A Brief Introduction to Property Testing

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

April 30, 2010

Abstract

This short article provides a brief description of the main issues that underly the study of property testing. It is meant to serve as a general introduction to a collection of surveys and extended abstracts that cover various specific subareas and research directions in property testing.

1 Introduction

Property Testing is the study of super-fast (randomized) algorithms for approximate decision making. These algorithms are given direct access to items of a huge data set, and determine whether this data set has some predetermined (global) property or is far from having this property. Remarkably, this approximate decision is made by accessing a small portion of the data set.

Property Testing has been a subject of intensive research in the last couple of decades, with hundreds of studies conducted in it and in closely related areas. Indeed, Property Testing is closely related to Probabilistically Checkable Proofs (PCPs), and is related to Coding Theory, Combinatorics, Statistics, Computational Learning Theory, Computational Geometry, and more.

This brief introduction to the area of Property Testing is confined to conceptual issues; that is, it focuses on the main notions and models being studied, while hardly mentioning the numerous results obtained in the various models. This deficiency of the current article is corrected by the various surveys and extended abstracts presented in the current volume. In addition, we refer the interested reader to two recent surveys of Ron [9, 10].

2 The Issues

Property testing is a relaxation of decision problems and it focuses on algorithms that can only read parts of the input. Thus, the input is represented as a function (to which the tester has oracle access) and the tester is required to accept functions that have some predetermined property (i.e., reside in some predetermined set) and reject any function that is "far" from the set of functions having the property. Distances between functions are defined as the fraction of the domain on which the functions disagree, and the threshold determining what is considered far is presented as a proximity parameter, which is explicitly given to the tester.

An asymptotic analysis is enabled by considering an infinite sequence of domains, functions, and properties. That is, for any n, we consider functions from D_n to R_n, where $|D_n| = n$. (Often, one just assumes that $D_n = [n] \stackrel{\rm def}{=} \{1, 2, ..., n\}$.) Thus, in addition to the input oracle, representing a function $f : D_n \to R_n$, the tester is explicitly given two parameters: a size parameter, denoted n, and a proximity parameter, denoted ϵ.

Definition 1 *Let $\Pi = \bigcup_{n \in \mathsf{N}} \Pi_n$, where Π_n contains functions defined over the domain D_n. A* tester for a property Π *is a probabilistic oracle machine T that satisfies the following two conditions:*

1. *The tester accepts each $f \in \Pi$ with probability at least $2/3$; that is, for every $n \in \mathsf{N}$ and $f \in \Pi_n$* (and every $\epsilon > 0$)*, it holds that $\Pr[T^f(n,\epsilon) = 1] \geq 2/3$.*
2. *Given $\epsilon > 0$ and oracle access to any f that is ϵ-far from Π, the tester rejects with probability at least $2/3$; that is, for every $\epsilon > 0$ and $n \in \mathsf{N}$, if $f : D_n \to R_n$ is ϵ-far from Π_n, then $\Pr[T^f(n,\epsilon) = 0] \geq 2/3$, where f is* ϵ-far from Π_n *if, for every $g \in \Pi_n$, it holds that $|\{e \in D_n : f(e) \neq g(e)\}| > \epsilon \cdot n$.*

If the tester accepts every function in Π with probability 1, then we say that it has one-sided error*; that is, T has one-sided error if for every $f \in \Pi$ and every $\epsilon > 0$, it holds that $\Pr[T^f(n,\epsilon) = 1] = 1$. A tester is called* non-adaptive *if it determines all its queries based solely on its internal coin tosses* (and the parameters n and ϵ)*; otherwise it is called* adaptive.

Definition 1 does not specify the query complexity of the tester, and indeed an oracle machine that queries the entire domain of the function qualifies as a tester (with zero error probability...). Needless to say, we are interested in testers that have significantly lower query complexity.

Research in property testing is often categorized according to the type of functions and properties being considered. In particular, algebraic property testing focuses on the case that the domain and range are associated with some algebraic structures (e.g., groups, fields, and vector spaces) and studies algebraic properties such as being a polynomial of low degree (see, e.g., [3, 11]). In the context of testing graph properties (see, e.g., [4]), the functions represent graphs or rather allow certain queries to such graphs (e.g., in the adjacency matrix model, graphs are represented by their adjacency relation and queries correspond to pairs of vertices where the answers indicate whether or not the two vertices are adjacent in the graph).[1]

Ramifications. While most research in property testing refers to distances with respect to the uniform distribution on the function's domain, other distributions and even distribution-free models were also considered. That is, for a (known or unknown) distribution μ on the domain, we say that f is ϵ-far from g (w.r.t μ) if $\Pr_{e \sim \mu}[f(e) \neq g(e)] > \epsilon$. Indeed, Definition 1 refers to the case that μ is uniform over the domain (i.e., D_n).

A somewhat related model is one in which the tester obtains random pairs $(e, f(e))$, where each sample e is drawn (independently) from the aforementioned distribution. Such random (f-labeled) example can be either obtained on top of the queries to f or instead of them. This is also the context of testing distributions, where the examples are actually unlabeled and the aim is testing properties of the underlying distribution (rather than properties of the labeling which is null here).

