Recent Progress in the Classification for Testability

Charles Jordan and Thomas Zeugmann

Division of Computer Science, Hokkaido University

June 22, 2011

Property Testing

Induction

Imagine we have a huge structure (e.g., a graph or database) and we randomly take a *small* sample to examine.

What can we say about the *entire* structure?

This problem is fundamental to statistics, machine learning and property testing.

Essentially fast, randomized approximation of model-checking.

Basic Theme

Classification Problem for Testability

A *prefix vocabulary class* of first-order logic is defined by a pattern of quantifiers and the predicate symbols allowed.

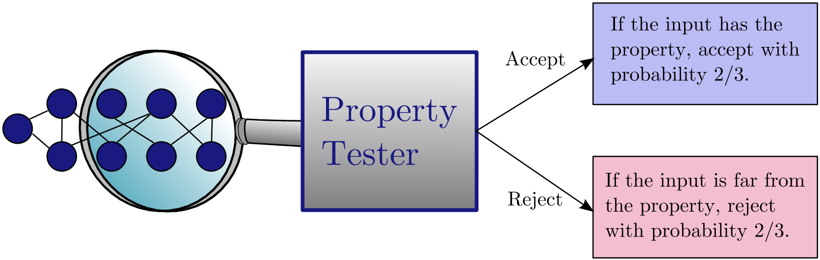
Classify *all* such classes as testable or untestable.

- This topic began with Alon *et al.* Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.
- We want a classification like that for decidability, the finite model property, etc.
- We generalize from graphs to relational structures.

A Property Tester

Property testers are **probabilistic approximation algorithms** that make queries for input bits.

- We can't distinguish between a large structure that has our property and one that *almost* has it.
- Maybe we only want to distinguish if the input has the property or is *far* from having the property.



Motivation I

Theorem (Alon *et al.* (2000))

All graph properties expressible with quantifiers $\exists^*\forall^*$ are testable and there is an untestable property expressible with quantifiers $\forall^*\exists^*$.

- The untestable property of Alon *et al.* has prefix $\forall^{12}\exists^{5}$.
- Is this optimal? Where is the border for testability?

Motivating Questions

What is the minimum number of first-order

- universal quantifiers,
- existential quantifiers,
- (total) quantifiers

needed to express an untestable property (in any vocabulary)?

Motivation II

Corollaries of Recent Results and Open Question

The minimum numbers are

- two universal quantifiers,
- one existential quantifier,
- three total quantifiers.

For these minima, the vocabulary of *directed* graphs suffices but vocabularies of strings do not.

Ultimate Goal: Classification for Testability

We'd like a *complete* classification of the testable and untestable prefix-vocabulary classes. Such a finite classification exists.

Current Classification for Testability

Testable Classes

The following are testable.

$[\exists^*\forall^*, all]_=$	Alon *et al.* (2000), Jordan, Zeugmann (2010b)
Monadic	Alon *et al.* (2001), McNaughton, Papert (1971)
$[\exists^*\forall\exists^*, all]_=$	Jordan, Zeugmann (2009)

Untestable Classes

The following contain untestable graph properties.

$[\forall^3\exists, (0,1)]_=$	Alon *et al.* (2000), Jordan, Zeugmann (2010a)
$[\forall\exists\forall, (0,1)]_=$	Jordan, Zeugmann (2011)

Consistent with classifications for finite model property, associated SO 0-1 laws, finite satisfiability, etc.

Open Problems (Future Work)

1. It would be very interesting to know the testability of graph properties expressible with quantifier prefix $\forall^2\exists$ (more generally relational properties and $\exists^*\forall^2\exists^*$). This may suffice to *complete* the classification for testabilility and predicate logic with equality.
2. For the known testable cases, it's possible to automatically generate testers from the syntax of queries - possible applications to databases, differential privacy, etc.?
3. It would be nice to know if equality is *necessary* to express untestable properties in our classes.
4. Testability of other prefix vocabulary fragments.

Introductions and Surveys

1. Oded Goldreich. Introduction to Testing Graph Properties. ECCC TR10-082, 2010.
2. Dana Ron. Algorithmic and Analysis Techniques in Property Testing. Found. Trends Theor. Comput. Sci., 5(2):73–205, 2009.
3. Dana Ron. Property Testing: A Learning Theory Perspective. Found. Trends Mach. Learn., 1(3):307–402, 2008.
4. Eldar Fischer. The Art of Uninformed Decisions. Bulletin of EATCS, 75:97–126, 2001.

