and on $\varepsilon$.

The starting point of our algorithm is the tester for sortedness from [26]. This tester picks a random element of the input array and performs a binary search for that element. It rejects if the binary search does not lead to the right position. The first challenge is that the tester always queries the middle element of the array and is very likely to query other elements that are close to the root in the binary search tree. So, it will break if these elements are erased. To make the tester resilient to erasures, we randomize the binary tree with respect to which it performs the binary search. The second challenge is that the tester does not know which points are erased. To counteract that, our tester samples points from appropriate intervals until it encounters a nonerased point.

Bounding the expected query complexity of our tester (Claim 3.2) is the most interesting part of the analysis. We view the tester as performing a binary search for a uniformly random nonerased point in the array (obtained via sampling), where at every step of the binary search, the nonerased point that *guides* the search at that step is sampled uniformly at random from the interval at that step. The intuition behind our analysis is that such a randomized binary search for a uniformly random search point is biased towards visiting intervals containing a larger fraction of nonerased points. In other words, conditioned on picking a specific nonerased point to split the current interval, the probability of the search point being in the left (or right) subinterval of the current interval is proportional to the fraction of nonerased points in that subinterval.

*BDPs on the hypergrid.* In Sections 4-5, we generalize our monotonicity tester in two ways: (1) to work over general hypergrid domains, and (2) to apply to all BDPs. We achieve it by giving (1) a reduction from testing BDPs on the line to testing monotonicity on the line that applies to erasure-resilient testers and (2) an erasure-resilient version of the dimension reduction from [17].

THEOREM 1.7 (BDP tester on the hypergrid). *For every BDP $\mathcal{P}$ of real-valued functions on the hypergrid $[n]^d$, there exists a one-sided error $\alpha$-erasure-resilient $\varepsilon$-tester that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/970d$, with query complexity $O\left(\frac{d \log n}{\varepsilon(1-\alpha)}\right)$.*

Every known tester of a BDP for real-valued functions over hypergrid domains work by sampling an *axis-parallel line* uniformly at random and checking for violations on the sampled line. Our erasure-resilient testers also follow this paradigm. To check for violations on the sampled line, we use one iteration of our BDP tester for the line. We show (in Section 5.4) the existence of $\alpha$-erased functions $f : \{0,1\}^d \mapsto \mathbb{R}$ that are $\varepsilon$-far from monotone for $\alpha = \Theta(\varepsilon/\sqrt{d})$ but do not have violations to monotonicity along any of the axis parallel lines (which are the edges of the hypercube, in this case). It implies that every tester for monotonicity that follows the paradigm above will fail when $\alpha = \Omega(\varepsilon/\sqrt{d})$. Thus, some restriction on $\alpha$ in terms of $d$ and $\varepsilon$ is necessary for such testers.

*Convexity on the line.* Finally, in Section 6, we develop additional techniques to design a tester for convexity (which is not a BDP) on the line. The query complexity of our tester has the same dependence on $n$ and $\varepsilon$ as in the standard convexity tester of Parnas et al. [43]. The dependence on $n$ is optimal for nonadaptive testers [14], and the tester from [43] is conjectured to be optimal in the standard model.

THEOREM 1.8 (Convexity tester on the line). *There exists a one-sided error $\alpha$-erasure-resilient $\varepsilon$-tester for convexity of real-valued functions on the line $[n]$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Our algorithm for testing convexity combines ideas on testing convexity from [43], testing sortedness from [26], and our idea of randomizing the search. The tester of [43] traverses a uniformly random path in a binary tree on the array $[n]$ by selecting one of the half-intervals of an interval uniformly at random at each step. Instead of doing this, our tester samples a uniformly random nonerased search point and traverses the path to that point in a random binary search tree just as in our modification of the tester of [26]. This is done to bias our algorithm to traverse paths containing intervals that have a larger fraction of nonerased points. However, instead of checking whether the selected point can be found, as in our monotonicity tester, the convexity tester checks a more complicated "goodness condition" in each visited interval of the binary search tree. It boils down to checking that the slope of the functions between pairs of carefully selected points satisfies the convexity condition. In addition to spending queries on erased points due to sampling, like in the monotonicity tester, our tester also performs "walking queries" to find the nearest nonerased points to the left and to the right of the pivots in our random binary search tree. We show that the overhead in the query complexity due to querying erased points is at most a factor of $O(1/(1-\alpha))$.

**2. Generic transformations.** In this section, we present our transformations that make two classes of testers erasure-resilient. Our transformations apply to (1) Proximity Oblivious Testers (POTs) that are additionally restricted to be sample-based (Theorem 2.3), and (2) sample-based testers for extendable properties (Theorem 2.8).

Recall that a tester is called sample-based if its queries are distributed uniformly and independently at random. A one-sided error sample-based tester always accepts functions that satisfy the specified property. Sample-based testers were first considered by Goldreich, Goldwasser, and Ron [34] and systematically studied by Goldreich and Ron [37] and Fischer et al. [31]. In particular, [37, 31] show that certain types of query-based testers yield sample-based testers with sublinear (but dependent on the size of the input) sample complexity.

**2.1. Sample-based POTs.** In this section, we give a simple transformation that makes every POT that queries uniformly and independently random domain points erasure-resilient. POTs were defined by Goldreich and Ron [36] and further studied by Goldreich and Kaufman [35] and Goldreich and Shinkar [38]. We first define POTs.

DEFINITION 2.1 (POT, [38]). *Let $\mathcal{P}$ be a property, let $\rho : (0,1] \mapsto (0,1]$ be a monotone function and let $c \in (0,1]$ be a constant. A tester $T$ is a $(\rho, c)$-*POT *for $\mathcal{P}$ if*

- *for every function $f \in \mathcal{P}$, the probability that $T$ accepts $f$ is at least $c$, and*
- *for every function $f \notin \mathcal{P}$, the probability that $T$ accepts $f$ is at most $c - \rho(\varepsilon_f)$, where $\varepsilon_f$ denotes the relative Hamming distance of $f$ to $\mathcal{P}$.*

It is important to note that POTs in general can query arbitrary domain points. Next, we define sample-based POTs, a restriction of POTs.

DEFINITION 2.2 (Sample-based POT). *A POT whose queries are distributed uniformly and independently at random is called a sample-based POT.*

Erasure-resilient versions of POTs and sample-based POTs can be defined analogously. Next, we state our generic transformation for sample-based POTs.

THEOREM 2.3. *If $T$ is a sample-based $(\rho,c)$-POT for a property $\mathcal{P}$ that makes $q$ queries, then there exists a sample-based $\alpha$-erasure-resilient $(\rho',c)$-POT $T'$ for $\mathcal{P}$ that works for all $\alpha < \rho(\varepsilon_f\cdot(1-\alpha))/q$ and makes $q$ queries, where $\rho'(x) = \rho(x\cdot(1-\alpha))-\alpha\cdot q$ for all $x \in (0,1]$.*

*Proof.* Let $\mathcal{P}$ be a property of functions over a domain $\mathcal{D}$. The tester $T'$ queries $q$ uniform and independent points from $\mathcal{D}$. It accepts if the sample has an erased point. Otherwise, it runs $T$ on the $q$ sampled nonerased points and accepts if and only if $T$ accepts.

Consider an $\alpha$-erased function $f \in \mathcal{P}$ and a completion $f^r \in \mathcal{P}$. The tester $T$ accepts $f^r$ with probability at least $c$. If $T$ accepts $f^r$ on querying a sample $S \subseteq \mathcal{D}$, then $T'$ also accepts $f$ on $S$. Thus, the probability that $T'$ accepts $f$ is at least $c$.

A tuple $W \in \mathcal{D}^q$ is a *witness* for a function $g \notin \mathcal{P}$, if $T$ rejects $g$ upon sampling $W$. Consider an $\alpha$-erased function $f \notin \mathcal{P}$. Every completion $f^r$ of $f$ is $\varepsilon_f(1-\alpha)$-far from $\mathcal{P}$. Since $T$ rejects $f^r$ with probability at least $1-c+\rho(\varepsilon_f(1-\alpha))$, at least $(1-c+\rho(\varepsilon_f(1-\alpha)))\cdot|\mathcal{D}|^q$ tuples in $\mathcal{D}^q$ are witnesses for $f^r$. Erasing one point can affect at most $q\cdot|\mathcal{D}|^{q-1}$ witnesses. Thus, erasing an $\alpha$ fraction of points can affect at most $\alpha\cdot q\cdot|\mathcal{D}|^q$ witnesses. At least $(1-c+\rho(\varepsilon_f(1-\alpha))-\alpha\cdot q)\cdot|\mathcal{D}|^q$ out of $|\mathcal{D}|^q$ tuples are witnesses with no points (in them) erased. The probability that $T'$ samples such a tuple (and rejects $f$) is at least $1-c+\rho(\varepsilon_f(1-\alpha))-\alpha\cdot q = 1-c+\rho'(\varepsilon_f)$. Hence, the probability that $T'$ accepts $f$ is at most $c-\rho'(\varepsilon_f)$. This probability is less than $c$ for all $\alpha < \rho(\varepsilon_f(1-\alpha))/q$. ∎

*Low degree univariate polynomials.* We apply Theorem 2.3 to a POT designed by Rubinfeld and Sudan [47] for the property of being a univariate polynomial of degree at most $d$ over a finite field $\mathbb{F}$ and get an $\alpha$-erasure-resilient $\varepsilon$-tester for this property. Consider a function $f : \mathbb{F} \mapsto \mathbb{F}$ that we would like to test for being a univariate polynomial of degree at most $d$. The tester from [47] selects $d+2$ points uniformly and independently at random from $\mathbb{F}$ and checks whether there is a univariate polynomial of degree at most $d$ that fits all these points (by interpolation). It accepts if there is such a polynomial and rejects otherwise. Call this tester $T$. It is evident that $T$ always accepts univariate polynomials of degree at most $d$. The authors of [47] also prove that $T$ rejects with probability at least $\varepsilon_f$ if $f$ is not a univariate polynomial of degree at most $d$, where $\varepsilon_f$ denotes the relative Hamming distance of $f$ from the property. Therefore, $T$ is a sample-based $(\rho,1)$-POT for this property, where $\rho$ is the identity function. By Theorem 2.3, there exists a sample-based $\alpha$-erasure-resilient $(\rho',1)$-POT, say $T'$, that makes $d+2$ queries, where $\rho'(x) = x(1-\alpha)-\alpha\cdot(d+2)$. The probability that $T'$ rejects an $\alpha$-erased function $f$ that is $\varepsilon$-far from being a univariate polynomial of degree at most $d$ is at least $\varepsilon(1-\alpha)-\alpha\cdot(d+2)$. The corollary follows.

COROLLARY 2.4. *There exists a sample-based $\alpha$-erasure-resilient $\varepsilon$-tester for the property of being a univariate polynomial of degree at most $d$ over a finite field $\mathbb{F}$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$, where $\alpha < \frac{\varepsilon}{d+2+\varepsilon}$, with query complexity $O\left(\frac{d+2}{\varepsilon(1-\alpha)-\alpha\cdot(d+2)}\right)$.*

**2.2. Sample-based testers for extendable properties.** In this section, we define extendable properties and present our generic transformation for sample-based testers of such properties. First, we define the extension of a function.

DEFINITION 2.5 (Extension of a function). *Given $\mathcal{S} \subseteq \mathcal{T}$, the* extension *of a*

*function $f : \mathcal{S} \mapsto \mathcal{R}$ to a domain $\mathcal{T}$ is a function $g : \mathcal{T} \mapsto \mathcal{R}$ that agrees with $f$ on every point in $\mathcal{S}$.*

Our definition of extendable properties is a generalization of the notion of *edge-transitive properties that allow extension* by Jha and Raskhodnikova [40]. A property is edge-transitive if it is fully characterized by predicates on pairs of domain points. Such a property allows extension if every function that satisfies the property on a subdomain can be extended to one that satisfies the property on the whole domain. We now define extendable properties.

DEFINITION 2.6 (Extendable property). *For a domain $\mathcal{D}$ and all $\mathcal{S} \subseteq \mathcal{D}$, let $\mathcal{P}_{\mathcal{S}}$ denote a set of functions over domain $\mathcal{S}$. The property $\bigcup_{\mathcal{S} \subseteq \mathcal{D}} \mathcal{P}_{\mathcal{S}}$ is extendable if, for all $\mathcal{S}, \mathcal{T} : \mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{D}$,*

1. *for every function $f \in \mathcal{P}_{\mathcal{S}}$, there is an extension $f' \in \mathcal{P}_{\mathcal{T}}$, and*
2. *for every function $f$ that is $\varepsilon$-far from $\mathcal{P}_{\mathcal{S}}$, every function $f' \in \mathcal{P}_{\mathcal{T}}$ differs from $f$ on at least an $\varepsilon$ fraction of points in $\mathcal{S}$.*

We now give examples of some properties that are extendable and some that are not. A lot of properties that we deal with, in this paper, are edge-transitive properties that allow extension. Monotonicity over arbitrary partial orders, $(\alpha, \beta)$-Lipschitz properties over hypergrids, and more generally BDPs over hypergrids are all edge-transitive properties that allow extension. However, the class of properties for which we are able to design erasure-resilient testers is a strict superset of edge-transitive properties that allow extension. One such property is convexity of real-valued functions over the domain $[n]$. Another one is the property of Boolean functions over $[n]$ whose value alternates between 0 and 1 at most $k$ times when moving from domain point 1 to domain point $n$. Both properties are not edge-transitive, but are extendable.

We now describe two properties defined over $[n]$ that are not extendable. The first among these, denoted $\mathcal{P}'$, is equal to $\bigcup_{I \subseteq [n]} \mathcal{P}'_I$, where $\mathcal{P}'_I$ for $I \subseteq [n]$ is the set of all integer-valued functions that are strictly increasing w.r.t. the ordering on points in $I$. The property $\mathcal{P}'$ is not extendable, since it does not satisfy the first condition in Definition 2.6. To see this, consider the function $f : \{1, 3\} \mapsto \mathbb{Z}$ such that $f(1) = 1, f(3) = 2$. Clearly, $f$ belongs to $\mathcal{P}'_{\{1,3\}}$ and hence to $\mathcal{P}'$. But $f$ cannot be extended to $\{1, 2, 3\}$ while satisfying $\mathcal{P}'$. The second property, denoted $\hat{\mathcal{P}}$, is equal to $\bigcup_{I \subseteq [n]} \hat{\mathcal{P}}_I$. A function $f : I \mapsto [n]$ is in $\hat{\mathcal{P}}_I$ for $I \subseteq [n]$, if for each $i \in I$, we have $f(i) \leq |\{j \in I : j \leq i\}|$. In other words, a function satisfies $\hat{\mathcal{P}}_I$ if the value of the function at each point $i$ is at most the number of points $j \leq i$ where it is defined. It is easy to see that this property satisfies the first condition in Definition 2.6. We will show that it does not satisfy the second condition. Consider the function $f$ defined on $\{1, 3\}$, where $f(1) = 1$ and $f(3) = 3$. This function is 1/2-far from being in $\hat{\mathcal{P}}_I$. But this can be extended to the function $g$ over $[n]$ as $g(i) = i$ for all $i \in [n]$. Clearly, $g$ satisfies $\hat{\mathcal{P}}$, violating the second condition in Definition 2.6.

We are now ready to describe our generic transformation for extendable properties. In what follows, we will be talking about functions defined over various domains, their extensions, and completions[5]. The next lemma is used in the proof of our generic

[5]When we say that $g : \mathcal{T} \mapsto \mathcal{R}$ is an extension of $f : \mathcal{S} \mapsto \mathcal{R}$, we mean that $g$ and $f$ are functions defined on different domains $\mathcal{T}$ and $\mathcal{S}$ and that $\mathcal{S} \subseteq \mathcal{T}$. The function $f$ is *not defined* on the set $\mathcal{T} \setminus \mathcal{S}$ and $f(x) = g(x)$ for all $x \in \mathcal{S}$. The functions $f$ and/or $g$ may or may not have some of their function values erased. On the other hand, when we say that $g$ is a completion of $f$, it must be the case that $f$ and $g$ are functions defined on the same domain, say $\mathcal{T}'$, with $f$ being (possibly) erased,

transformation.