A third ramification refers to the related notions of *tolerant testing* and *distance approximation* (cf. [8]). In the latter, the algorithm is required to estimate the distance of the input (i.e., f) from the predetermined set of instances having the property (i.e., Π). Tolerant testing usually means only a crude distance approximation that guarantees that inputs close to Π (rather than only inputs in Π) are accepted while inputs that are far from Π are rejected (as usual).

[1] In an alternative model, known as the incidence-list model, graphs are represented by functions that assign to the pair (v, i) the ith neighbor of vertex v.

On the current focus on query complexity. Current research in property testing focuses mainly on query (and/or sample) complexity, while either ignoring time complexity or considering it a secondary issue. The current focus on these information theoretic measures is justified by the fact that even the latter are far from being understood. (Indeed, this stands in contrast to the situation in, say, PAC learning.)

On the importance of representation. The representation of problems' instances is crucial to any study of computation, since the representation determines the type of information that is explicit in the input. This issue becomes much more acute when one is only allowed partial access to the input (i.e., making a number of queries that result in answers that do not fully determine the input). An additional issue, which is unique to property testing, is that the representation may effect the distance measure (i.e., the definition of distances between inputs). This is crucial because property testing problems are defined in terms of this distance measure.

The importance of representation is forcefully demonstrated in the gap between the complexity of testing numerous natural graph properties in two natural representations: the adjacency matrix representation (cf. [4]) and the incidence lists representation (cf. [5]).

Things get to the extreme in the study of locally testable codes, which may be viewed as evolving around testing whether the input is "well formed" with respect to some fixed error correcting code. Interestingly, the general study of locally testable codes seeks an arbitrary succinct representation (i.e., a code of good rate) such that well-formed inputs (i.e., codewords) are far apart and testing well-formness is easy (i.e., there exists a low complexity codeword test).

3 A Brief Historical Perspective

Property testing first appeared as a tool towards program checking (see the linearity tester of [3]) and the construction of PCPs (see the low-degree tests and their relation to locally testable codes, as discussed in [11]). In these settings it was natural to view the tested object as a function, and this convention continued also in [4], which defined property testing in relation to PAC learning. More importantly, in [4] property testing is promoted as a new type of computational problems, which transcends all its natural applications.

While [3, 11] focused on algebraic properties, the focus of [4] was on graph properties. From this perspective the choice of representation became less obvious, and oracle access was viewed as allowing local inspection of the graph rather than being the graph itself.[2] The distinction between objects and their representations became more clear when an alternative representation of graphs was studied in [5, 6]. At this point, query complexity that is polynomially related to the size of the object (e.g., its square root) was no longer considered inhibiting. This shift in scale is discussed next.

Recall that initially property testing was viewed as referring to functions that are implicitly defined by some succinct programs (as in the context of program checking) or by "transcendental" entities (as in the context of PAC learning). From this perspective the yardstick for efficiency is being polynomial in the length of the query, which means being polylogarithmic in the size of the object. However, when viewing property testing as being applied to (huge) objects that may exist in explicit form in reality, it is evident that any sub-linear complexity may be beneficial.

[2]That is, in this case the starting point is the (unlabeled) graph itself, and its representation as a (labeled) graph by either its adjacency matrix or incidence list is an auxiliary conceptual step.

The realization that property testing may mean any algorithm that does not inspect its entire input seems crucial to the study of testing distributions, which emerged with [2]. In general, property testing became identified as a study of a special type of sublinear-time algorithms.

Another consequence of the aforementioned shift in scale is the decoupling of the representation from the query types. In the context of graph properties, this culminated in the model of [7].

Nevertheless, the study of testing properties within query complexity that only depends on the proximity parameter (and is thus totally independent of the size of the object) remains an appealing and natural direction. A remarkable result in this direction is the characterization of graph properties that are testable within such complexity in the adjacency matrix model [1].

References

[1] N. Alon, E. Fischer, I. Newman, and A. Shapira. A Combinatorial Characterization of the Testable Graph Properties: It's All About Regularity. In *38th STOC*, pages 251–260, 2006.

[2] T. Batu, L. Fortnow, R. Rubinfeld, W.D. Smith and P. White. Testing that Distributions are Close. In *41st FOCS*, pages 259–269, 2000.

[3] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *JCSS*, Vol. 47, No. 3, pages 549–595, 1993. Extended abstract in *22nd STOC*, 1990.

[4] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

[5] O. Goldreich and D. Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, Vol. 32 (2), pages 302–343, 2002. Extended abstract in *29th STOC*, 1997.

[6] O. Goldreich and D. Ron. A Sublinear Bipartitness Tester for Bounded Degree Graphs. *Combinatorica*, Vol. 19 (3), pages 335–373, 1999. Extended abstract in *30th STOC*, 1998.

[7] T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. In *Proc. of RANDOM'03*, pages 341–353, 2003.

[8] M. Parnas, D. Ron, and R. Rubinfeld: Tolerant Property Testing and Distance Approximation. *JCSS*, Vol. 72 (6), pages 1012–1042, 2006. Preliminary version in ECCC, 2004.

[9] D. Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning*, Vol. 1 (3), pages 307–402, 2008.

[10] D. Ron. Algorithmic and Analysis Techniques in Property Testing. *Foundations and Trends in TCS*, to appear.

[11] R. Rubinfeld and M. Sudan. Robust Characterization of Polynomials with Applications to Program Testing. *SIAM Journal on Computing*, Vol. 25 (2), pages 252–271, 1996.