Baltic J. Modern Computing, Vol. 4 (2016), No. 4, pp. 736–752
DOI: http://dx.doi.org/10.22364/bjmc.2016.4.4.11

The Kahr–Moore–Wang Class Contains Untestable Properties

Charles JORDAN* and Thomas ZEUGMANN**

Division of Computer Science, Hokkaido University, N-14, W-9, Sapporo, Japan

skip@ist.hokudai.ac.jp, thomas@ist.hokudai.ac.jp

Abstract. Property testing is a kind of randomized approximation in which one takes a small, random sample of a structure and wishes to determine whether the structure satisfies some property or is far from satisfying the property. We focus on the testability of classes of first-order expressible properties, and in particular, on the classification of prefix-vocabulary classes for testability. The main result is the untestability of $[\forall\exists\forall, (0,1)]_{=}$. We also show that this class remains untestable without equality in at least one model of testing. These classes are well-known and (at least one is) minimal for untestability. We discuss what is currently known about the classification for testability and briefly compare it to other classifications.

Keywords: property testing, logic, randomized algorithms, Kahr–Moore–Wang class

1 Introduction

Testing a property can be viewed as a form of approximation where we trade accuracy for efficiency. As far as we are aware, de Leeuw *et al.* (1956) first formalized probabilistic machines. They showed that such machines cannot compute uncomputable properties under reasonable assumptions, but mention the possibility that probabilistic machines could perhaps be more *efficient* than deterministic machines. This topic attracted considerable attention including Gill (1977). Early examples of such results were presented by Freivalds (1977), (1979) including his matrix multiplication checker.

In property testing, we take a random sample of some large structure and wish to distinguish between the case that it has some desired property and the case

* Supported by JSPS Grant Nos. 15H00847, 'Exploring the Limits of Computation' (ELC), and 16H02785.

** Supported by MEXT Grant-in-Aid for Scientific Research on Priority Areas under Grant No. 21013001.

that it is far from having the property. We focus on the testability of first-order expressible properties, and in particular on the classification of prefix-vocabulary classes of first-order logic for testability.

Rubinfeld and Sudan (1996) and Blum *et al.* (1993) introduced the notion of property testing in the context of formal verification. The basic idea was soon extended by Goldreich *et al.* (1998), who represented graphs as binary functions and focused on testing graph properties. We omit a detailed history of testing, see the introduction to testing graph properties by Goldreich (2010), two surveys by Ron (2008), (2009), and older surveys by Fischer (2001) and Ron (2001).

We are particularly interested in the testability of properties expressible in subclasses of first-order logic, and review relevant work in Subsection 1.1.

Here, we show that there exist untestable graph properties expressible with quantifier prefix $\forall\exists\forall$ when equality is allowed (see Section 3 for a formal statement). In addition, we use a variation of that proof to show that this prefix remains untestable in at least one of our models even when equality is forbidden. We suspect that the class is untestable without equality in all of our models.

Taking into account the related work described in Subsection 1.1 and using the notation of Definition 7, the current classification for testability is the following (cf. Jordan and Zeugmann (2012)). We omit the result for $[\forall\exists\forall, (0,1)]$ because it may depend on the choice of model.

- Testable classes
  1. Monadic first-order logic: $[all, (\omega)]_=$.
  2. Ackermann's class with equality: $[\exists^*\forall\exists^*, all]_=$.
  3. Ramsey's class: $[\exists^*\forall^*, all]_=$.
- Untestable classes
  1. $[\forall^3\exists, (0,1)]_=$.
  2. $[\forall\exists\forall, (0,1)]_=$.

We are especially interested in determining the testability of variants of the Gödel class (i.e., classes whose prefix contains at least $\forall^2\exists$) as this would suffice to complete the classification for the special case of predicate logic with equality. The above classification is consistent with several other well-known classifications, such as that for the finite model property (see, e.g., Chapter 6 of Börger *et al.* (1997), for docility (or finite satisfiability, see Kolaitis and Vardi (2000)) and for associated 0-1 laws for fragments of existential second-order logic (see Kolaitis and Vardi (2000)). It would be interesting to know if the classification for testability coincides with one of these classifications.