LEMMA 2.7. *Let $\bigcup_{\mathcal{S}\subseteq\mathcal{D}} \mathcal{P}_{\mathcal{S}}$ be an extendable property. Consider an $\alpha$-erased function $f$ over domain $\mathcal{D}$ and let $\mathcal{N} \subseteq \mathcal{D}$ be the set of nonerased points in it. If $f \in \mathcal{P}_{\mathcal{D}}$, then $f_{|\mathcal{N}} \in \mathcal{P}_{\mathcal{N}}$. If $f$ is $\varepsilon$-far from $\mathcal{P}_{\mathcal{D}}$, then $f_{|\mathcal{N}}$ is $\varepsilon$-far from $\mathcal{P}_{\mathcal{N}}$.*

*Proof.* Suppose that $f \in \mathcal{P}_{\mathcal{D}}$. Let $f_* \in \mathcal{P}_{\mathcal{D}}$ be a completion of $f$. As $f_*$ is a function defined over $\mathcal{D}$ and $f_*$ has no erasures, $f_*$ is an extension of $f_{|\mathcal{N}}$. Therefore, $f_{|\mathcal{N}} \in \mathcal{P}_{\mathcal{N}}$, since $\bigcup_{\mathcal{S}\subseteq\mathcal{D}} \mathcal{P}_{\mathcal{S}}$ is an extendable property.

Now, suppose that $f$ is $\varepsilon$-far from $\mathcal{P}_{\mathcal{D}}$. Then, every completion of $f$ needs to be changed in at least an $\varepsilon$ fraction of nonerased points to satisfy $\mathcal{P}_{\mathcal{D}}$. Assume for the sake of contradiction that the relative Hamming distance of $f_{|\mathcal{N}}$ to $\mathcal{P}_{\mathcal{N}}$ is $\varepsilon' < \varepsilon$. Let $g$ be the function in $\mathcal{P}_{\mathcal{N}}$ closest to $f_{|\mathcal{N}}$. Let $g^e$ be an extension of $g$ to $\mathcal{D}$ that satisfies $\mathcal{P}_{\mathcal{D}}$. Define an extension of $f_{|\mathcal{N}}$ to $\mathcal{D}$, say $f^e$ as follows. The function $f^e$ takes the same values as $f_{|\mathcal{N}}$ on points in $\mathcal{N}$ and takes the same values as $g^e$ on the remaining points. Note that $f^e$ is a completion of $f$ as well. Clearly, $f^e$ can be made to satisfy $\mathcal{P}_{\mathcal{D}}$ by changing an $\varepsilon' < \varepsilon$ fraction of points on $\mathcal{N}$, which contradicts the assumption that $f$ is $\varepsilon$-far from $\mathcal{P}_{\mathcal{D}}$. ∎

Our generic transformation for sample-based testers of extendable properties follows.

THEOREM 2.8. *Let $q(\cdot,\cdot)$ be a function that is nondecreasing in the first argument and nonincreasing in the second argument. Let $\bigcup_{\mathcal{S}\subseteq\mathcal{D}} \mathcal{P}_{\mathcal{S}}$ be an extendable property. Let $\varepsilon \in (0,1)$. Suppose $T$ is a one-sided error sample-based $\varepsilon$-tester for the property $\bigcup_{\mathcal{S}\subseteq\mathcal{D}} \mathcal{P}_{\mathcal{S}}$, such that for every $\mathcal{S} \subseteq \mathcal{D}$, the tester $T$ makes $q(|\mathcal{S}|,\varepsilon)$ queries from $\mathcal{S}$ to test for $\mathcal{P}_{\mathcal{S}}$. Assume also that for every $\mathcal{S} \subseteq \mathcal{D}$, the probability that $T$ tests $\mathcal{P}_{\mathcal{S}}$ correctly does not decrease when it makes more queries. Then, there is a one-sided error sample-based $\alpha$-erasure-resilient $\varepsilon$-tester for $\mathcal{P}_{\mathcal{D}}$ that works for all $\alpha \in [0,1)$ and makes $O\left(\frac{q(|\mathcal{D}|,\varepsilon)}{1-\alpha}\right)$ queries.*

*Proof.* Let $Q = 2q(|\mathcal{D}|,\varepsilon)/(1-\alpha)$. Consider the tester $T'$ that samples $Q$ points uniformly and independently at random from $\mathcal{D}$. If there are fewer than $q(|\mathcal{D}|,\varepsilon)$ nonerased points in the sample, $T'$ accepts. Otherwise, it runs $T$ on the sampled nonerased points and accepts if and only if $T$ accepts.

The expected number of nonerased points in a uniform sample of size $Q$ from $\mathcal{D}$ is at least $Q \cdot (1-\alpha) = 2q(|\mathcal{D}|,\varepsilon)$. By the Chernoff bound, the probability that $T'$ samples fewer than $q(|\mathcal{D}|,\varepsilon)$ nonerased points is at most $e^{-q(|\mathcal{D}|,\varepsilon)/4}$.

Consider an $\alpha$-erased function $f$ over domain $\mathcal{D}$. Let $\mathcal{N}$ be the set of nonerased points. If $f \in \mathcal{P}$, then $f_{|\mathcal{N}} \in \mathcal{P}_{\mathcal{N}}$ by Lemma 2.7, and the tester $T'$ always accepts. Assume now that $f$ is $\varepsilon$-far from $\mathcal{P}$. Then $f_{|\mathcal{N}}$ is $\varepsilon$-far from $\mathcal{P}_{\mathcal{N}}$ by Lemma 2.7. Therefore $T$ rejects with probability at least $2/3$ on a sample of size at least $q(|\mathcal{N}|,\varepsilon)$. Thus, by a union bound, the probability that $T'$ accepts is at most $1/3 + e^{-q(|\mathcal{D}|,\varepsilon)/4}$. This probability can be brought below $1/3$ by repeating $T'$ a small constant number of times, whenever $q(|\mathcal{D}|,\varepsilon) \geq 8$. ∎

In the following, we show a few applications of Theorem 2.3.

*Convexity of images.* A black and white image, represented by a function $f : S \mapsto \{0,1\}$ for a subset $S$ of $[n]^2$, is convex if and only if for every pair of points $u, v \in S$ such that $f(u) = f(v) = 1$, every point $t \in S$ on the line joining $u$ and $v$

but not undefined, on some points in $\mathcal{T}'$. Also, the values of $f$ and $g$ must be equal on the points in $\mathcal{T}'$ where $f$ is nonerased.

satisfy $f(t) = 1$. Convexity is an extendable property. Testing whether an image, represented by a function $f : [n]^2 \mapsto \{0,1\}$, is convex has been studied by Berman, Murzabulatov and Raskhodnikova [10]. The authors of [10] give a one-sided error sample-based $\varepsilon$-tester for this property that makes $O(1/\varepsilon^{4/3})$ uniform queries. Their proofs go through even if the domain of $f$ is an arbitrary subset of $[n]^2$. The corollary now follows by applying Theorem 2.8 to the tester in [10].

COROLLARY 2.9. *There exists a sample-based $\alpha$-erasure-resilient $\varepsilon$-tester for convexity of black and white images that works for all $\alpha \in [0,1), \varepsilon \in (0,1/2)$, with query complexity $O\left(\frac{1}{(1-\alpha)\varepsilon^{4/3}}\right)$.*

*Monotonicity over poset domains.* A real-valued function $f$ defined on a partially ordered domain is monotone if the function values respect the order relation of the poset. Monotonicity is an extendable property. The tester by Fischer et al. [32] samples $O(\sqrt{N/\varepsilon})$ points uniformly at random and checks for violations to monotonicity among them. The corollary follows by applying Theorem 2.8 to this tester.

COROLLARY 2.10. *There exists a sample-based $\alpha$-erasure-resilient $\varepsilon$-tester for monotonicity of real-valued functions over $N$ element posets that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$, with query complexity $O\left(\frac{1}{(1-\alpha)} \cdot \sqrt{\frac{N}{\varepsilon}}\right)$.*

*Boolean functions with $k$-runs.* A function $f : [n] \mapsto \{0,1\}$ has $k$ *runs* if the list $f(1), f(2), \ldots, f(n)$ has at most $k-1$ alternations of values. The problem is to test whether a given function $f : [n] \mapsto \{0,1\}$ has $k$ runs or is $\varepsilon$-far from this property. Kearns and Ron [41] studied a relaxation of this problem. Specifically, they showed that $O(1/\varepsilon^2)$ queries suffice to test whether a Boolean function has $k$ runs or is $\varepsilon$-far from being a $k/\varepsilon$-run function. They also developed a sample-based $O(\sqrt{k}/\varepsilon^{2.5})$-query tester for this relaxation and proved that every sample-based $\varepsilon$-tester for the $k$-run property requires $\Omega(\sqrt{k})$ queries. Balcan et al. [4] obtained a $O(1/\varepsilon^4)$-query tester for this property in the active testing model. They also developed a sample-based $O(\sqrt{k}/\varepsilon^6)$-query tester[6]. Canonne et al. [16] designed a one-sided error nonadaptive $\varepsilon$-tester for Boolean $k$-run functions that works for all $k \in [n], \varepsilon \in (0,1)$, with query complexity $O(\frac{k+1}{\varepsilon})$. In addition, they also show a two-sided nonadaptive $\varepsilon$-tester for Boolean $k$-run functions that works for all $k \in [n], \varepsilon \in (0,1)$ with query complexity $\tilde{O}(1/\varepsilon^7)$. We show the following.

THEOREM 2.11. *There exists a sample-based $\varepsilon$-tester for the property of being a Boolean function with $k$ runs over $[n]$ that works for all $\varepsilon \in (\frac{k^2}{n}, 1)$, with query complexity $O\left(\min\left\{\frac{k \cdot \log k}{\varepsilon}, \frac{\sqrt{k}}{\varepsilon^6}\right\}\right)$.*

**Algorithm 1** TESTER FOR $k$-RUN BOOLEAN FUNCTIONS

**Input:** parameters $k \in \mathbb{N}, \varepsilon \in (\frac{k^2}{n}, 1)$; oracle access to $f : [n] \mapsto \{0,1\}$

1: Query the values at $\frac{3(k+1)\cdot\log(k+1)}{\varepsilon}$ points uniformly and independently at random.
2: **Reject** if the values of $f$ at these points alternate $k$ or more times with respect to the ordering on the domain; **accept** otherwise.

Our tester for being a $k$-run function is given in Algorithm 1. It always accepts

[6]Both [41] and [4] study Boolean functions over $[0,1]$. We note that their algorithms also work for Boolean functions over $[n]$.

a function $f$ that has at most $k$ runs. The following lemma implies Theorem 2.11.

LEMMA 2.12. *If $f$ is $\varepsilon$-far from being a $k$-run function, Algorithm 1 rejects with probability at least $2/3$.*

*Proof.* For $j \in [n]$ and $b \in \{0,1\}$, let $T_{b,j}$ denote the set consisting of the smallest $\lceil n \cdot \varepsilon/(k+1) \rceil$ points in the set $\{x : j \le x \le n \text{ and } f(x) = b\}$, that is, the set of points between $j$ and $n$ where $f$ takes the value $b$. For a set $S \subseteq [n]$, let $\max(S)$ denote the largest element in $S$. We will first describe a process to construct a few disjoint subsets of $[n]$ with some special properties.

- Let $S_1 = T_{b,1}$ such that $\max(T_{b,1}) < \max(T_{1-b,1})$.
- For $i \ge 2$, the sets $S_i$ are defined as follows. Let the value that $f$ takes on the elements in $S_{i-1}$ be $b$ and let $j = \max(S_{i-1})$. Set $S_i = T_{1-b,j+1}$. Stop if $\max(S_i) = n$ or $S_i = \emptyset$.

The sets that this process constructs have the following properties. All $S_i$'s are subsets of $[n]$. Each point in $S_{i+1}$ is larger than every point in $S_i$ for all $i$. The function $f$ takes the same value on all points in $S_i$ for all $i$. The value of $f$ on points in $S_{i+1}$ is the complement of the value of $f$ on points in $S_i$ for all $i$.

Next, we show that our process constructs sets $S_1, S_2, \dots S_{k+1}$ each of size $\lceil n \cdot \varepsilon/(k+1) \rceil$, if $f$ is $\varepsilon$-far from satisfying the property. Let the process construct nonempty sets $S_1, S_2, \dots S_t$. Assume for the sake of contradiction that $t \le k$. Let $S_1' = \{x : 1 \le x \le \max(S_1)\}$. Let $S_i' = \{x : \max(S_{i-1}) < x \le \max(S_i)\}$ for all $1 < i \le t$. Note that for all $i \in [t]$, if $f$ takes the value $b$ on elements in $S_i$, then $f$ takes the value $1-b$ on elements in $S_i' \setminus S_i$. We will describe a function $f'$ that has at most $t$ runs. Set the values of $f'$ on each $x \in S_1' \setminus S_1$ to the value that $f$ takes on $S_1$. For each $1 < i \le t$, set the values of $f'$ on $S_i$ to the value of $f$ on $S_i' \setminus S_i$. On the rest of the points, $f'$ takes the same value as $f$. We will now show that $f'$ has at most $t$ alternating intervals. The function $f'$ takes the same value on points in $S_1' \cup S_2'$. Also, for each $1 < i \le t$, the function $f'$ is constant on $S_i'$. Thus, $f'$ has at most $t$ runs. Also, $f'$ differs from $f$ in at most $t \cdot \lceil n \cdot \varepsilon/(k+1) \rceil \le k \cdot \lceil n \cdot \varepsilon/(k+1) \rceil \le n\varepsilon$ points, for $k < \sqrt{n\varepsilon}$. This is a contradiction.

Using the fact that $k+1$ such subsets exist, we show that the tester will detect a violation with high probability. For a particular $i$, the probability that none of the points selected by the algorithm lie in $S_i$ is at most

$$(1 - \varepsilon/(k+1))^{3(k+1)\log(k+1)/\varepsilon} \le 1/(k+1)^3.$$

Therefore, by a union bound, the probability that there exists an $i$ such that none of the points selected by the algorithm lies in $S_i$ is at most $(k+1)^{-2} < 1/3$ for $k \ge 1$. ∎

Since the property of being a $k$-run function is extendable, applying Theorem 2.8 to Theorem 2.11 yields the following corollary.

COROLLARY 2.13. *There exists a sample-based $\alpha$-erasure-resilient $\varepsilon$-tester for the property of being a $k$-run Boolean function over $[n]$, that works for all $\alpha \in [0,1)$, $\varepsilon \in \left(\frac{k^2}{n}, 1\right)$, with query complexity $O\left(\frac{1}{1-\alpha} \cdot \min\left\{\frac{k \cdot \log k}{\varepsilon}, \frac{\sqrt{k}}{\varepsilon^6}\right\}\right)$.*

**3. Erasure-Resilient Monotonicity Tester for the Line.** In this section, we prove Theorem 1.6. Recall that, for a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, the set of nonerased points (the ones that map to $\mathbb{R}$) is denoted by $\mathcal{N}$. The function $f$ is monotone if $x < y$ implies $f(x) \le f(y)$ for all $x, y \in \mathcal{N}$. Given a function $f : [n] \to \mathbb{R} \cup \{\perp\}$ that is not monotone, a *violation to monotonicity* of $f$ is a pair of points $x, y \in \mathcal{N}$ such that $x < y$ and $f(x) > f(y)$. The points $x$ and $y$ are said to *violate the monotonicity* of $f$.