History I

1. Randomization and approximation for efficiency
    - de Leeuw *et al.* Computability by probabilistic machines. *Automata Studies*, 183–212, 1956.
    - Freivalds, Fast probabilistic algorithms. *Proc. MFCS 1979*, LNCS 74, 57–69, 1979.
2. Property testing began in *formal verification*.
    - Quickly check that a program is “probably approximately correct” before doing expensive verification.
    - Rubinfeld and Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1995.
    - Blum, *et al.* Self-testing/correcting with applications to numerical problems. *J. of Comput. Syst. Sci.*, 47(3):549–595, 1993.
3. Extension to graphs using a functional representation of adjacency matrices.
    - Goldreich, *et al.* Property testing and its connection to learning and approximation. *J. ACM*, 45(4)653–750, 1998.

History II

4. The classification for testability started in *graph* testing.
   - Alon *et al.* Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.
5. Uniform hypergraph testing and regularity/removal lemmas
   - Rödl and Schacht. Generalizations of the removal lemma. *Combinatorica*, 29(4):467–501, 2009.
6. Recent models for non-uniform hypergraph[1] testing
   - Fischer, *et al.* Approximate hypergraph partitioning and applications. *Proc. FOCS 2007*, pp. 579–589, 2007.
   - Austin and Tao. On the testability and repair of hereditary hypergraph properties. *Random Struc. Alg.*, 36(4):373–463, 2010.
   - Jordan and Zeugmann, 2009.

[1] Equivalent to *relational structures* in logic.

Relational Structures I

A *binary string* is a sequence over $\{0,1\}$, e.g.,

$$w = 0_0 1_1 1_2 0_3 1_4.$$

This can be represented by a set of *bit positions* U and a monadic predicate $\mathcal{S} \subseteq U$ denoting the positions that are 1. So, our previous example can be written as

$$w = \{\{0,1,\ldots,4\},\{1,2,4\}\}.$$

Graphs can be represented by a set U of vertices and a binary predicate $\mathcal{E} \subseteq U^2$ for the edge set.

Generalization

How about a generalization?

Relational Structures II

Definition: Vocabulary

A *vocabulary* is a set of predicate symbols with their arities,

$$\tau := \{R_1^{a_1}, \ldots, R_s^{a_s}\}.$$

Examples: $\{S^1\}$ for binary strings and $\{E^2\}$ for graphs.

Definition: Structures

A *structure* of type τ is an $(s+1)$-tuple, $A := (U, \mathcal{R}_1, \ldots, \mathcal{R}_s)$, where U is a finite set and $\mathcal{R}_i \subseteq U^{a_i}$ a predicate for $R_i \in \tau$.

- Our first-order logic is a predicate logic with equality.

Distance Measures

A and B are structures of type τ with size n, and $\oplus$ is x-or.

1. The *fraction of tuples with different assignments*,

Definition: $\mathrm{dist}(A, B)$

$$\mathrm{dist}(A,B) := \frac{\sum_{1 \leqslant i \leqslant s} |\{\mathbf{x} \mid \mathbf{x} \in U^{a_i} \text{ and } \mathcal{R}_i^A(\mathbf{x}) \oplus \mathcal{R}_i^B(\mathbf{x})\}|}{\sum_{i=1}^{s} n^{a_i}}.$$

Distance Measures

A and B are structures of type τ with size n, and $\oplus$ is x-or.

1. The *fraction of tuples with different assignments*,

Definition: $\mathrm{dist}(A, B)$

$$\mathrm{dist}(A,B) := \frac{\sum_{1\leqslant i\leqslant s} |\{\mathbf{x} \mid \mathbf{x} \in U^{a_i} \text{ and } \mathcal{R}_i^A(\mathbf{x}) \oplus \mathcal{R}_i^B(\mathbf{x})\}|}{\sum_{i=1}^{s} n^{a_i}}.$$

2. But then, high-arity relations have more weight. If all relations are equal,

Definition: $\mathrm{rdist}(A, B)$

$$\mathrm{rdist}(A,B) := \max_{1\leqslant i\leqslant s} \frac{|\{\mathbf{x} \mid \mathbf{x} \in U^{a_i} \text{ and } R_i^A(\mathbf{x}) \oplus R_i^B(\mathbf{x})\}|}{n^{a_i}}.$$

3. $\mathrm{mrdist}(A, B)$ is similar but tuples like (a, a, b) are treated as separate low-arity relations (*subrelations*).

Testing Definitions

The distance measures extend to properties in the usual way,

$$\operatorname{dist}(A, P) := \min_{B \in P} \operatorname{dist}(A, B) .$$

ε-tester

An *ε-tester for P* is a randomized algorithm that makes queries for tuples in relations, accepts structures that have P with probability 2/3 and rejects those that are ε-far from P with probability 2/3.

Testability

A property P is *testable* if there exists a function $c(\varepsilon)$ and, for every $\varepsilon > 0$, an ε-tester with query complexity at most $c(\varepsilon)$.