The rest of the paper is organized as follows: First, we review related work in Subsection 1.1. Definitions and notation are in Section 2. The proof of untestability for $[\forall\exists\forall, (0,1)]_=$ is presented in Section 3, while the case without equality is considered in Section 4.

1.1 Related Work

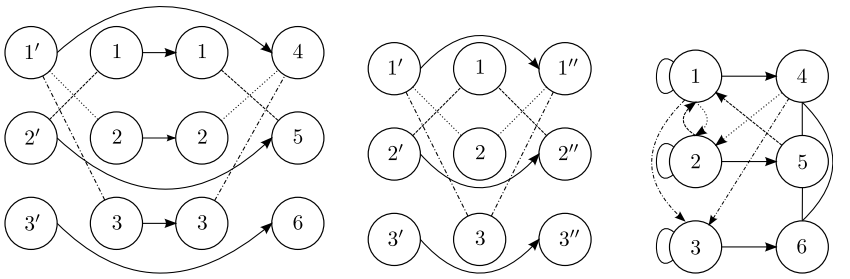
Alon *et al.* (2001) proved that the regular languages are testable, implying that monadic first-order is testable given the well-known results of Büchi (1960) or

McNaughton and Papert (1971). Alon *et al.* (2000) were the first to directly consider the classification problem for testability, restricted to properties of undirected, loop-free graphs. They showed the testability of all such properties expressible by prenex sentences with quantifier prefix $\exists^*\forall^*$, and also proved that there exists an untestable property expressible with quantifier prefix $\forall^*\exists^*$ (examining the proof shows that $\forall^{12}\exists^5$ suffices).

These are (restrictions of) well-known classes. Jordan and Zeugmann (2012) extended the positive result to the full Ramsey's class $([\exists^*\forall^*, all]_=)$, proved the testability of Ackermann's class with equality $([\exists^*\forall\exists^*, all]_=)$, and sharpened the negative result to prefixes $\forall^3\exists$, $\forall^2\exists\forall$, $\forall\exists\forall\exists$ and $\forall\exists\forall^2$ (on directed graphs with equality). This paper sharpens these last three prefixes and proves that $\forall\exists\forall$ is a minimal prefix class for untestability. This particular class is the restriction of the Kahr–Moore–Wang (1962) class to directed graphs.

It is easy to show that $[\forall\exists\forall, (0, 1)]$ (even without equality) has infinity axioms[1]. Vedø (1997) showed that a 0-1 law does not hold for second-order existential logic when the first-order part is in this class (again, even without equality).

The current paper sharpens some (prefixes $\forall^2\exists\forall$, $\forall\exists\forall\exists$, $\forall\exists\forall^2$) of the results of Jordan and Zeugmann (2012) and so we briefly outline the improvement that allows us to minimize the prefix considered. The untestable property considered there is closely related to the untestable property of Alon *et al.* (2000), but modified to minimize the number of quantifiers used. These properties are essentially first-order expressible versions of checking an explicitly given isomorphism between two graphs[2]. In fact, restricting the properties to checking an explicitly given isomorphism between undirected, bipartite graphs maintains hardness for testing. See Figure 1(a) for an example where the goal is to check whether the directed edges give an isomorphism between the two bipartite subgraphs. However, graph isomorphism seems to require one to discuss at least four vertices simultaneously (because one wishes to state that an edge is present iff its image is present and the edges are disjoint in general).



(a) A graph with P_b (b) A graph with P_e (c) A graph with P_f

Fig. 1. Properties P_b, P_e and P_f

[1] An infinity axiom is a sentence that has only infinite models.
[2] Graph isomorphism is generally hard for testing, cf. Fischer and Matsliah (2008).

Sharing one of the partitions would seem to remove the need for four quantifiers. See Figure 1(b) for an example where the goal is again to check whether the directed edges give an isomorphism between the two halves of the graph. The resulting property is perhaps closer to a variant of *function* isomorphism, e.g., for functions $f, g\colon \{1,\dots,n\} \to \{0,1\}^n$ where the bit i of $f(j)$ is 1 if there is an edge from j in the leftmost partition to i in the middle partition and likewise for $g(j)$ and the right partition. This property is not first-order expressible, but there is a somewhat tedious first-order encoding that is sufficiently similar. Figure 1(c) gives an example of this first-order property; details are in Section 3.