We present our tester in Algorithm 2. It has oracle access to $f : [n] \to \mathbb{R} \cup \{\perp\}$ and takes $\alpha$ and $\varepsilon$ as inputs. The tester knows neither the set $\mathcal{N}$ nor the value of $|\mathcal{N}|$ in advance. However, it gets a lower bound on $|\mathcal{N}|$ in the form of $n(1-\alpha)$. In each iteration, it performs a randomized binary search for a nonerased index sampled uniformly at random (u.a.r.) from $\mathcal{N}$ and rejects if it finds violations to monotonicity. In the description of our tester, we use $I[i,j]$ to denote the set of natural numbers from $i$ until and including $j$. We alternatively refer to $I[i,j]$ as the interval from $i$ to $j$.

**Algorithm 2** Erasure-Resilient Monotonicity Tester for the Line

**Input:** parameters $\alpha \in [0,1), \varepsilon \in (0,1)$; oracle access to $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$

1: **Set** $Q = \left\lceil \frac{60 \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2: **Accept** at any point if the number of queries exceeds $Q$.
3: **repeat** $\frac{2}{\varepsilon}$ times:
4: Sample points uniformly at random from $I[1,n]$ and query them until we get a point $s \in \mathcal{N}$.
5: **Set** $\ell \leftarrow 1$, $r \leftarrow n$.
6: **while** $\ell \leq r$ **do**
7: Sample points uniformly at random from $I[\ell, r]$ and query them until we get a point $m \in \mathcal{N}$.
8: **if** $s < m$ **then set** $r \leftarrow m-1$ and **Reject** if $f(s) > f(m)$.
9: **if** $s > m$ **then set** $\ell \leftarrow m+1$ and **Reject** if $f(s) < f(m)$.
10: **if** $s = m$ **then** Go to Step 3. ▷ Search completed.
11: **Accept**.

One of the key ideas in our analysis is to view each iteration of the loop in Step 3 as first sampling a binary search tree $\mathcal{T}$ on $\mathcal{N}$ according to a particular distribution $\mathcal{D}_{\mathcal{T}}$, and then traversing a uniformly random rooted path in that tree, where a rooted path refers to a path from the root of a tree to an arbitrary node in that tree. This view enables us to prove the correctness of the tester by generalizing an argument from [26] for the case when Algorithm 2 manages to complete all iterations of Step 3 before it runs out of queries. The challenge is that the algorithm might get stuck pursuing long paths in the search tree and waste many queries on erased points. To resolve the issue of many possible queries to erased points, we prove an upper bound on the expected number of queries made while traversing a uniformly random rooted path in a binary search tree sampled from $\mathcal{D}_{\mathcal{T}}$. We combine this with the fact that the expected depth of a binary search tree sampled from $\mathcal{D}_{\mathcal{T}}$ is $O(\log n)$, in order to obtain the final bound on the probability that the algorithm exceeds its query budget (and wrongly accepts functions that are far from monotone).

**3.1. Analysis.** We analyze the tester in this section. The query complexity of the tester is clear from its description. The main statement of Theorem 1.6 follows from Lemma 3.1, proved next.

LEMMA 3.1. *Algorithm 2 accepts if $f$ is monotone, and rejects with probability at least $2/3$ if $f$ is $\varepsilon$-far from monotone.*

*Proof.* The tester accepts whenever $f$ is monotone. To prove the other part of the lemma, assume that $f$ is $\varepsilon$-far from monotone. Let $A$ be the event that the tester accepts $f$. Let $q$ denote the total number of queries made. We prove that $\Pr[A] \leq 1/3$.

The event $A$ occurs if either $q > Q$ or the tester does not find a violation in any of the $2/\varepsilon$ iterations of Step 3. Thus, $\Pr[A] \leq \Pr[A \mid q \leq Q] + \Pr[q > Q]$.

Each iteration of Step 3 can be viewed as traversing a uniformly random search path in a binary search tree on $\mathcal{N}$, where the tree is sampled from a particular distribution. More formally, we describe a two-stage random process that provides such an alternate view of a single iteration of Step 3.

The first stage of the process involves constructing a binary search tree $\mathcal{T}$ over $\mathcal{N}$ as follows. Each node of $\mathcal{T}$ is associated with an interval $I[a, b]$ and has a nonerased point from $I[a, b]$ set as its key, where $1 \leq a \leq b \leq n$. The root node of $\mathcal{T}$ is associated with the interval $I[1, n]$. Consider an arbitrary node $\Gamma$ of the tree whose key has not been set yet. Let $I[a, b]$ be the interval associated with it. Repeatedly sample points u.a.r. from $I[a, b]$ and query them, until we get a point $p \in \mathcal{N} \cap I[a, b]$. Set the key of the node $\Gamma$ to $p$. If $p$ is the *leftmost* nonerased point in $I[a, b]$, then the node $\Gamma$ does not have a left child and its right child is associated with $I[p + 1, b]$. Similarly, if $p$ is the *rightmost* nonerased point in $I[a, b]$, then $\Gamma$ does not have a right child and its left child is associated with $I[a, p - 1]$. Otherwise, we associate the intervals $I[a, p - 1]$ and $I[p+1, b]$ with the left and right children of $\Gamma$, respectively. We use $\mathcal{D}_{\mathcal{T}}$ to denote the distribution on binary search trees sampled in this way. Note that the leaves of $\mathcal{T}$ are nodes associated with intervals containing exactly one nonerased point. Since the key of each node in $\mathcal{T}$ is a unique nonerased point in $\mathcal{N}$, we will henceforth refer to the nodes of $\mathcal{T}$ using their keys.

In the second stage of the random process, we sample a node $s \in \mathcal{N}$ of $\mathcal{T}$ u.a.r. and reject if $s$ with any one of its ancestors in $\mathcal{T}$ violate the monotonicity of $f$. In other words, we sample a uniformly random rooted path from $\mathcal{T}$ and check whether the deepest node on that path violates the monotonicity of $f$ with any of its ancestors.

We now argue that each iteration of Step 3 of our algorithm simulates the above random process. Consider the search path traversed by the algorithm in an iteration. It is easy to see that there exists a binary search tree $\mathcal{T}$ sampled from $\mathcal{D}_{\mathcal{T}}$ such that $\mathcal{T}$ contains the search path traversed by the algorithm. Each node of this binary search tree $\mathcal{T}$ is a unique element of $\mathcal{N}$. As the algorithm samples its search point $s$ u.a.r. from $\mathcal{N}$, the node corresponding to $s$ in $\mathcal{T}$ is a node sampled u.a.r. from the tree $\mathcal{T}$. The algorithm checks whether the monotonicity of $f$ is violated by $s$ and any of the nonerased points on its search path. This is exactly the same as checking for violations to monotonicity of $f$ between $s$ and its ancestors in $\mathcal{T}$. The number of queries made to the intervals associated with the nodes along the path from the root of $\mathcal{T}$ to $s$ during the random process has the same distribution as the number of queries made by the tester while traversing the search path to $s$.

We are now ready to bound the probability that the tester does not reject in an iteration of Step 3, conditioned on the event that $q \leq Q$. Consider a binary search tree $\mathcal{T}$ over $\mathcal{N}$ sampled from the distribution $\mathcal{D}_{\mathcal{T}}$. A point $s \in \mathcal{N}$ is called *searchable* with respect to $\mathcal{T}$ if $s$ does not violate the monotonicity of $f$ with any of its ancestors in $\mathcal{T}$. Consider two points $i, j \in \mathcal{N}$, where $i < j$, both searchable with respect to $\mathcal{T}$. Let $a \in \mathcal{N}$ be the lowest common ancestor of the nodes $i$ and $j$ in $\mathcal{T}$. Since $i$ and $j$ are both *searchable*, it must be the case that $f(i) \leq f(a)$ and $f(a) \leq f(j)$ and hence, $f(i) \leq f(j)$. Thus, for every tree $\mathcal{T}$ sampled from $\mathcal{D}_{\mathcal{T}}$, the function $f$ restricted to the domain points that are *searchable* with respect to $\mathcal{T}$ is monotone. Therefore, if $f$ is $\varepsilon$-far from monotone, for every binary search tree $\mathcal{T}$ sampled from $\mathcal{D}_{\mathcal{T}}$, at least an $\varepsilon$-fraction of the points in $\mathcal{N}$ are not *searchable* with respect to $\mathcal{T}$. Thus, the random process, and each iteration of Step 3 of the tester, reject with probability at least $\varepsilon$.

Consequently,

$$\Pr[A \mid q \le Q] \le (1-\varepsilon)^{\frac{2}{\varepsilon}} \le e^{-2} < \frac{1}{6}.$$

In the rest of the proof, we bound $\Pr[q > Q]$. We first prove an upper bound on the expected number of queries to traverse a uniformly random rooted path in a binary search tree $\mathcal{T}$ sampled according to $\mathcal{D}_{\mathcal{T}}$. Recall that a rooted path in a search tree $\mathcal{T}$ is a path from the root to some node in $\mathcal{T}$. Let $I$ be the interval associated with a node $\Gamma$ of $\mathcal{T}$ and let $\alpha_I$ denote the fraction of erased points in $I$. The number of queries needed to sample a nonerased point from $I$ with uniform sampling is a geometric random variable with expectation $1/(1-\alpha_I)$. We define the *query-weight* of node $\Gamma$ to be this expectation. The query-weight of a path in $\mathcal{T}$ is the sum of query-weights of the nodes on the path (which is equal to the expected total number of queries made by the random process to all the intervals on that path while constructing $\mathcal{T}$).

Claim 3.2. *Let $f$ be an $\alpha$-erased function, where $\alpha \in [0,1)$. Consider a binary search tree $\mathcal{T}$ of height $h$ over $\mathcal{N}$, sampled according to the distribution $\mathcal{D}_{\mathcal{T}}$. The expected query-weight of a uniformly random rooted path in $\mathcal{T}$ is at most $\frac{h}{1-\alpha}$.*

*Proof.* There are exactly $|\mathcal{N}|$ rooted paths in $\mathcal{T}$. Let $S$ denote the sum of query-weights of all the rooted paths. The expected query-weight of a uniformly random rooted path in $\mathcal{T}$ is then equal to $S/|\mathcal{N}|$.

Consider a node $\Gamma$ in $\mathcal{T}$ associated with an interval $I$. There are $|I|(1-\alpha_I)$ nonerased points in $I$. The paths from the root of $\mathcal{T}$ to the nodes corresponding to each of these nonerased points pass through $\Gamma$. Hence, the query-weight of $\Gamma$ gets added to the query-weights of all those paths. Therefore, the total contribution of $\Gamma$ towards $S$ is $|I|$, since the query-weight of $\Gamma$ is $1/(1-\alpha_I)$. Note that the intervals associated with nodes at the same level of $\mathcal{T}$ are disjoint from each other. Hence, the total contribution to $S$ from all nodes on the same level of $\mathcal{T}$ is at most $n$. Therefore, the value of $S$ is at most $n \cdot h$, where $h$ is the depth of $\mathcal{T}$. Observe that this quantity is independent of the fraction of erasures $\alpha$. Therefore, the expected query-weight of a search path is at most $n \cdot h/|\mathcal{N}|$, which is at most $h/(1-\alpha)$, since $|\mathcal{N}| \ge n \cdot (1-\alpha)$. □

Next, we bound the expected height of a tree $\mathcal{T}$ sampled from $\mathcal{D}_{\mathcal{T}}$. Consider the following random process that constructs a binary tree $T$ on the set $S = [k]$. The root of $T$ is associated with the set $S$. The key of an arbitrary node $v$ in $T$ associated with a set $S' \subseteq S$ is a uniformly random element $x \in S'$. The left child of $v$ is associated with the set $\{y \in S' : y < x\}$ if this set is nonempty. The right child of $v$ is associated with the set $\{y \in S' : y > x\}$ if this set is nonempty. A tree constructed in this way is called a random binary search tree on $k$ nodes. We now state a fact on the expected height of a random binary search tree.

Claim 3.3 ([45, 20, 25, 46]). *If $H_k$ is the random variable denoting the height of a random binary search tree on $k$ nodes, then $\mathbf{E}[H_k] \le 5 \log k$.*

It is easy to see that the height of a tree $\mathcal{T}$ sampled from $\mathcal{D}_{\mathcal{T}}$ has the same distribution as the random variable $H_{|\mathcal{N}|}$, since the key associated with each node in $\mathcal{T}$ is a uniformly random nonerased point from the interval associated with that node. Hence, the expected depth of $\mathcal{T}$ is at most $5 \log(|\mathcal{N}|) \le 5 \log n$. The corollary below follows immediately.

Corollary 3.4. *The expected total number of queries made to all the intervals of a uniformly random rooted path in a random binary search tree $\mathcal{T}$ on $\mathcal{N}$, sampled according to $\mathcal{D}_{\mathcal{T}}$, is at most $\frac{5 \log n}{1-\alpha}$.*

It follows from Corollary 3.4 that the expected number of queries made by Algorithm 2 in a single iteration is at most $5 \log n/(1-\alpha)$. Hence, by the linearity of expectation, the expected number of queries made by the tester over all its iterations, $\mathbf{E}[q]$, is at most $10 \log n/(\varepsilon \cdot (1-\alpha))$. Applying Markov's inequality to $q$, we can then see that

$$\Pr[q > Q] \leq \frac{1}{6}.$$

Therefore, the probability that the tester does not reject is

$$\Pr[A] \leq \Pr[A \mid a \leq Q] + \Pr[q > Q] < \frac{1}{6} + \frac{1}{6} = \frac{1}{3}.$$

This completes the proof of the lemma. ∎

**4. Erasure-Resilient Monotonicity Testers for the Hypergrid.** In this section, we present our erasure-resilient tester for monotonicity over hypergrid domains and prove the following theorem, which is a special case of Theorem 1.7. We present the erasure-resilient testers for general BDPs in Section 5.

THEOREM 4.1. *There exists a one-sided error $\alpha$-erasure-resilient $\varepsilon$-tester for monotonicity of real-valued functions on the hypergrid $[n]^d$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/250d$, with query complexity $O(\frac{d \log n}{\varepsilon(1-\alpha)})$.*

Let $\mathcal{L}$ denote the set of all *axis-parallel lines* in the hypergrid, where an axis-parallel line is a set of $n$ distinct points in $[n]^d$ that agree on all but one coordinate. Our monotonicity tester, which is described in Algorithm 3, samples a uniformly random *axis-parallel line* in each iteration of Step 2 and does a randomized binary search for a uniformly random nonerased point on that line (as in Algorithm 2). It rejects if and only if a violation to monotonicity is found within its query budget. To analyze the tester, we first state two important properties of a uniformly random axis-parallel line in Lemma 4.2 and Lemma 4.3, which we jointly call the erasure-resilient dimension reduction. The statements and proofs of more general versions of these lemmas, applicable to all BDPs, are given in Section 5.

LEMMA 4.2 (Dimension reduction: distance). *Let $\varepsilon_f$ be the relative Hamming distance of an $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$ from monotonicity. For an axis-parallel line $\ell \in \mathcal{L}$, let $f_\ell : [n] \mapsto \mathbb{R} \cup \{\perp\}$ denote the restriction of $f$ to $\ell$ and let $\varepsilon_\ell$ denote the relative Hamming distance of $f_\ell$ from monotonicity. Then*

$$\mathbf{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell] \geq \frac{(1-\alpha) \cdot \varepsilon_f}{4d} - \alpha.$$

LEMMA 4.3 (Dimension reduction: fraction of erasures). *Consider an $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$. For $\ell \in \mathcal{L}$, let $\alpha_\ell$ denote the fraction of erased points in $\ell$. Then, for every $\eta \in (0,1)$,*

$$\Pr_{\ell \sim \mathcal{L}}\left[\alpha_\ell > \frac{\alpha}{\eta}\right] \leq \eta.$$

The query complexity of the tester is evident from its description. We will now prove its correctness in the following lemma, which will then imply Theorem 4.1.