Quick Outline: Classification Problem for Testability

We classify formulas in prenex normal form, based on

1. The pattern of quantifiers;
2. The *vocabulary* (number and arity of predicate symbols).

Each *prefix vocabulary class* has a representation, for example

$$[\exists^*, (0,1)]_=$$

is the set of formulas with only existential quantifiers, one binary predicate symbol that may contain the '=' symbol. These are the first-order existential graph properties.

The Classification Problem for Testability

We want to classify *all* such classes as testable or untestable.

Classification Formalities

Definition: Prefix-Vocabulary Fragment

A *prefix-vocabulary fragment* is a triple, $[\Pi, p]_e$, where

1. Π is a string over $\{\exists, \forall, \exists^*, \forall^*\}$,
2. p is a sequence over $\mathbb{N}$ and ω or the phrase *all*, and
3. e is $=$ or the empty string.

Definition: Prefix-Vocabulary Class

Fragment $[\Pi, p = (p_1, p_2, \ldots)]_e$ defines the class of sentences in prenex normal form satisfying the following.

1. The quantifiers match the language specified by Π when interpreted as a regular expression.
2. If p is not *all*, at most p_i-many distinct predicate symbols of arity i appear.
3. If equality $(=)$ appears in the sentence, then e is $=$.

Result I: Ackermann's Class is Testable

(Informal) Definition: Ackermann's Class with Equality

Ackermann's class with equality is $[\exists^*\forall\exists^*, all]_=$, the set of (first-order) sentences of pure predicate logic with equality that have *one* $\forall$ (and any number of $\exists$). No function symbols, ordering or arithmetic is present.

Theorem: Ackermann's Class with Equality is Testable

All formulas in Ackermann's class with equality are testable under all our definitions.

- This class was first studied by Ackermann (1928).
- The proof uses model-theoretic properties of this class.

Result II: Ramsey's Class is Testable

Definition: Ramsey's Class

Ramsey's class is $[\exists^*\forall^*, all]_=$, the set of (first-order) sentences of pure predicate logic with equality where all existential quantifiers precede all universal quantifiers. No function symbols, ordering or arithmetic is present.

Alon *et al.* (2000) showed that the restriction to undirected, loop-free graphs is testable.

Theorem: Ramsey's Class is Testable

All formulas in Ramsey's class are testable under all our definitions.

- This class was first studied by Ramsey (1930).
- The proof applies a strong result by Austin and Tao (2010).

Result III: Untestable Prefix-Vocabulary Classes

- We simplify the untestable property (prefix $\forall^{12}\exists^5$) of Alon *et al.* (2000) to obtain the following.

Theorem

There is an untestable (in all our models) graph property expressible with each of the following quantifier prefixes.

1. $\forall\exists\forall\exists$
2. $\forall\exists\forall^2$
3. $\forall^2\exists\forall$
4. $\forall^{\mathbf{3}}\exists$

- The proof uses an untestable variant of checking explicit bipartite graph isomorphism, expressible in these classes.
- Very recently, we've improved on three of these classes, but we still need the result for $\forall^{\mathbf{3}}\exists$.

Result IV: Kahr-Moore-Wang Class is Untestable

- Very recently, we've used a property more closely related to *function* isomorphism to prove the following.

Theorem: Minimal Kahr-Moore-Wang Class is Untestable

There is an untestable[a] (in all of our models) graph property expressible in the class $[\forall\exists\forall, (0,1)]_=$.

[a] Even with $o(\sqrt{n})$ queries.

- This class was first studied by Kahr, Moore, Wang (1962).

Result IV: Kahr-Moore-Wang Class is Untestable

- Very recently, we've used a property more closely related to *function* isomorphism to prove the following.

Theorem: Minimal Kahr-Moore-Wang Class is Untestable

There is an untestable[a] (in all of our models) graph property expressible in the class $[\forall\exists\forall, (0,1)]_=$.

[a] Even with $o(\sqrt{n})$ queries.

- This class was first studied by Kahr, Moore, Wang (1962).

An Observation

The only remaining classes *with* equality contain $\forall^2\exists$. Determining the testability for these classes would *complete* the classification for predicate logic with equality.

Other Classifications

Definition: Some Properties of Prefix-Vocabulary Classes

Let C be a class of sentences of first-order logic. We say that

- *satisfiability problem* is decidable if given any $\varphi \in C$, we can decide whether there is a model (possibly infinite) of φ.
- The *complexity* of satisfiability is the complexity of this problem, given φ as input (usually very hard)
- C has the *finite model property* if, for every $\varphi \in C$ it is true that: if φ has a model then it has a finite model.
- C has *infinity axioms* if it contains formulas that have only infinite models.
- The associated fragment of second-order existential logic (SO∃) has a *0-1 law* if for all $\varphi \in C$ and all SO∃ sentences $\gamma := \exists S_1 \dots S_a \varphi$, the limit as $n \to \infty$ of the probability that a random structure of size n models γ exists and is 0 or 1.