The connection with function isomorphism allows us to leverage recent work on the testability of (Boolean) function isomorphism and use recent ideas and techniques from Alon and Blais (2010) to prove Lemma 2. In Section 4, we use a variation of this property that removes the use of equality.

2 Preliminaries

The goal in property testing is always to distinguish structures that have some property from those that are far from having the property. Here, we focus on first-order expressible properties of directed graphs and so we begin with the necessary definitions.

Definition 1. A graph is an ordered pair $G = (V, E)$, where V is a finite set and $E \subseteq V \times V$ a binary relation defined on V.

Let $\mathbb{N} = \{1, 2, 3, \dots\}$ denote the set of all natural numbers. We generally identify V with the first n natural numbers $[n] := \{1, \dots, n\}$ and call $\#(G) := |V| = n$ the size of a graph G. Furthermore, let $\mathcal{G}^n$ be the set of graphs of size n and let $\mathcal{G} := \cup_{n \geq 0} \mathcal{G}^n$ be the set of all (finite) graphs. Note that our graphs are directed and may contain loops.

A property $P \subseteq \mathcal{G}$ of graphs is any set of graphs. We are particularly interested in first-order expressible properties. Our logic is a basic first-order predicate logic with equality. There are no function or constant symbols. We focus on first-order properties of graphs, and so the only predicate symbol (besides the special symbol $=$) is the binary edge symbol E.

A sentence φ defines a property in the natural way,

$$P_\varphi := \{G \mid G \in \mathcal{G}, G \models \varphi\}\,.$$

We require a distance between graphs and properties, which we define in the following way: We denote the symmetric difference of sets M and N by $M \triangle N$ and let E^A and E^B be the edge predicates of graphs A and B, respectively.

Definition 2. Let $n \in \mathbb{N}$, let $A = (V, E^A)$ and $B = (V, E^B)$ be two graphs defined such that $|V| = n$. The distance between A and B is

$$\operatorname{dist}(A, B) := |E^A \triangle E^B| / n^2\,.$$

The distance generalizes to properties, $\mathrm{dist}(A, P) := \min_{B \in P} \mathrm{dist}(A, B)$. Definition 2 results in a typical model of testing based on the dense graph model introduced by Goldreich *et al.* (1998). We now proceed to the remaining testing definitions.

Definition 3. An *ε-tester* for property P is a randomized algorithm that makes queries for the existence of edges in a graph A. The tester must accept with probability at least $2/3$ if A has P and must reject with probability at least $2/3$ if $\mathrm{dist}(A, P) \geq \varepsilon$.

Definition 4. Property P is called *testable* if there is some function $c(\varepsilon)$ and for every $\varepsilon > 0$, an ε-tester for P such that the tester makes at most $c(\varepsilon)$ queries.

Note that the query complexity is bounded by a function that does not depend on the size of the graphs. We allow different ε-testers for each $\varepsilon > 0$ and so this is a non-uniform model. However, we are focused on proving untestability and our results hold even in the non-uniform case.

Next, we will define *indistinguishability*, a relation on properties introduced by Alon *et al.* (2000) that preserves testability. However, testers can focus on *loops* and distinguish between structures that have an asymptotically small difference (because the number of loops is asymptotically dominated by the number of non-loops). We therefore begin with an alternative definition of distance. In the following, $\oplus$ denotes exclusive-or.

Definition 5. Let $n \in \mathbb{N}$, $n \geq 2$, and let U be any universe such that $|U| = n$. Furthermore, let $A = (U, E^A)$ and $B = (U, E^B)$ be any two graphs with universe U. For notational convenience, let

$$d_1(A, B) := \frac{\left|\left\{x \mid x \in U \text{ and } E^A(x, x) \oplus E^B(x, x)\right\}\right|}{n}, \text{ and}$$

$$d_2(A, B) := \frac{\left|\left\{(x_1, x_2) \mid x_1, x_2 \in U,\, x_1 \neq x_2, \text{ and } E^A(x_1, x_2) \oplus E^B(x_1, x_2)\right\}\right|}{n(n-1)}.$$

The *mr-distance* between A and B is

$$\mathrm{mrdist}(A, B) := \max\left\{d_1(A, B), d_2(A, B)\right\}.$$

Definition 6. Two properties P and Q of graphs are *indistinguishable* if they are closed under isomorphisms and for every $\varepsilon > 0$ there exists an N_ε such that for any graph A with universe of size $n \geq N_\varepsilon$, if A has P then $\mathrm{mrdist}(A, Q) \leq \varepsilon$ and if A has Q then $\mathrm{mrdist}(A, P) \leq \varepsilon$.