LEMMA 4.4. *Algorithm 3 accepts if $f$ is monotone, and rejects with probability at least $2/3$ if $f$ is $\varepsilon$-far from monotone.*

**Algorithm 3** Erasure-Resilient Monotonicity Tester for $[n]^d$

**Input:** parameters $\varepsilon \in (0,1), \alpha \in \left[0, \frac{\varepsilon}{250d}\right]$; oracle access to $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$

1: **Set** $Q = \left\lceil \frac{1200d \cdot \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2: **repeat** $\frac{12d}{\varepsilon(1-\alpha)-4d\alpha}$ times:
3: Sample a line $\ell \in \mathcal{L}$ uniformly at random.
4: Sample points u.a.r. from $\ell$ and query them until we get a point $s \in \mathcal{N}$.
5: Perform a randomized binary search for $s$ on $\ell$ as in Algorithm 2.
6: **Reject** if any violation to monotonicity is found.
7: **Accept** at any point if the number of queries exceed $Q$.

*Proof.* The tester accepts if $f$ is monotone. So, assume that $f$ is $\varepsilon$-far from being monotone. Let $A$ denote the event that the tester does not find a violation to monotonicity in any of its iterations. If $q$ denotes the total number of queries made by the tester,

$$\Pr[A] \leq \Pr[A | q \leq Q] + \Pr[q > Q].$$

Let $t$ denote the number of iterations of Step 2 of the tester. Let $A_i$ denote the event that the tester does not find a violation to the monotonicity of $f$ in its $i$-th iteration. For $\ell \in \mathcal{L}$, let $f_\ell$ denote $f$ restricted to the line $\ell$. Let $\varepsilon_\ell$ denote the relative Hamming distance of $f_\ell$ from monotonicity. Let $E_\ell$ denote the event that the tester samples the line $\ell$ in a particular iteration.

We then have, $\Pr[A_i \mid q \leq Q] \leq \sum_{\ell \in \mathcal{L}} (1 - \varepsilon_\ell) \Pr[E_\ell] = 1 - \mathbf{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell]$. Using Lemma 4.2 and the fact that $\varepsilon_f \geq \varepsilon$, we have,

$$\mathbf{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell] \geq \frac{(1-\alpha) \cdot \varepsilon_f}{4d} - \alpha \geq \frac{(1-\alpha) \cdot \varepsilon}{4d} - \alpha.$$

Therefore,

$$\Pr[A \mid q \leq Q] = \prod_{i=1}^{t} \Pr[A_i \mid q \leq Q] \leq \left(1 - \frac{(1-\alpha) \cdot \varepsilon - 4d\alpha}{4d}\right)^t < \frac{1}{10}.$$

It now remains to bound $\Pr[q > Q]$. Let $\eta$ stand for $1/10t$. Let $\alpha_i$ denote the fraction of erasures in the line sampled during iteration $i$ and let $q_i$ denote the number of queries made by the algorithm during iteration $i$. Let $G$ denote the (good) event that $\alpha_i \leq \alpha/\eta$ for all iterations $i \in [t]$. By Corollary 3.4, $\mathbf{E}[q_i \mid G] \leq 5\eta \cdot \log n/(\eta - \alpha)$, and by the linearity of expectation,

$$\mathbf{E}[q \mid G] \leq \frac{\log n}{2(\eta - \alpha)} \leq \frac{120d \log n}{\varepsilon(1-\alpha)},$$

where the last inequality above follows from our assumption that $\alpha \leq \varepsilon/250d$. Using Markov's inequality, $\Pr[q > Q \mid G] \leq 1/10$. Also, by combining Lemma 4.3 with a union bound over the iterations of Step 2 of the tester, we can see that $\Pr[\overline{G}] \leq 1/10$. Therefore, $\Pr[q > Q] \leq \Pr[q > Q \mid G] + \Pr[\overline{G}] \leq 1/5$. Thus, the probability that the tester does not reject $f$ is

$$\Pr[A] \leq \Pr[A \mid q \leq Q] + \Pr[q > Q] < \frac{1}{10} + \frac{1}{5} < \frac{1}{3}.$$

**5. Erasure-Resilient BDP Testing.** In this section, we discuss our erasure-resilient testers for all bounded derivative properties over hypergrid domains and prove Theorem 1.7. First, we show in Lemma 5.4 that testing for any BDP on $[n]$ reduces to testing monotonicity on $[n]$. Next, we prove Lemma 5.7 and Lemma 5.8 that reduces the problem of erasure-resilient testing of a BDP over hypergrid domains to testing of the same property over the line.

**5.1. Erasure-Resilient BDP Tester for the Line.** In this section, we show that (erasure-resilient) testing of bounded derivative properties (BDPs) on the line reduces to monotonicity testing on the line and prove Theorem 5.5. As noted in Section 1.3, BDPs comprise of a large class of properties that have been studied in the property testing literature.

Given a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, and a bounded derivative property $\mathcal{P}$, we first define violated pairs in $f$ with respect to $\mathcal{P}$.

DEFINITION 5.1 (Violated pair). *Given a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$ and bounding family $\mathbf{B}$ consisting of functions $l, u : [n-1] \mapsto \mathbb{R}$, two points $x, y \in \mathcal{N}$ such that $x < y$* violate *the property $\mathcal{P}(\mathbf{B})$ with respect to $f$ if $f(x) - f(y) > \mathfrak{m}_{\mathbf{B}}(x, y) = -\sum_{t=x}^{y-1} l(t)$ or $f(y) - f(x) > \mathfrak{m}_{\mathbf{B}}(y, x) = \sum_{t=x}^{y-1} u(t)$. The pairs $(x, y)$ and $(y, x)$ are called* violated.

Consider a bounded derivative property $\mathcal{P}$ of functions defined over $[n]$ and associated bounding functions $l, u : [n-1] \mapsto \mathbb{R}$. The following claim states that, we may assume w.l.o.g. that $l(i) = -u(i)$ for all $i \in [n-1]$. We use it in the proof of Claim 5.3.

CLAIM 5.2. *Consider a function $f : [n] \to \mathbb{R} \cup \{\perp\}$ and a bounding function family $\mathbf{B}$ over $[n]$ with $l, u : [n-1] \mapsto \mathbb{R}$. Let $g : [n] \mapsto \mathbb{R} \cup \{\perp\}$ be a function that takes the value $f(i) + \sum_{j=i}^{n-1} \frac{l(j)+u(j)}{2}$ for each $i \in \mathcal{N}$ and is erased on the remaining points. Let $\mathbf{B}'$ be a bounding function family over $[n]$ with $l', u' : [n-1] \mapsto \mathbb{R}$ such that $u'(i) = -l'(i) = \frac{u(i)-l(i)}{2}$ for all $i \in [n-1]$. Then $x, y \in \mathcal{N}$ violate $\mathcal{P}(\mathbf{B})$ with respect to $f$ if and only if $x, y$ violate $\mathcal{P}(\mathbf{B}')$ with respect to $g$.*

*Proof.* Note that $x, y \in \mathcal{N}$, where $x < y$, is not violated with respect to $f$ if and only if $\max\{f(x) - f(y) - \mathfrak{m}_{\mathbf{B}}(x, y), f(y) - f(x) - \mathfrak{m}_{\mathbf{B}}(y, x)\} \leq 0$. We have

$$\begin{aligned} g(x) - g(y) - \mathfrak{m}_{\mathbf{B}'}(x, y) &= f(x) - f(y) + \sum_{i=x}^{y-1} \frac{u(i) + l(i)}{2} - \sum_{i=x}^{y-1} \frac{u(i) - l(i)}{2} \\ &= f(x) - f(y) - \sum_{i=x}^{y-1} l(i) = f(x) - f(y) - \mathfrak{m}_{\mathbf{B}}(x, y). \end{aligned}$$

Also,

$$\begin{aligned} g(y) - g(x) - \mathfrak{m}_{\mathbf{B}'}(y, x) &= f(y) - f(x) - \sum_{i=x}^{y-1} \frac{u(i) + l(i)}{2} - \sum_{i=x}^{y-1} \frac{u(i) - l(i)}{2} \\ &= f(y) - f(x) - \sum_{i=x}^{y-1} u(i) = f(y) - f(x) - \mathfrak{m}_{\mathbf{B}}(y, x). \end{aligned}$$

Thus, $\max\{g(x) - g(y) - \mathfrak{m}_{\mathbf{B}'}(x, y), g(y) - g(x) - \mathfrak{m}_{\mathbf{B}'}(y, x)\} = \max\{f(x) - f(y) - \mathfrak{m}_{\mathbf{B}}(x, y), f(y) - f(x) - \mathfrak{m}_{\mathbf{B}}(y, x)\}$. The claim follows. ∎

The following claim shows a reduction from testing BDPs over $[n]$ to testing monotonicity over $[n]$.

CLAIM 5.3. *Consider an $\alpha$-erased function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$ and bounding functions $l, u : [n-1] \mapsto \mathbb{R}$ such that $-l(i) = u(i) = \gamma(i)$ for all $i \in [n-1]$. Let $\mathcal{P}$ be the BDP defined by $l$ and $u$. Let $g, h : [n] \mapsto \mathbb{R} \cup \{\perp\}$ be two functions that take the values $g(i) = f(i) - \sum_{r=i}^{n-1} \gamma(r)$ and $h(i) = -f(i) - \sum_{r=i}^{n-1} \gamma(r)$ for all $i \in \mathcal{N}$ and are erased on the remaining points. Then, the following conditions hold:*

*(1) $x, y \in \mathcal{N}$ violate $\mathcal{P}$ with respect to $f$ iff $x, y$ violate monotonicity with respect to either $g$ or $h$.*

*(2) If $f$ is in $\mathcal{P}$, then both $g$ and $h$ are both monotone.*

*(3) If $f$ is $\varepsilon$-far from $\mathcal{P}$, then either $g$ or $h$ is at least $\varepsilon/4$-far from monotonicity.*

*Proof.* Consider a pair $(i, j) \in \mathcal{N} \times \mathcal{N}$ where $i < j$. We have,

$$g(i) - g(j) = f(i) - f(j) - \sum_{r=i}^{j-1} \gamma(r);$$

$$h(i) - h(j) = f(j) - f(i) - \sum_{r=i}^{j-1} \gamma(r).$$

If $(i, j)$ does not violate $\mathcal{P}$ with respect to $f$, we have $f(j) - f(i) - \sum_{r=i}^{j-1} \gamma(r) \leq 0$ and $f(i) - f(j) - \sum_{r=i}^{j-1} \gamma(r) \leq 0$. Thus, $(i, j)$ satisfies the monotonicity property with respect to $g$ and $h$. On the other hand, if $(i, j)$ violates $\mathcal{P}$ with respect to $f$, then either $f(j) - f(i) - \sum_{r=i}^{j-1} \gamma(r) > 0$ or $f(i) - f(j) - \sum_{r=i}^{j-1} \gamma(r) > 0$. That is, $(i, j)$ violates monotonicity with respect to either $g$ or $h$. Parts (1) and (2) of the lemma follow directly from these arguments.

To prove part (3) of the lemma, assume that $f$ is $\varepsilon$-far from the property $\mathcal{P}$. Define the violation graph $G_f$ as follows. The vertex set corresponds to $\mathcal{N}$. For each $(i, j) \in \mathcal{N} \times \mathcal{N}$ such that $i < j$, there is an (undirected) edge between $i \in \mathcal{N}$ and $j \in \mathcal{N}$ iff the pair $(i, j)$ violates $\mathcal{P}$ with respect to $f$. By Lemma 2.5 in [17], the size of every maximal matching in $G_f$ is at least $\varepsilon \cdot |\mathcal{N}|/2$. Consider a maximal matching $M$ in $G_f$. From the discussion above, the pair of nonerased points corresponding to each edge in $M$ violates monotonicity with respect to either $g$ or $h$. Therefore, at least $\varepsilon \cdot |\mathcal{N}|/4$ pairs $(i, j) \in \mathcal{N} \times \mathcal{N}$ such that $i < j$, violate monotonicity with respect to at least one of $g$ and $h$. Assume w.l.o.g. that at least $\varepsilon \cdot |\mathcal{N}|/4$ such pairs violate monotonicity with respect to $h$. One has to change the function value of $h$ on at least one endpoint of each such pair to repair it. This means that $h$ is at least $\varepsilon/4$-far from monotone. ∎

Therefore, in order to test the bounded derivative property $\mathcal{P}$ on $f$ with proximity parameter $\varepsilon$, one can test monotonicity on $g$ and $h$ with proximity parameter $\varepsilon/4$ and error probability $1/6$ and accept iff both tests accept.

LEMMA 5.4. *Let $\varepsilon \in (0, 1), \alpha \in [0, 1)$. Let $Q_{\texttt{mon}}(\alpha, \varepsilon, n)$ denote the query complexity of $\alpha$-erasure-resilient $\varepsilon$-testing of monotonicity of real-valued functions on the line. Then, for every BDP, $\alpha$-erasure-resilient $\varepsilon$-testing of real-valued functions on the line has query complexity $O(Q_{\texttt{mon}}(\alpha, \varepsilon/4, n))$. The same statement holds for 1-sided error testing.*

The following theorem is a direct consequence of Lemma 5.4 and Theorem 1.6.

THEOREM 5.5 (BDP tester on the line). *For every BDP $\mathcal{P}$, there exists a one-sided error $\alpha$-erasure-resilient $\varepsilon$-tester for $\mathcal{P}$ of real-valued functions over $[n]$ that works for all $\alpha \in [0, 1), \varepsilon \in (0, 1)$, with query complexity $O\left(\frac{1}{1-\alpha} \cdot \frac{\log n}{\varepsilon}\right)$.*

**5.2. Erasure-Resilient Dimension Reduction.** In this section, we prove two important properties of a uniformly random axis parallel line in the hypergrid $[n]^d$. We do this in Lemma 5.7 and Lemma 5.8, which we jointly call erasure-resilient dimension reduction. We first introduce some notation.

Let $g$ be an $\alpha$-erased function on $\mathcal{D}$, and $\mathcal{N} \subseteq \mathcal{D}$ be the set of nonerased points in $g$. Recall that the Hamming distance of $g$ from $\mathcal{P}$, denoted by $\text{dist}(g, \mathcal{P})$, is the least number of nonerased points on which every completion of $g$ needs to be changed to satisfy $\mathcal{P}$. The relative Hamming distance between $g$ and $\mathcal{P}$ is $\text{dist}(g, \mathcal{P})/|\mathcal{N}|$. We use $g_{|\mathcal{S}}$ to denote the restriction of $g$ to a subset $\mathcal{S} \subseteq \mathcal{D}$. Note that all these definitions make sense even for functions with no erasures in them.

Let $\mathcal{L}$ denote the set of all *axis-parallel lines* in $[n]^d$. Let $\mathcal{P}$ be a bounded derivative property of functions over $[n]^d$ defined by a bounding family $\mathbf{B} = \{l_1, u_1, \ldots, l_d, u_d\}$ where $l_i, u_i : [n-1] \mapsto \mathbb{R}$ for all $i \in [d]$. For $i \in [d]$, let $\mathcal{P}^i$ denote the set of functions over $[n]^d$ with no violations to $\mathcal{P}$ along dimension $i$. Let $\mathcal{P}^i_{\mathsf{line}}$ denote the bounded derivative property of functions over $[n]$ defined by the bounding functions $l_i, u_i : [n-1] \mapsto \mathbb{R}$.

Consider an $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$. Let $\mathcal{N} \subseteq [n]^d$ denote the set of nonerased points in $f$. For an axis-parallel line $\ell \in \mathcal{L}$, let $\mathcal{N}_\ell$ denote the set of nonerased points on $\ell$ and $f_\ell$ denote the function $f$ restricted to $\ell$.

Lemma 5.7 shows that, for a uniformly random axis-parallel line $\ell \in \mathcal{L}$, the expected relative Hamming distance of $f_\ell$ from $\mathcal{P}^i_{\mathsf{line}}$ is roughly proportional to the relative Hamming distance of $f$ from $\mathcal{P}$, where $i$ is the dimension along which $\ell$ lies. First, we prove Claim 5.6 that we use in our proof of Lemma 5.7.