Complete Classification for Decidability

Undecidable	Undecidable	Decidable
$[\forall\exists\forall, (\omega,1), (0)]$	$[\forall^3\exists, (\omega,1), (0)]$	$[\exists^*\forall^*, all, (0)]_=$
$[\forall^*\exists, (0,1), (0)]$	$[\forall\exists\forall^*, (0,1), (0)]$	$[\exists^*\forall^2\exists^*, all, (0)]$
$[\forall\exists\forall\exists^*, (0,1), (0)]$	$[\forall^3\exists^*, (0,1), (0)]$	$[all, (\omega), (\omega)]$
$[\forall\exists^*\forall, (0,1), (0)]$	$[\exists^*\forall\exists\forall, (0,1), (0)]$	$[\exists^*\forall\exists^*, all, all]$
$[\exists^*\forall^3\exists, (0,1), (0)]$	$[\forall, (0), (2)]_=$	$[\exists^*, all, all]_=$
$[\forall, (0), (0,1)]_=$	$[\forall^2, (0,1), (1)]$	$[all, (\omega), (1)]_=$
$[\forall^2, (1), (0,1)]$	$[\forall^2\exists, (\omega,1), (0)]_=$	$[\exists^*\forall\exists^*, all, (1)]_=$
$[\exists^*\forall^2\exists, (0,1), (0)]_=$	$[\forall^2\exists^*, (0,1), (0)]_=$	

1915-1984

Classification for the Finite Model Property

Predicate Logic with Equality

The following fragments allow predicate symbols and (possibly) equality, but do not allow function symbols.

Finite Model Property	Infinity Axioms
$[\exists^*\forall^*, all]_=$	$[\forall^3\exists, (0,1)]$
$[\exists^*\forall\exists^*, all]_=$	$[\forall\exists\forall, (0,1)]$
$[all, (\omega)]_=$	$[\forall^2\exists, (\omega,1)]_=$
$[\exists^*\forall^2\exists^*, all]$	$[\forall^2\exists^*, (0,1)]_=$
	$[\exists^*\forall^2\exists, (0,1)]_=$

A starting point for the testing classification?

- $\forall\exists\forall$, $\forall^3\exists$ and $\exists^*\forall^2\exists^*$ look interesting.

Classification for Associated 0-1 Laws

Holds w/ $=$	Fails w/ $=$
$[\exists^*\forall^*, all]_=$	$[\forall\exists\forall, (0,1)]_=$
$[\exists^*\forall\exists^*, all]_=$	$[\forall^3\exists, (0,1)]_=$
	$[\forall^2\exists, (0,1)]_=$

With equality (1987-1998), the same classification as *docility*.

Classification without Equality

The classification for 0-1 laws is the same as above, but the classification for *docility* is different.

- $[\exists^*\forall^2\exists^*, all]$ (the Gödel class) is decidable and docile but the associated 0-1 law fails.

Classification for Associated 0-1 Laws

Holds w/ $=$	Fails w/ $=$
$[\exists^*\forall^*, all]_=$	$[\forall\exists\forall, (0,1)]_=$
$[\exists^*\forall\exists^*, all]_=$	$[\forall^3\exists, (0,1)]_=$
	$[\forall^2\exists, (0,1)]_=$

With equality (1987-1998), the same classification as *docility*.

Classification without Equality

The classification for 0-1 laws is the same as above, but the classification for *docility* is different.

- $[\exists^*\forall^2\exists^*, all]$ (the Gödel class) is decidable and docile but the associated 0-1 law fails.

- Well-behavedness of $\exists^*\forall^2\exists^*$ (the Gödel class) is *fragile*.
- Interesting: $\forall\exists\forall$, $\forall^3\exists$, $\forall^2\exists$ and maybe $\exists^*\forall^2\exists^*$

Kolaitis and Vardi. 0-1 laws for fragments of existential second-order logic: a survey. Proc. MFCS 2000, LNCS 1893, pp. 84–98 (2000)

Constructive Testability

Constructive Testability

All current classification results are *constructive*.
Given a formula from these classes and ε, we can construct an ε-tester from the formula's *syntax*.

Possible(?) applications include:

1. Automatic generation of testers for model checkers/etc
2. Automatic generation of testers for some SQL queries
    - See Libkin. Expressive power of SQL. *Proc. ICDT 2001*, LNCS 1973 (2001) for connections between SQL and first-order logic.

Question: Which Classification?

Is the classification for *constructive* testability most important?

Thankfully the classifications all coincide so far
(mrdist may be preferable for technical reasons).