CLAIM 5.6. *Let $\alpha \in [0, 1)$. For every $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$ and every bounded derivative property $\mathcal{P}$ over $[n]^d$, we have,*

$$\frac{1}{4}\text{dist}(f, \mathcal{P}) \leq \alpha \cdot d \cdot n^d + \sum_{i=1}^{d} \text{dist}(f, \mathcal{P}^i).$$

*Proof.* Let $g : [n]^d \mapsto \mathbb{R}$ be a function in $\mathcal{P}$ such that $\text{dist}(g_{|\mathcal{N}}, f_{|\mathcal{N}})$ is minimum. We define $f_* : [n]^d \mapsto \mathbb{R}$, a completion of $f$, such that $f_*(x) = f(x)$ for all $x \in \mathcal{N}$ and $f_*(x) = g(x)$ for all $x \notin \mathcal{N}$. Note that $g$ is the function closest to $f_*$ in $\mathcal{P}$, as $g$ is the function that minimizes $\text{dist}(g_{|\mathcal{N}}, f_{|\mathcal{N}})$.

For all $i \in [d]$, let $g^i : [n]^d \mapsto \mathbb{R}$ in $\mathcal{P}^i$ be such that $\text{dist}(g^i_{|\mathcal{N}}, f_{|\mathcal{N}})$ is minimum. Also, for all $i \in [d]$, let $h^i : [n]^d \mapsto \mathbb{R}$ be defined as $h^i(x) = f(x)$ for all $x \in \mathcal{N}$, and $h^i(x) = g^i(x)$ for $x \in [n]^d \setminus \mathcal{N}$. Note that for all $i \in [d]$, the function $h^i$ is a completion

of $f$. We have,

$$\begin{aligned}
\frac{1}{4}\mathrm{dist}(f,\mathcal{P}) &\le \frac{1}{4}\mathrm{dist}(f_*,\mathcal{P}) && \text{as } f_* \text{ is a completion of } f\\
&\le \sum_{i=1}^{d}\mathrm{dist}(f_*,\mathcal{P}^i) && \text{by dimension reduction from [17]}\\
&\le \sum_{i=1}^{d}\mathrm{dist}(f_*,g^i) && \text{because } g^i\in\mathcal{P}^i\\
&\le \sum_{i=1}^{d}\mathrm{dist}(f_*,h^i)+\sum_{i=1}^{d}\mathrm{dist}(h^i,g^i) && \text{by triangle inequality}\\
&\le d\cdot\alpha\cdot n^d+\sum_{i=1}^{d}\mathrm{dist}(f,\mathcal{P}^i).
\end{aligned}$$

To see the last inequality, notice that $f_*$ and $h^i$ differ only on points in $[n]^d\setminus\mathcal{N}$. Hence, for all $i\in[d]$, we have, $\mathrm{dist}(f_*,h^i)\le\alpha\cdot n^d$. Also, for all $i\in[d]$, $\mathrm{dist}(f,\mathcal{P}^i)$ is defined as the minimum number of points in $\mathcal{N}$ that every completion of $f$ need to be changed to get a function in $\mathcal{P}^i$. Since $h^i$ is a completion of $f$ for all $i\in[d]$ and $g^i$ is the function that minimizes $\mathrm{dist}(g^i_{|\mathcal{N}},f_{|\mathcal{N}})$, we can see that $\mathrm{dist}(f,\mathcal{P}^i)\ge\mathrm{dist}(h^i,g^i)$ for all $i\in[d]$. ∎

We now use Claim 5.6 to prove the first part of our dimension reduction.

LEMMA 5.7 (Dimension reduction: distance). *Let $\varepsilon_f$ be the relative Hamming distance of $f$ from $\mathcal{P}$. Given $\ell\in\mathcal{L}$, let $\varepsilon_\ell$ denote the relative Hamming distance of $f_\ell$ from $\mathcal{P}^i_{\mathsf{line}}$, where $i\in[d]$ is the dimension along which $\ell$ lies. Then*

$$\mathbf{E}_{\ell\sim\mathcal{L}}[\varepsilon_\ell]\ge\frac{(1-\alpha)\cdot\varepsilon_f}{4d}-\alpha.$$

*Proof.* There are $d$ axis-parallel directions and, therefore, $dn^{d-1}$ axis-parallel lines in $[n]^d$. Thus, $\Pr[E_\ell]=1/dn^{d-1}$, where $E_\ell$ is the event of getting a specific axis parallel line $\ell$ while sampling u.a.r. from $\mathcal{L}$. Let $\mathcal{L}_i$ denote the set of axis parallel lines along

dimension $i$.

$$\begin{aligned}
\mathbf{E}_{\ell\sim\mathcal{L}}[\varepsilon_\ell] &= \sum_{\ell\in\mathcal{L}} \varepsilon_\ell \cdot \Pr[E_\ell] \\
&= \sum_{i=1}^{d} \sum_{\ell\in\mathcal{L}_i} \varepsilon_\ell \cdot \Pr[E_\ell] \\
&= \frac{1}{dn^{d-1}} \cdot \sum_{i=1}^{d} \sum_{\ell\in\mathcal{L}_i} \frac{\operatorname{dist}(f_\ell, \mathcal{P}^i_{\mathsf{line}})}{|\mathcal{N}_\ell|} \\
&\geq \frac{1}{dn^{d}} \cdot \sum_{i=1}^{d} \sum_{\ell\in\mathcal{L}_i} \operatorname{dist}(f_\ell, \mathcal{P}^i_{\mathsf{line}}) && \text{since } |\mathcal{N}_\ell| \leq n \\
&= \frac{1}{dn^{d}} \cdot \sum_{i=1}^{d} \operatorname{dist}(f, \mathcal{P}^i) \\
&\geq \frac{1}{dn^{d}} \cdot \left( \frac{\operatorname{dist}(f, \mathcal{P})}{4} - \alpha d \cdot n^d \right) && \text{by Claim 5.6} \\
&\geq \frac{1-\alpha}{4d} \cdot \varepsilon_f - \alpha.
\end{aligned}$$

∎

We conclude this section with the second part of our dimension reduction.

LEMMA 5.8 (Dimension reduction: fraction of erasures). *Consider an $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$. Given an axis-parallel line $\ell \in \mathcal{L}$, let $\alpha_\ell$ denote the fraction of erased points in $\ell$. Then, for every $\eta \in (0, 1)$,*

$$\Pr_{\ell\sim\mathcal{L}}[\alpha_\ell > \alpha/\eta] \leq \eta.$$

*Proof.* Note that a uniformly randomly sampled point in $[n]^d$ is erased with probability $\alpha$. We can sample a point uniformly at random by first sampling a line $\ell \in \mathcal{L}$ uniformly at random and then sampling a point uniformly randomly on $\ell$, which is erased with probability $\alpha_\ell$. Therefore we have

$$\alpha = \sum_{\ell\in\mathcal{L}} \Pr[E_\ell] \cdot \alpha_\ell = \mathbf{E}_{\ell\sim\mathcal{L}}[\alpha_\ell].$$

The claim then follows from Markov's inequality. ∎

**5.3. Erasure-Resilient BDP Testers for the Hypergrids.** We now present our erasure-resilient tester for an arbitrary BDP $\mathcal{P}$ and complete the proof of Theorem 1.7. Let $\mathbf{B} = \{\ell_i, u_i : i \in [d]\}$ be a bounding family for $\mathcal{P}$, where $\ell_i, u_i : [n-1] \mapsto \mathbb{R}$. Let $\mathcal{L}_i$ denote the set of axis-parallel lines along dimension $i$. Our tester is described in Algorithm 4.

The bound on the query complexity of the tester is evident from its description. We will now prove its correctness in Lemma 5.9, which will then imply Theorem 1.7. The proof of Lemma 5.9 is very similar to the proof of Lemma 4.4. The only difference is that we use an additional step in the analysis to reduce BDP testing to monotonicity testing over the line, given by Claim 5.3. This step introduces constant-factor differences in the mathematical expressions.

LEMMA 5.9. *Algorithm 4 accepts if $f$ is in $\mathcal{P}$, and rejects with probability at least $2/3$ if $f$ is $\varepsilon$-far from $\mathcal{P}$.*

**Algorithm 4** Erasure-Resilient Tester for BDP $\mathcal{P}$ over $[n]^d$

**Input:** parameters $\varepsilon \in (0,1), \alpha \in [0, \varepsilon/970d]$; oracle access to $f : [n]^d \to \mathbb{R} \cup \{\perp\}$

1: **Set** $Q = \left\lceil \frac{4800d \cdot \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2: **repeat** $\frac{48d}{\varepsilon(1-\alpha) - 4d\alpha}$ times:
3: Sample a line $\ell \in \mathcal{L}$ uniformly at random.
4: Define $g$ and $h$ from $f_\ell$, $\ell_i$ and $u_i$ as in Claim 5.3 if $\ell$ is sampled from $\mathcal{L}_i$.
5: Sample points u.a.r. from $\ell$ and query them until we get a point $s \in \mathcal{N}$.
6: Perform a randomized binary search for $s$ on $\ell$ as in Algorithm 2.
7: **Reject** if any violation to monotonicity is found in either $g$ or $h$.
8: **Accept** at any point if the number of queries exceed $Q$.

*Proof.* To prove the first part of the lemma, consider an $\alpha$-erased function $f \in \mathcal{P}$ and consider an arbitrary iteration of the tester. Suppose the tester samples a line $\ell \in \mathcal{L}$ such that $\ell$ is along the $i^{\text{th}}$ dimension. Let $g_\ell, h_\ell : [n] \mapsto \mathbb{R}$ denote the functions $g$ and $h$ obtained by applying Claim 5.3 to $f_\ell$ and $\mathcal{P}^i_{\mathsf{line}}$. By Claim 5.3, we know that $f_\ell$ is in $\mathcal{P}^i_{\mathsf{line}}$ iff both $h_\ell$ and $g_\ell$ are monotone. As is clear from Algorithm 4, the tester runs (one-sided error) erasure-resilient monotonicity testers for two such functions and therefore, the tester accepts $f$ in that iteration. Hence, the tester accepts $f$.

Consider an $\alpha$-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$ that is $\varepsilon$-far from $\mathcal{P}$. Let $A$ denote the event that the tester does not reject $f$ in any of its iterations. If $q$ denotes the total number of queries made by the tester, we have, $\Pr[A] \leq \Pr[A \mid q \leq Q] + \Pr[q > Q]$.

Let $t$ denote the number of iterations of the tester. Let $A_i$ denote the event that the tester accepts in its $i^{\text{th}}$ iteration. As before, let $E_\ell$ denote the event that the tester gets the line $\ell$ when it samples lines u.a.r. from $\mathcal{L}$. Let $\varepsilon_\ell$ denote the relative Hamming distance of $f_\ell$ from $\mathcal{P}^j_{\mathsf{line}}$, where $j$ is the index of the dimension along which $\ell$ lies. By Claim 5.3, either $g_\ell$ or $h_\ell$ is $\varepsilon_\ell/4$-far from monotone. Thus, the tester rejects with probability at least $\varepsilon_\ell/4$ if it samples $\ell$. We then have,

$$\Pr[A_i \mid q \leq Q] \leq \sum_{\ell \in \mathcal{L}} \left(1 - \frac{\varepsilon_\ell}{4}\right) \Pr[E_\ell] = 1 - \frac{1}{4} \cdot \mathbf{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell].$$

Using Lemma 5.7 and the fact that $\varepsilon_f \geq \varepsilon$, we have,

$$\mathbf{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell] \geq \frac{(1-\alpha) \cdot \varepsilon_f}{4d} - \alpha \geq \frac{(1-\alpha) \cdot \varepsilon}{4d} - \alpha.$$

Therefore,

$$\Pr[A \mid q \leq Q] = \prod_{i=1}^{t} \Pr[A_i \mid q \leq Q] \leq \left(1 - \frac{(1-\alpha) \cdot \varepsilon - 4d\alpha}{16d}\right)^t < \frac{1}{10}.$$

It now remains to bound $\Pr[q > Q]$. Let $\eta$ stand for $1/10t$. Let $\alpha_i$ denote the fraction of erasures in the line sampled during iteration $i$ and let $q_i$ denote the number of queries made by the algorithm during iteration $i$. Let $G$ denote the (good) event that $\alpha_i \leq \alpha/\eta$ for all iterations $i \in [t]$. By Corollary 3.4, $\mathbf{E}[q_i \mid G] \leq 5\eta \cdot \log n/(\eta - \alpha)$,

and by the linearity of expectation,

$$\mathbf{E}[q \mid G] \leq \frac{\log n}{2(\eta - \alpha)} \leq \frac{480d \log n}{\varepsilon(1-\alpha)},$$

where the last inequality follows from our assumption that $\alpha \leq \varepsilon/970d$. Using Markov's inequality,

$$\Pr[q > Q \mid G] \leq \frac{1}{10}.$$

Also, by combining Lemma 5.8 with a union bound, we can see that $\Pr[\overline{G}] \leq 1/10$. Therefore,

$$\Pr[q > Q] \leq \Pr[q > Q \mid G] + \Pr[\overline{G}] \leq \frac{1}{5}.$$

**5.4. Limitations of Dimension Reduction in Erasure-Resilient Testing.** In this section, we show that when the fraction of erasures is large enough, dimension reduction based testers that sample axis parallel lines uniformly at random and check for violations on them, are bound to fail. Axis-parallel lines in hypercubes (Definition 1.5) are called *edges*. We prove the following claim.

LEMMA 5.10. *For all $\varepsilon \in (0, 1/2]$, all large enough even $d$ and $\alpha = \Theta(\varepsilon/\sqrt{d})$, there exists an $\alpha$-erased function $f : \{0,1\}^d \mapsto \mathbb{R} \cup \{\perp\}$, such that $f$ is $\varepsilon$-far from monotone but $f$ has no violations to monotonicity along the edges of the hypercube $\{0,1\}^d$.*

*Proof.* For the ease of exposition, we prove this lemma for $\varepsilon = 1/2$. We note that similar calculations could extend this proof to any $\varepsilon \in (0, 1/2]$. Assume that $d$ is even. For $x \in \{0,1\}^d$, let $||x||_0$ denote the number of nonzero coordinates in $x$. The function $f$, for $x \in \{0,1\}^d$ is defined as:

$$f(x) = \begin{cases} \perp & \text{if } ||x||_0 = d/2 \\ 1 & \text{if } ||x||_0 < d/2 \\ 0 & \text{otherwise.} \end{cases}$$

Recall that the distance to monotonicity of a function $f : \{0,1\}^d \mapsto \{0,1\}$ is the fraction of nonerased function values that we need to change to make $f$ monotone. As $d$ is even, the number of points in $\{0,1\}^d$ above and below the middle layer are equal. We need to change function values at either all the points below the middle layer, or all the points above middle layer, to make $f$ monotone. Hence, $f$, as described above is $\frac{1}{2}$-far from monotone. Also, no axis-parallel edge is violated with respect to monotonicity as the middle layer is erased. This completes the proof for the case when $\varepsilon = 1/2$, since $\alpha = \Theta(1/\sqrt{d})$.

For general $\varepsilon$, we can define the set of erased points to be the points in $\{0,1\}^d$, such that their Hamming weight is $\beta \cdot d$, where $\beta = \beta(\varepsilon) < 1/2$ is chosen so that $|S|/2^d = \varepsilon$, where $S$ is the set of all points in $\{0,1\}^d$ with Hamming weight less than $\beta \cdot d$. As in the above case, we set all points with Hamming weight smaller than $\beta \cdot d$ to 1 and the ones with Hamming weight larger than $\beta \cdot d$ to be 0. Similar calculations help us prove that for large enough $d$, fraction of erased points is $\Theta(\varepsilon/\sqrt{d})$. In this case, $f$ is $\varepsilon$-far from monotone, but no axis-parallel edge is violated in $f$ with respect to monotonicity. ∎

**6. Erasure-Resilient Convexity Tester for the Line.** In this section, we prove Theorem 1.8. Given an $\alpha$-erased function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, let $\nu_i$ denote the $i$-th nonerased domain point in $[n]$. The derivative of $f$ at a point $\nu_i \in \mathcal{N}$, denoted by $\Delta f(\nu_i)$, is $\frac{f(\nu_{i+1})-f(\nu_i)}{\nu_{i+1}-\nu_i}$, whenever $\nu_{i+1} \leq n$. The function $f$ is convex iff $\Delta f(\nu_i) \leq \Delta f(\nu_{i+1})$ for all $i \in [|\mathcal{N}| - 2]$. In other words, a function is convex iff its derivative is monotone.

Looking at the above definition, it would seem that testing convexity of a function can be reduced to testing monotonicity of its derivative. However, this reduction does not work even for the case of testing when there are no erasures, since there are functions that are very far from convex but whose derivatives are very close to monotone. One such example, given in [43], is the following function $f : [n] \mapsto \mathbb{R}$ defined for even $n$. For $i \leq n/2$, we have $f(i) = i$ and for $i > n/2$, we have $f(i) = i-1$. This function is $\frac{1}{2}$-far from convex. But its derivative, $\Delta f$, takes the value 1 on all points except for a single point, where it is 0. Hence, $\Delta f$ is $\frac{1}{n-1}$-close to monotone.

The authors of [43] then describe a convexity tester by utilizing the relationship between convexity of a function and the monotonicity of its derivative more cleverly. Our tester builds upon the ideas of the convexity tester from [43].

A high-level idea of the tester is as follows. Our tester (Algorithm 5) has several iterations. Every iteration of the tester can be thought of as a traversal of a uniformly random rooted path in a random binary search tree $\mathcal{T}$ on $\mathcal{N}$ sampled from the distribution $\mathcal{D}_{\mathcal{T}}$, just as Algorithm 2. For each interval on such a path, we check a set of conditions computed based on the values at some nonerased points in the interval, called *anchor points*, and two real numbers, called the left and right slopes. More specifically, we verify that the function restricted to the sampled nonerased points in the interval is convex, by comparing the slopes across consecutive points. The algorithm accepts if all the intervals it sees pass these checks. The main steps in the

**Algorithm 5** Erasure-Resilient Convexity Tester

**Input:** parameters $\varepsilon \in (0,1), \alpha \in [0,1)$; oracle access to $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$.

1: Set $Q = \left\lceil \frac{180 \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2: **Accept** at any point if the number of queries exceeds $Q$.
3: **repeat** $\frac{2}{\varepsilon}$ times
4: Sample points in $I[1,n]$ u.a.r and query them until we get a point $s \in \mathcal{N}$.
5: TEST-INTERVAL$(I[1,n], \emptyset, -\infty, +\infty, s)$ and **Reject** if it rejects.
6: **Accept**.

analysis of the tester follow that of the analysis of Algorithm 2. To analyze the tester, we first prove that, with high probability, the algorithm does not run out of its budget of queries $Q$. For this, we classify the queries that the tester makes into two kinds as follows and analyze them separately.

DEFINITION 6.1 (Sampling queries). *The queries made by the tester when it repeatedly samples and queries points from an interval until it finds a nonerased domain point are called* sampling queries.

DEFINITION 6.2 (Walking queries). *The queries made by the tester when it keeps querying consecutive points from intervals, starting from one nonerased point until it finds the next nonerased point, are called* walking queries.

In the proof of Lemma 6.3, we first show that the expected number of walking queries

is at most twice the number of the expected number of the sampling queries and then use Corollary 3.4 to bound the expected number of sampling queries.

In the second part of the analysis we prove that, conditioned on the total number of queries made by the algorithm not exceeding $Q$, in each iteration, with probability at least $\varepsilon$, the tester rejects a function that is $\varepsilon$-far from being convex. This part of the proof draws ideas from the proof of correctness of the tester in [43].

**Procedure 6** TEST-INTERVAL$(I[i,j], \mathcal{A}, m_\ell, m_r, s)$

**Input:** interval $I[i,j]$; a set of nonerased points $\mathcal{A}$; left slope $m_\ell \in \mathbb{R}$; right slope $m_r \in \mathbb{R}$; search point $s \in \mathcal{N}$.

1: Sample points u.a.r. from $I[i,j]$ and query them until we get a point $x \in \mathcal{N}$.
2: Sequentially query points $x+1, x+2 \ldots$ until we get a nonerased point $y$.
3: ▷ Set $y \leftarrow x$ if there is no nonerased point in $I[i,j]$ to the right of $x$.
4: Sequentially query points $x-1, x-2 \ldots$ until we get the nonerased point $z$.
5: ▷ Set $z \leftarrow x$ if there is no nonerased point in $I[i,j]$ to the left of $x$.
6: Let $(a_1, a_2, \ldots, a_k)$ denote the sorted list of points in the set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{x, y, z\}$.
7: Let $m_i = (f(a_{i+1}) - f(a_i))/(a_{i+1} - a_i)$ for all $i \in [k-1]$.
8: **Reject** if $m_\ell \le m_1 \le m_2 \le \cdots \le m_{k-1} \le m_r$ is not true.
9: Let $\mathcal{A}'_\ell$ and $\mathcal{A}'_r$ be the sets of points in $\mathcal{A}'$ that are smaller and larger than $x$, respectively.
10: **if** $s < x$ **then**
11: **Reject** if TEST-INTERVAL$(I[i,z], \mathcal{A}'_\ell, m_\ell, \Delta f(z), s)$ rejects.
12: **if** $s > x$ **then**
13: **Reject** if TEST-INTERVAL$(I[y,j], \mathcal{A}'_r, \Delta f(x), m_r, s)$ rejects.
14: **Accept**.

LEMMA 6.3. *Algorithm 5 accepts if $f$ is convex, and rejects with probability at least $2/3$ if $f$ is $\varepsilon$-far from convex.*

*Proof.* We first define some notation for our analysis. Consider a search path traversed by the algorithm. Similar to the analysis of Algorithm 2, this path can be viewed as a uniformly random rooted path in a binary tree $\mathcal{T}$ over $\mathcal{N}$, sampled according to $\mathcal{D}_\mathcal{T}$. Let $I[i,j]$ be an interval on the path. Consider the execution of TEST-INTERVAL (Procedure 6) called with $I[i,j]$ as the first argument. We call the nonerased point $x$ sampled in Step 1 of Procedure 6 its *pivot*, the set of points $\mathcal{A}'$ in Step 6 of Procedure 6 its *anchor set* and the values $m_\ell$ and $m_r$ (passed to the procedure TEST-INTERVAL) as its *left* and *right slopes*, respectively. That is, given a binary search tree $\mathcal{T}$ over $\mathcal{N}$, we associate each node in the tree with an interval, a pivot, an anchor set and two slopes. Note that the size of the anchor set $\mathcal{A}'$ in Step 6 is at most 5, since each interval can have at most two anchor points of its ancestors carried down to it (the extreme nonerased points of the interval), and also have at most 3 of its own anchor points.

It is evident that the tester accepts whenever $f$ is convex. To prove the other part of the lemma, assume that $f$ is $\varepsilon$-far from being convex. Let $A$ be the event that the tester accepts $f$. Let $q$ denote the total number of queries made. We have, $\Pr[A] \le \Pr[A \mid q \le Q] + \Pr[q > Q]$.

We first bound $\Pr[q > Q]$. As mentioned earlier, the queries made by the tester can be classified into sampling queries (Definition 6.1) and walking queries (Definition 6.2). By Corollary 3.4, the expected number of sampling queries made in one

iteration of the tester is at most $5 \log n/(1-\alpha)$.

We will now bound the expected number of walking queries. Consider an interval $I$ with $\alpha_I$ fraction of erasures in it. A point in $I$ can get queried as part of the walking queries if either the first nonerased point to its right or the first nonerased point to its left on the line $[n]$ gets sampled as the pivot of $I$. For a nonerased point $i \in I$, let $w(i)$ denote the number of walking queries to be made if the algorithm samples $i$ as the pivot. Therefore

$$\sum_{i \in \mathcal{N} \cap I} w(i) \leq 2|I|,$$

since every point in $I$ gets counted at most twice in this sum. There are exactly $|I|(1-\alpha_I)$ nonerased points in $I$ and each of them could be the pivot in $I$ with equal probability. Hence, the expected number of walking queries that Algorithm 5 makes in $I$ is at most $2/(1-\alpha_I)$. This is at most twice the expected number of sampling queries that the algorithm makes in $I$.

Therefore, by the linearity of expectation, the expected number of walking queries made in one iteration of the tester is at most $10 \log n/(1-\alpha)$. Thus, the expected value of the total number of queries made by the tester in one iteration is at most $15 \log n/(1-\alpha)$ and that over all iterations is at most $30 \log n/\varepsilon(1-\alpha)$. Thus, by Markov's inequality,

$$\Pr[q > Q] \leq \frac{1}{6}.$$

Next, we bound $\Pr[A \mid q \leq Q]$. Consider a binary search tree $\mathcal{T}$ over $\mathcal{N}$, sampled according to the distribution $\mathcal{D}_{\mathcal{T}}$, and a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$. Let $\Gamma(I[i,j], \mathcal{A}, m_\ell, m_r)$ be a node in $\mathcal{T}$ with interval $I[i,j]$, anchor set $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ and slopes $m_\ell$ and $m_r$ such that $a_i \leq a_{i+1}$ for all $i \in [k-1]$. Let $m_i = (f(a_{i+1}) - f(a_i))/(a_{i+1} - a_i)$ $\forall i \in [k-1]$.

Definition 6.4 (Good Node, Bad Node). *A node $\Gamma(I[i,j], \mathcal{A}, m_\ell, m_r)$ is* good *if $m_\ell \leq m_1 \leq m_2 \leq \cdots \leq m_{k-1} \leq m_r$. Otherwise, it is* bad.

Definition 6.5 (Violator Node). *An node $\Gamma$ is a* violator *if it is* bad *and all its ancestor nodes in $\mathcal{T}$ are* good.

Definition 6.6 (Witness). *A nonerased domain point is a* witness *with respect to $\mathcal{T}$ if it belongs the interval associated with a violator node in $\mathcal{T}$.*

We prove that if $f$ is $\varepsilon$-far from being convex, then, for every binary search tree $\mathcal{T}$, the fraction of nonerased domain points that are witnesses is at least $\varepsilon$. We start by assuming that there is a tree in which the fraction of witnesses is less than $\varepsilon$. We show that we can correct the function values only on the witnesses and get a convex function, which gives a contradiction.

Claim 6.7. *If $f$ is $\varepsilon$-far from convex, then the fraction of witnesses in every binary search tree $\mathcal{T}$ is at least $\varepsilon$.*

*Proof.* Assume for the sake of contradiction that there is a binary search tree $\mathcal{T}$ such that the fraction of witnesses with respect to $\mathcal{T}$ is less than $\varepsilon$. In the following, we will construct a convex function $g : [n] \mapsto \mathbb{R} \cup \{\perp\}$ by changing the values of $f$ only on witnesses with respect to $\mathcal{T}$. Since the fraction of witnesses is less than $\varepsilon$, functions $f$ and $g$ will differ on less than an $\varepsilon$ fraction of nonerased domain points, which will give us the desired contradiction.

Consider a violator node $\Gamma$ in $\mathcal{T}$ and let $I[i,j]$ be the interval associated with it. If $\Gamma$ is the root of $\mathcal{T}$, every nonerased domain point in $f$ is a witness by definition. This

contradicts our assumption that the fraction of witnesses is less than $\varepsilon$. Therefore, we can assume that $\Gamma$ is a non-root node in $\mathcal{T}$. Let the anchor set and slopes associated with the *parent node* of $\Gamma$ in $\mathcal{T}$ be $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$, $m_\ell$ and $m_r$, respectively. Assume without loss of generality that $a_i \leq a_{i+1}$ for all $i \in [k-1]$. Suppose that $\Gamma$ is the right child of its parent. The case when $\Gamma$ is the left child of its parent is analogous and is hence omitted. Let $\{a_u, a_{u+1}, \ldots, a_k\}$ be the set of points common to $I[i,j]$ and $\mathcal{A}$. By definition, $a_u$ is the smallest nonerased domain point in $I[i,j]$. Also, the left slope of $I[i,j]$ is $(f(a_u) - f(a_{u-1}))/(a_u - a_{u-1})$ and its right slope is equal to $m_r$.

Let $m_v = (f(a_{v+1}) - f(a_v))/(a_{v+1} - a_v)$ for all integers $v$ such that $v \in [u-1, k)$. We define $g$ as follows.

- For each $t \in \{a_u, a_{u+1}, \ldots, a_k\}$, set $g(t) = f(t)$ .
- For each integer $v \in [u, k)$ and $t \in \mathcal{N} \cap (a_v, a_{v+1})$, set

$$g(t) = f(a_v) + m_v \cdot (t - a_v)$$

- For each $t \in \mathcal{N}$ such that $j \geq t > a_k$, set

$$g(t) = f(a_k) + m_{k-1} \cdot (t - a_k).$$

Since $\Gamma$ is a violator node, the parent node of $\Gamma$ is good, by definition. This implies that $m_{u-1} \leq m_u \leq \ldots \leq m_{k-1} \leq m_r$. Therefore, the derivatives of nonerased points in $I[i,j]$ are non-decreasing with respect to $g$, by virtue of our assignment.

To prove that $g$ is convex, we first show that every node in $\mathcal{T}$ is good with respect to $g$.

1. Consider a node $\Gamma$ in $\mathcal{T}$ that is good with respect to $f$. Let $I[i,j]$ be the interval associated with $\Gamma$. If $\Gamma$ has no ancestors or descendants that are violators (w.r.t. $f$), it remains good with respect to $g$ as well, since $g(t) = f(t)$ for all $t \in I[i,j]$.
2. Consider a node $\Gamma$ in $\mathcal{T}$ such that $\Gamma$ and its ancestors are all good w.r.t. $f$. Let $I$ be the interval associated with $\Gamma$. To prove that $\Gamma$ is good w.r.t. $g$, it is enough to show that $f(t) = g(t)$ for every anchor point $t \in I$ of $\Gamma$. Note that the only points $t \in I$ for which $f(t)$ and $g(t)$ could be different are the points belonging to intervals associated with violator nodes in $\mathcal{T}$ that are descendants of $\Gamma$. Consider a node $\Gamma'$ in $\mathcal{T}$ such that (1) $\Gamma'$ is a descendant of $\Gamma$, and (2) $\Gamma'$ is a violator node w.r.t. $f$. Let $I'$ be the interval associated with $\Gamma'$. The definition of $g$ on points in $I'$ ensures that $g(t) = f(t)$ for every point $t$ common to the anchor sequence of $\Gamma$ and the interval $I'$. Thus, we can see that $f(t) = g(t)$ for every anchor point $t \in I$ of $\Gamma$. Hence, $\Gamma$ remains good with respect to $g$.
3. Consider a node $\Gamma$ that is either a violator node or has a violator ancestor $\Gamma$ (w.r.t. $f$). Let $I$ and $I'$ be the intervals associated with $\Gamma$ and $\Gamma'$ respectively. By definition, the parent of $\Gamma'$ is good with respect to $f$. Therefore, by the definition of $g$ on $I'$, we have $\Delta g(t-1) \leq \Delta g(t)$ for all $t \in \mathcal{N} \cap I'$. Therefore, $\Gamma'$ is good with respect to $g$, and hence $\Gamma$ is also good with respect to $g$. □

We proved that every node in the tree $\mathcal{T}$ is good with respect to $g$. We now prove that $g$ is convex. Consider a point $\nu_t \in \mathcal{N}$ such that $2 \leq t \leq |\mathcal{N}| - 1$, where $\nu_i$ denotes the $i^{\text{th}}$ nonerased point in $[n]$. This point occurs in $\mathcal{T}$ either as the pivot of a non-leaf node or as the sole nonerased domain point in the interval associated with a leaf node. In the former case, the condition $\Delta g(\nu_{t-1}) \leq \Delta g(\nu_t)$ is part of the *goodness condition*

of the corresponding node and is satisfied. In the latter case, $\Delta g(\nu_{t-1})$ and $\Delta g(\nu_t)$ are the left and right slopes of the leaf and are compared as part of the goodness condition of the leaf. Thus, $\Delta g(\nu_{t-1}) \leq \Delta g(\nu_t)$ for all $\nu_t \in \mathcal{N}$ such that $2 \leq t \leq |\mathcal{N}| - 1$. Thus, $g$ is convex. ∎

We conclude our analysis by bounding the probability that the tester does not find a violation. Since the search point $s$ is chosen uniformly at random from the set of nonerased domain points, the probability that it is a witness is at least $\varepsilon$ and thus, the tester detects a violation to convexity with probability at least $\varepsilon$ in every iteration. Therefore,

$$\Pr[A \mid q \leq Q] \leq (1-\varepsilon)^{\frac{2}{\varepsilon}} < \frac{1}{6}.$$

**7. Relations to Other Testing Models.** In this section, we describe the relationships between erasure-resilient testing model and the other models of property testing. We first describe a property that is easy to test in the standard model, but is hard to test in the erasure-resilient model. This effectively separates the erasure-resilient testing model from the standard model. We discuss this result in Section 7.1. Next, we study the connection of erasure-resilient testing to tolerant testing and distance approximation algorithms and show that the existence of a distance approximation algorithm or a tolerant tester for a property implies the existence of an erasure-resilient testing algorithm for that property. We describe it in Section 7.2.

**7.1. Separation Between Erasure-Resilient and Standard Testing.** In this section, we show that erasure-resilient testing is, in general, harder than standard testing and prove Theorem 1.3. More formally, we construct a property $R$ such that it only takes a constant number of queries to test $R$ in the standard model, whereas $\alpha$-erasure-resilient $\frac{1}{4}$-testing of $R$ requires polynomially many queries for all constant $\alpha \in (0,1)$.

Our proof of Theorem 1.3 follows closely, the proof that Fischer and Fortnow [30] used to separate tolerant property testing [44] from standard property testing. In fact, the property $R$ guaranteed by Theorem 1.3 is the property constructed by [30] for their purposes. We first give a high-level description of the property $R$ and an intuition of why $R$ is easy to test in the standard property testing model (proven by [30]) and hard to test in the erasure-resilient property testing model (proved in this section).

Property $R$ is constructed from another property $H$ such that the query complexity of testing $H$ in the standard model is linear in the input size. However, $H$ has the feature that, given oracle access to an additional *proof* string, it takes only a constant number of queries to test $H$. Note that the query complexity in the latter scenario includes the number of queries made to the proof. In loose terms, the property $R$ is defined as the set of all strings where the first part consists of repetitions of a string that satisfies $H$, and the second part is a proof of the membership of the first part in $H$. It follows that $R$ is easy to test in the standard model. However, if the *proof* part of an input string is erased, then erasure-resilient testing of $R$ reduces to testing of $H$ without access to a proof, which demands a high query complexity.

The following lemma from [8] shows the existence of a property $H$ that is hard to test in the standard model.

LEMMA 7.1 ([8]). *There exists a property $H$ of $m$-bit strings that is decidable in polynomial time but any $\frac{1}{3}$-test of which requires at least $\Omega(m)$ queries.*

The main tool used in constructing the property $R$ from the above property $H$ is algebraic objects called PCPs of proximity (also called assignment testers [21]), which were defined by [7]. PCPs of proximity are proof systems for $\varepsilon$-testing promise problems. We state the definition given in [30].

DEFINITION 7.2 (PCP of proximity [7, 30]). *Let $\mathcal{P}$ be a property of strings in $\{0,1\}^m$ and $\varepsilon \in (0,1)$ be a parameter. A (one-sided error) PCP of proximity for the $\varepsilon$-testing of $\mathcal{P}$ is a set of Boolean functions $f_1, \ldots, f_l$, where $l$ is polynomial in $m$, such that:*

- *For all $i \in [l]$, the number of variables that $f_i$ depends on is independent of $m$. The variables that $f_i$ depends on is a subset of $\{x_i, y_j : i \in [m], j \in [r]\}$, where the $x_i$'s are the variables corresponding to the positions in the input $\vec{x}$ (to the $\varepsilon$-testing problem), $y_j$'s are a set of auxiliary variables and $r$ is polynomial in $m$.*
- *For $\vec{x} \in \{0,1\}^m$ that is in $\mathcal{P}$, there is an assignment to $y_j$'s such that all $f_i$'s are satisfied.*
- *For $\vec{x} \in \{0,1\}^m$ that is $\varepsilon$-far from $\mathcal{P}$ and for every assignment of values to $y_j$'s, at most half of the $f_i$'s are satisfied.*

Therefore, if there exists a PCP of proximity for the problem of $\varepsilon$-testing $H$, we can use assignments to the $y$ variables in the PCP as a candidate proof string for $\varepsilon$-testing membership in $H$ (for the same value of $\varepsilon$). The following theorem from [7] states that there exist PCPs of proximity for all properties decidable in polynomial time and, in particular, for the property $H$, as by Lemma 7.1, it is polynomial time decidable.

LEMMA 7.3 (from [7], as restated by [30]). *Let $\mathcal{P}$ be a property of strings in $\{0,1\}^m$ that is decidable by a circuit of size $k$, and $t$ be such that $t < \frac{\log\log k}{\log\log\log k}$. There exists a PCP of proximity for the problem of $(1/t)$-testing of $\mathcal{P}$. Moreover, the number of additional variables and the number of functions in the PCP of proximity are both bounded by $k^2$, and each function depends on $O(t)$ variables.*

Now we describe the property $R$ that is hard to test when there are adversarial erasures in the input. Let $p(m)$ be a polynomial bound on the size of a circuit deciding $H$ on instances of size $m$, where $H$ is the property given by Lemma 7.1. Let $m$ be large enough such that $\lfloor \log\log\log p(m) \rfloor < \frac{\log\log p(m)}{\log\log\log p(m)}$. Let $t = \lfloor \log\log\log p(m) \rfloor$.

Consider a bit string of length $n = m \cdot (p(m))^2$. This string satisfies the property $R$ if the first $(m-t)(p(m))^2$ bits of the string are repetitions of a string $\vec{x} \in \{0,1\}^m$ satisfying $H$, and the remaining $t(p(m))^2$ bits are the satisfying assignments to the $y$ variables in the PCPs of proximity for testing $H$ on $\vec{x}$ with various distance parameters larger than $1/t$. More formally, label the first $(m-t)(p(m))^2$ bits by $x_{i,j}$ where $i \in [(m-t)(p(m))^2/n]$ and $j \in [m]$. Label the remaining bits by $y_{i,j}$ where $i \in [(p(m))^2]$ and $j \in [t]$. The string is said to have the property $R$ if all of the following conditions hold:

- $x_{1,1} \ldots x_{1,m}$ is a string that satisfies the property $H$.
- For each $1 < i \leq (m-t)(p(m))^2/n$ and $j \in [m]$, we have $x_{1,j} = x_{i,j}$.
- For every $j \in [t]$, the sequence $y_{1,j}, \ldots, y_{(p(m))^2,j}$ is an assignment satisfying the PCP of proximity for $(1/j)$-testing of $x$ for the property $H$.

As mentioned earlier, the property $R$ described above is the property used by [30] to separate tolerant testing from standard testing. We use the fact (from [30]) that $R$ is easy to test in the standard model, without a proof.

THEOREM 7.4 ([30]). *Property $R$ can be $\varepsilon$-tested in the standard property testing model using $O(1/\varepsilon)$ queries.*

We now prove the following theorem which, together with Theorem 7.4, implies Theorem 1.3.

THEOREM 7.5. *There exists some $c > 0$ such that for all constant $\alpha \in (0,1)$ every $\alpha$-erasure-resilient $\frac{1}{4}$-tester for $R$ makes $\Omega(n^c)$ queries, where $n$ is the size of the input.*

*Proof.* The proof is by a simple reduction from $\frac{1}{3}$-testing of $H$ in the standard model, which, by Lemma 7.1, has a high query complexity.

Given a string $x_1x_2\dots x_m \in \{0,1\}^m$ for which we need to $\frac{1}{3}$-test $H$, we can construct a partially erased string $I$ of length $n = m(p(m))^2$ as follows, where $p(m)$ denotes the polynomial bound on the size of a circuit computing $H$. Let $t = \lfloor \log\log\log p(m) \rfloor$. Let $x_{i,1}x_{i,2}\dots x_{i,m}$ be set to the string $x_1x_2\dots x_m$ for all $i \in [(m-t)(p(m))^2/m]$, where $x_{i,1}x_{i,2}\dots x_{i,m}$ denotes the $i$-th block of $m$ bits in $I$ from the left. Let the remaining bits of $I$ be set to the erased symbol $\perp$. A query to this new string can be simulated by at most one query to the string $x_1x_2\dots x_m$ (which we have query access to).

If $x_1x_2\dots x_m$ satisfies $H$, then the new string $I$ satisfies $R$ by the definition of erasure-resilient property testing model and the fact that $\varepsilon$-testing of $H$ has PCPs of proximity for various $\varepsilon > \frac{1}{t}$. If $x_1x_2\dots x_m$ is $\frac{1}{3}$-far from satisfying $H$, then $I$ is $(1-\frac{t}{m})\cdot\frac{1}{3}$-far from $R$ since each $m$-length block among the first $(m-t)(p(m))^2$ of $I$ is $\frac{1}{3}$-far from being in $H$. Note that $(1-\frac{t}{m}) \geq \frac{1}{4}$ for large enough $m$. The fraction of erasures in $I$ is $t/m$, which is $O(\frac{\log\log\log m}{m})$, smaller than every constant $\alpha$. Therefore, an $\alpha$-erasure resilient $\frac{1}{4}$-tester for $R$ for constant $\alpha$ yields a $\frac{1}{3}$-tester for $H$ with the same query complexity. Since every $\frac{1}{3}$-tester for $H$ requires $\Omega(m)$ queries on inputs of length $m$, every $\alpha$-erasure-resilient $\frac{1}{4}$-tester for $R$ requires $\Omega(m)$ queries. As $n = m\cdot(p(m))^2$, this implies that every $\alpha$-erasure-resilient $\frac{1}{4}$-tester for $R$ requires $\Omega(n^c)$ queries, where $c$ is a constant that depends on the degree of the polynomial $p(m)$. ∎

**7.2. Connections to Distance Approximation Algorithms.** In this section, we show that the existence of a fully tolerant tester for a property implies the existence of an erasure-resilient tester for that property, and prove Theorem 1.4. We also state and prove a slightly different version of Theorem 1.4 for distance approximation algorithms, and apply that latter version to design erasure-resilient testers for sortedness, monotonicity, and convexity.

*Proof of Theorem 1.4.* Let $e$ be an arbitrary element in the range $\mathcal{R}$. For an $\alpha$-erased function $f$, let $f_e^r$ denote the completion of $f$ obtained by assigning the value $e$ to all the erased points.

An $\alpha$-erasure-resilient $\varepsilon$-tester $A'$ for $\mathcal{P}$, when given oracle access to an $\alpha$-erased function $f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$, simulates oracle access to the the function $f_e^r$, runs algorithm $A$ on $f_e^r$ with proximity parameters $\varepsilon_1 = \alpha, \varepsilon_2 = \varepsilon(1-\alpha)$ and accepts if and only if $A$ accepts. The condition $\varepsilon_1 < \varepsilon_2$ is satisfied by the restriction on $\alpha$.

If $f$ satisfies $\mathcal{P}$ there is a completion $f'$ of $f$ that satisfies $\mathcal{P}$ and every other completion of $f$ is $\alpha$-close to $f'$. Therefore, $f_e^r$ is $\alpha$-close to $\mathcal{P}$. Since $\varepsilon_1 = \alpha$ and $A$ accepts with probability at least $2/3$, every function that is $\varepsilon_1$-close to $\mathcal{P}$, the tester $A'$ also accepts $f$ with probability at least $2/3$.

If $f$ is $\varepsilon$-far from $\mathcal{P}$, then every completion of $f$ is $\varepsilon(1-\alpha)$-far from $\mathcal{P}$. This is, in particular, true for $f_e^r$. As $\varepsilon_2 = \varepsilon(1-\alpha)$, and $A$ rejects with probability at least

2/3, every function that is $\varepsilon_2$-far from $\mathcal{P}$, the tester $A'$ rejects $f$ with probability at least 2/3. ∎

Tolerant testers are intimately connected to algorithms that approximate the distance of a function to a specified property, when given oracle access to the function. For a property $\mathcal{P}$ and a function $f$, we denote by $\varepsilon_{\mathcal{P}}(f)$ the relative Hamming distance of $f$ to $\mathcal{P}$. Let $\eta \geq 1$ and $\delta \in [0,1)$. An algorithm $A$ is a *$\eta$-multiplicative $\delta$-additive distance approximation algorithm* for $\mathcal{P}$, if, given oracle access to a function $f$, the algorithm outputs, with probability at least 2/3, a value $\hat{\varepsilon}$ such that $\frac{1}{\eta} \cdot \varepsilon_{\mathcal{P}}(f) - \delta \leq \hat{\varepsilon} \leq \varepsilon_{\mathcal{P}}(f)$. If $A$ works for all $\delta \in [0,1)$, we call it an $\eta$- multiplicative distance approximation algorithm.

Parnas et al. [44] prove that the existence of a distance approximation algorithm for a property imply the existence of a tolerant tester for the same property. They also show that the existence of a fully tolerant tester for a property implies the existence of a distance approximation algorithm for the same property. Tolerant testers for many of the properties discussed in Section 1.3 are usually expressed as distance approximation algorithms. We now prove that the existence of a distance approximation algorithm for a property implies the existence of an erasure-resilient tester for that property (that works for a restricted range of parameters). Due to the equivalence between distance approximation and tolerant testing, the following theorem can be seen as a different version of Theorem 1.4.

THEOREM 7.6. *Let $A$ be an $\eta$-multiplicative $\delta$-additive distance approximation algorithm for a property $\mathcal{P}$ of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$. Then there exists an $\alpha$-erasure-resilient $\varepsilon$-tester $A'$ for $\mathcal{P}$ that makes the same number of queries as $A$ and works for all $\varepsilon \in (0,1), \alpha \in [0,1)$ satisfying $\alpha < \frac{\varepsilon - \delta \cdot \eta}{\varepsilon + \eta}$.*

*Proof.* Fix an element $e \in \mathcal{R}$. As before, let $f_e^r$ denote the completion of an $\alpha$-erased function $f$ where the erased points are assigned the value $e$.

Consider the following algorithm $A'$. The algorithm $A'$, when given oracle access to an $\alpha$-erased function $f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$, simulates oracle access to $f_e^r$ and runs the tester $A$ on $f_e^r$. Let $\hat{\varepsilon}$ denote the estimate that $A$ computes at the end of its execution. If $\hat{\varepsilon} \leq \alpha$, the algorithm $A'$ accepts. Otherwise, it rejects.

If an $\alpha$-erased function $f$ satisfies $\mathcal{P}$, then $\varepsilon_{\mathcal{P}}(f_e^r) \leq \alpha$. Since $\hat{\varepsilon} \leq \varepsilon_{\mathcal{P}}(f_e^r)$ with probability at least 2/3, the algorithm $A'$ will accept $f$ with probability at least 2/3.

If $f$ is $\varepsilon$-far from $\mathcal{P}$, then every completion of $f$ is $\varepsilon(1-\alpha)$-far from $\mathcal{P}$, and hence $\varepsilon_{\mathcal{P}}(f_e^r) \geq \varepsilon(1-\alpha)$. Since $\hat{\varepsilon} \geq \frac{\varepsilon_{\mathcal{P}}(f_e^r)}{\eta} - \delta \geq \frac{\varepsilon(1-\alpha)}{\eta} - \delta > \alpha$ with probability at least 2/3, the algorithm $A'$ will reject $f$ with probability at least 2/3. Note that the last inequality in the above expression follows from the restriction on $\alpha$. ∎

We now revisit the properties discussed in Section 1.3 for which distance approximation algorithms are known and apply Theorem 7.6 to those algorithms and obtain erasure-resilient testers. The parameters of these testers are much worse than what we obtained in previous sections, especially in terms of the restrictions on $\alpha$.

COROLLARY 7.7 ([48]). *Let $\eta \in (1,2)$. There exists an $\alpha$-erasure-resilient $\varepsilon$-tester for monotonicity of real-valued functions over $[n]$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$ such that $\alpha < \frac{\varepsilon}{\varepsilon+\eta}$, with query complexity $O((\frac{1}{\varepsilon(\eta-1)})^{O(\frac{1}{\eta-1})} \cdot \log^c n)$ (where $c$ is a large absolute constant).*

COROLLARY 7.8 ([28]). *Let $\delta \in [0,1]$. There exists an $\alpha$-erasure-resilient $\varepsilon$-tester for monotonicity of real-valued functions over $[n]^d$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$ such that $\alpha < \frac{\varepsilon - 5\delta \cdot d^2 \log n}{\varepsilon + 5d^2 \log n}$, with query complexity $\tilde{O}\left(\frac{\log n}{\delta^4}\right)$.*

COROLLARY 7.9 ([27]). *There exists an $\alpha$-erasure-resilient $\varepsilon$-tester for convexity of real-valued functions over $[n]$ that works for all $\alpha \in [0,1), \varepsilon \in (0,1)$ such that $\alpha < \frac{\varepsilon}{\varepsilon+25}$, with query complexity $\tilde{O}\left(\frac{\log n}{\varepsilon}\right)$.*

**8. Conclusions and Open Problems.** In this paper, we initiate a study of property testing in the presence of adversarial erasures. We design efficient erasure-resilient testers for several important properties such as monotonicity, the Lipschitz properties and convexity over different domains. All our testers for properties of functions on the line domain work for an arbitrary fraction of erasures. All our testers have only a small additional overhead of $O(1/(1-\alpha))$ in their query complexity in comparison to the query complexity of the currently best, and, in some cases, optimal, standard testers for the same properties. We also show that not all properties are easy to test in the erasure-resilient testing model by proving the existence of a property that is easy to test in the standard model but hard to test in the erasure-resilient model even for a small fraction of erasures. We now list some open problems.

- We show that tolerant testing is at least as hard as erasure-resilient testing. Determining if tolerant testing is strictly harder than erasure-resilient testing is an interesting direction.
- The fraction of erasures that our monotonicity tester for hypergrid domains ($[n]^d$) can tolerate decreases inversely with $d$. We also show that an inverse dependence on $\sqrt{d}$ is necessary for testers that work by sampling axis-parallel lines uniformly at random and then test for the property on them. It is an interesting combinatorial question to determine the exact tradeoff between the fraction of erasures and the fraction of axis parallel lines that are far from monotone.

**Acknowledgments.** We thank the anonymous referees for comments that helped greatly improve the presentation of this article.

REFERENCES

[1] N. AILON AND B. CHAZELLE, *Information theory in property testing and monotonicity testing in higher dimension*, Inf. Comput., 204 (2006), pp. 1704–1717, http://dx.doi.org/10.1016/j.ic.2006.06.001, http://dx.doi.org/10.1016/j.ic.2006.06.001.

[2] N. AILON, B. CHAZELLE, S. COMANDUR, AND D. LIU, *Estimating the distance to a monotone function*, Random Struct. Algorithms, 31 (2007), pp. 371–383, http://dx.doi.org/10.1002/rsa.20167, http://dx.doi.org/10.1002/rsa.20167.

[3] P. AWASTHI, M. JHA, M. MOLINARO, AND S. RASKHODNIKOVA, *Testing Lipschitz functions on hypergrid domains*, Algorithmica, 74 (2016), pp. 1055–1081, http://dx.doi.org/10.1007/s00453-015-9984-y, http://dx.doi.org/10.1007/s00453-015-9984-y.

[4] M. BALCAN, E. BLAIS, A. BLUM, AND L. YANG, *Active property testing*, in 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012, 2012, pp. 21–30, http://dx.doi.org/10.1109/FOCS.2012.64, http://dx.doi.org/10.1109/FOCS.2012.64.

[5] T. BATU, L. FORTNOW, R. RUBINFELD, W. D. SMITH, AND P. WHITE, *Testing closeness of discrete distributions*, J. ACM, 60 (2013), pp. 4:1–4:25, http://dx.doi.org/10.1145/2432622.2432626, http://doi.acm.org/10.1145/2432622.2432626.

[6] T. BATU, R. RUBINFELD, AND P. WHITE, *Fast approximate PCPs for multidimensional bin-packing problems*, Inf. Comput., 196 (2005), pp. 42–56, http://dx.doi.org/10.1016/j.ic.2004.10.001, http://dx.doi.org/10.1016/j.ic.2004.10.001.

[7] E. BEN-SASSON, O. GOLDREICH, P. HARSHA, M. SUDAN, AND S. P. VADHAN, *Robust PCPs of proximity, shorter PCPs, and applications to coding*, SIAM J. Comput., 36 (2006), pp. 889–974, http://dx.doi.org/10.1137/S0097539705446810, http://dx.doi.org/10.1137/S0097539705446810.

[8] E. Ben-Sasson, P. Harsha, and S. Raskhodnikova, *Some 3CNF properties are hard to test*, SIAM J. Comput., 35 (2005), pp. 1–21, http://dx.doi.org/10.1137/S0097539704445445, http://dx.doi.org/10.1137/S0097539704445445.

[9] P. Berman, M. Murzabulatov, and S. Raskhodnikova, *The power and limitations of uniform samples in testing properties of figures*, in 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India, 2016, pp. 45:1–45:14, http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2016.45, https://doi.org/10.4230/LIPIcs.FSTTCS.2016.45.

[10] P. Berman, M. Murzabulatov, and S. Raskhodnikova, *Testing convexity of figures under the uniform distribution*, in 32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA, 2016, pp. 17:1–17:15, http://dx.doi.org/10.4230/LIPIcs.SoCG.2016.17, https://doi.org/10.4230/LIPIcs.SoCG.2016.17.

[11] P. Berman, M. Murzabulatov, and S. Raskhodnikova, *Tolerant testers of image properties*, in 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, 2016, pp. 90:1–90:14, http://dx.doi.org/10.4230/LIPIcs.ICALP.2016.90, https://doi.org/10.4230/LIPIcs.ICALP.2016.90.

[12] A. Bhattacharyya, E. Grigorescu, K. Jung, S. Raskhodnikova, and D. P. Woodruff, *Transitive-closure spanners*, SIAM J. Comput., 41 (2012), pp. 1380–1425.

[13] E. Blais, J. Brody, and K. Matulef, *Property testing lower bounds via communication complexity*, Computational Complexity, 21 (2012), pp. 311–358, http://dx.doi.org/10.1007/s00037-012-0040-x, http://dx.doi.org/10.1007/s00037-012-0040-x.

[14] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev, *Lower bounds for testing properties of functions over hypergrid domains*, in IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014, 2014, pp. 309–320.

[15] J. Briët, S. Chakraborty, D. García-Soriano, and A. Matsliah, *Monotonicity testing and shortest-path routing on the cube*, Combinatorica, 32 (2012), pp. 35–53, http://dx.doi.org/10.1007/s00493-012-2765-1, http://dx.doi.org/10.1007/s00493-012-2765-1.

[16] C. L. Canonne, E. Grigorescu, S. Guo, A. Kumar, and K. Wimmer, *Testing k-monotonicity*, in 8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA, 2017, pp. 29:1–29:21, http://dx.doi.org/10.4230/LIPIcs.ITCS.2017.29, https://doi.org/10.4230/LIPIcs.ITCS.2017.29.

[17] D. Chakrabarty, K. Dixit, M. Jha, and C. Seshadhri, *Property testing on product distributions: Optimal testers for bounded derivative properties*, ACM Trans. Algorithms, 13 (2017), pp. 20:1–20:30, http://dx.doi.org/10.1145/3039241, http://doi.acm.org/10.1145/3039241.

[18] D. Chakrabarty and C. Seshadhri, *Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids*, in Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013, 2013, pp. 419–428, http://dx.doi.org/10.1145/2488608.2488661, http://doi.acm.org/10.1145/2488608.2488661.

[19] D. Chakrabarty and C. Seshadhri, *An optimal lower bound for monotonicity testing over hypergrids*, Theory of Computing, 10 (2014), pp. 453–464, http://dx.doi.org/10.4086/toc.2014.v010a017, http://dx.doi.org/10.4086/toc.2014.v010a017.

[20] L. Devroye, *A note on the height of binary search trees*, J. ACM, 33 (1986), pp. 489–498, http://dx.doi.org/10.1145/5925.5930, http://doi.acm.org/10.1145/5925.5930.

[21] I. Dinur and O. Reingold, *Assignment testers: Towards a combinatorial proof of the PCP theorem*, SIAM J. Comput., 36 (2006), pp. 975–1024, http://dx.doi.org/10.1137/S0097539705446962, https://doi.org/10.1137/S0097539705446962.

[22] K. Dixit, M. Jha, S. Raskhodnikova, and A. Thakurta, *Testing the Lipschitz property over product distributions with applications to data privacy*, in TCC, 2013, pp. 418–436, http://dx.doi.org/10.1007/978-3-642-36594-2_24, http://dx.doi.org/10.1007/978-3-642-36594-2_24.

[23] K. Dixit, S. Raskhodnikova, A. Thakurta, and N. M. Varma, *Erasure-resilient property testing*, in 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, 2016, pp. 91:1–91:15, http://dx.doi.org/10.4230/LIPIcs.ICALP.2016.91, https://doi.org/10.4230/LIPIcs.ICALP.2016.91.

[24] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky, *Improved testing algorithms for monotonicity*, in Randomization, Approximation, and Combinatorial Algorithms and Techniques, Third International Workshop on Randomization and Approximation Techniques in Computer Science, and Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems RANDOM-APPROX'99, Berkeley, CA, USA, August 8-11, 1999, Proceedings, 1999, pp. 97–108.

[25] M. Drmota, *An analytic approach to the height of binary search trees II*, J. ACM, 50

(2003), pp. 333–374, http://dx.doi.org/10.1145/765568.765572, http://doi.acm.org/10.1145/765568.765572.

[26] F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan, *Spot-checkers*, J. Comput. Syst. Sci., 60 (2000), pp. 717–751, http://dx.doi.org/10.1006/jcss.1999.1692, http://dx.doi.org/10.1006/jcss.1999.1692.

[27] S. Fattal and D. Ron, *Approximating the distance to convexity*. Unpublished manuscript. Uploaded at http://www.eng.tau.ac.il/ danar/Public-pdf/app-conv.pdf.

[28] S. Fattal and D. Ron, *Approximating the distance to monotonicity in high dimensions*, ACM Transactions on Algorithms, 6 (2010), http://dx.doi.org/10.1145/1798596.1798605, http://doi.acm.org/10.1145/1798596.1798605.

[29] E. Fischer, *On the strength of comparisons in property testing*, Inform. and Comput., 189 (2004), pp. 107–116.

[30] E. Fischer and L. Fortnow, *Tolerant versus intolerant testing for Boolean properties*, Theory of Computing, 2 (2006), pp. 173–183, http://dx.doi.org/10.4086/toc.2006.v002a009, http://dx.doi.org/10.4086/toc.2006.v002a009.

[31] E. Fischer, O. Lachish, and Y. Vasudev, *Trading query complexity for sample-based testing and multi-testing scalability*, in IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015, 2015, pp. 1163–1182, http://dx.doi.org/10.1109/FOCS.2015.75, https://doi.org/10.1109/FOCS.2015.75.

[32] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky, *Monotonicity testing over general poset domains*, in Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, STOC '02, New York, NY, USA, 2002, ACM, pp. 474–483, http://dx.doi.org/10.1145/509907.509977, http://doi.acm.org/10.1145/509907.509977.

[33] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky, *Testing monotonicity*, Combinatorica, 20 (2000), pp. 301–337.

[34] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750, http://dx.doi.org/10.1145/285055.285060, http://doi.acm.org/10.1145/285055.285060.

[35] O. Goldreich and T. Kaufman, *Proximity oblivious testing and the role of invariances*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings, 2011, pp. 579–592, http://dx.doi.org/10.1007/978-3-642-22935-0_49, http://dx.doi.org/10.1007/978-3-642-22935-0_49.

[36] O. Goldreich and D. Ron, *On proximity-oblivious testing*, SIAM J. Comput., 40 (2011), pp. 534–566, http://dx.doi.org/10.1137/100789646, http://dx.doi.org/10.1137/100789646.

[37] O. Goldreich and D. Ron, *On sample-based testers*, ACM Trans. Comput. Theory, 8 (2016), pp. 7:1–7:54, http://dx.doi.org/10.1145/2898355, http://doi.acm.org/10.1145/2898355.

[38] O. Goldreich and I. Shinkar, *Two-sided error proximity oblivious testing*, Random Struct. Algorithms, 48 (2016), pp. 341–383, http://dx.doi.org/10.1002/rsa.20582, http://dx.doi.org/10.1002/rsa.20582.

[39] S. Halevy and E. Kushilevitz, *Testing monotonicity over graph products*, Random Struct. Algorithms, 33 (2008), pp. 44–67, http://dx.doi.org/10.1002/rsa.20211, http://dx.doi.org/10.1002/rsa.20211.

[40] M. Jha and S. Raskhodnikova, *Testing and reconstruction of Lipschitz functions with applications to data privacy*, SIAM J. Comput., 42 (2013), pp. 700–731, http://dx.doi.org/10.1137/110840741, http://dx.doi.org/10.1137/110840741.

[41] M. J. Kearns and D. Ron, *Testing problems with sublearning sample complexity*, J. Comput. Syst. Sci., 61 (2000), pp. 428–456, http://dx.doi.org/10.1006/jcss.1999.1656, http://dx.doi.org/10.1006/jcss.1999.1656.

[42] E. Lehman and D. Ron, *On disjoint chains of subsets*, J. Combin. Theory Ser. A, 94 (2001), pp. 399–404.

[43] M. Parnas, D. Ron, and R. Rubinfeld, *On testing convexity and submodularity*, SIAM J. Comput., 32 (2003), pp. 1158–1184, http://dx.doi.org/10.1137/S0097539702414026, http://dx.doi.org/10.1137/S0097539702414026.

[44] M. Parnas, D. Ron, and R. Rubinfeld, *Tolerant property testing and distance approximation*, J. Comput. System Sci., 6 (2006), pp. 1012–1042.

[45] B. Pittel, *On growing random binary trees*, Journal of Mathematical Analysis and Applications, 103 (1984), pp. 461 – 480, http://dx.doi.org/http://dx.doi.org/10.1016/0022-247X(84)90141-0, http://www.sciencedirect.com/science/article/pii/0022247X84901410.

[46] B. A. Reed, *The height of a random binary search tree*, J. ACM, 50 (2003), pp. 306–332, http://dx.doi.org/10.1145/765568.765571, http://doi.acm.org/10.1145/765568.765571.

[47] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271, http://dx.doi.org/10.1137/S0097539793255151, http://dx.doi.org/10.1137/S0097539793255151.

[48] M. E. Saks and C. Seshadhri, *Estimating the longest increasing sequence in polylogarithmic time*, SIAM J. Comput., 46 (2017), pp. 774–823, http://dx.doi.org/10.1137/130942152, https://doi.org/10.1137/130942152.