An important property of indistinguishability is that it preserves testability. The proof of the following is analogous to that given in Alon *et al.* (2000).

Lemma 1. *If P and Q are indistinguishable, then P is testable if and only if Q is testable.*

In fact, as the proof constructs an ε-tester for P by iterating an $\varepsilon/2$-tester for Q three times, one can also relate the query complexities of P and Q.

Definition 5 (mrdist) is a distance measure that can be used in place of Definition 2 (dist) when defining testability. The resulting model makes testing strictly more difficult than using dist, see Jordan and Zeugmann (2012). We refer to properties that are testable using mrdist as mr-testable. In Section 3, we prove the untestability of $\forall\exists\forall$ with equality in both models[3]. However, our proof for the class without equality (cf. Section 4) is restricted to proving it is not mr-testable. We suspect that this restriction can be removed.

Many proofs of hardness for testability rely on Yao's Principle (1977), an interpretation of von Neumann's minimax theorem for randomized computation. For completeness, we state the version that we use.

Principle 1 (Yao's Principle). *If there is an $\varepsilon \in (0,1)$ and a distribution over $\mathcal{G}^n$ such that all deterministic testers with complexity c have an error-rate greater than $1/3$ for property P, then property P is not testable with complexity c.*

The definition of "testable" is of course our usual one involving random testers. In general, one seeks to show that for sufficiently large n and some increasing function $c := c(n)$, there is a distribution of inputs such that all deterministic testers with complexity c have error-rates greater than $1/3$.

Finally, we briefly define the notation we use to specify prefix-vocabulary classes. See Börger *et al.* (1997) for details and related material.

Definition 7. Let Π be a string over the four-character alphabet $\{\exists, \forall, \exists^*, \forall^*\}$. Then $[\Pi, (0,1)]_=$ is the set of sentences in prenex normal form which satisfy the following conditions:

1. The quantifier prefix is contained in the regular language given by Π (for technical reasons, one usually treats $\exists$ and $\forall$ as matching the relevant quantifier and also the empty string).
2. There are zero (0) monadic predicate symbols.
3. In addition to the equality predicate ($=$), there is at most one (1) binary predicate symbol.
4. There are no other predicate symbols.

That is, $[\Pi, (0,1)]_=$ is the set of prenex sentences in the logic defined above whose quantifier prefixes match Π. If the second component of the specification is *all*, then conditions two and three are removed (any number of predicate symbols with any arities are acceptable).

3 The Case with Equality

Our goal in this section is Theorem 1.

[3] The proof assumes the use of dist. However, testing with mrdist is strictly more difficult and so the proof also implies untestability using mrdist.

Theorem 1. *The prefix class* $[\forall\exists\forall, (0,1)]_=$ *is not testable.*

We begin by outlining the proof. First, we define P_f, a property expressible in the class $[\forall\exists\forall, (0,1)]_=$ which, as described in Subsection 1.1, is in some sense a somewhat tedious but first-order expressible variant of checking (explicit) isomorphism of undirected bipartite graphs in tripartite graphs. We then define a variant P_2, in which the isomorphism is not explicitly given and we must test whether there exists some suitable isomorphism. Although this increases the complexity of deciding the problem from checking an isomorphism to finding one, it does not change hardness for testing. We show that P_2 and P_f are indistinguishable and so P_2 is testable iff P_f is testable. Finally, we prove directly that P_2 is untestable, even with $o(\sqrt{n})$ queries, using an argument based on a recent proof by Alon and Blais (2010).

Proof (Theorem 1). We begin by defining P_f. Formally, it is the set of graphs satisfying the following conjunction of four clauses (see Figure 1(c) for an example):

$$\begin{aligned}
\forall x\exists y\forall z: \quad & \{\,((\neg E(x,x) \wedge \neg E(z,z) \wedge x \neq z) \rightarrow E(x,z)) \\
\wedge \quad & (E(x,x) \rightarrow (E(x,y) \wedge \neg E(y,y) \wedge [(\neg E(z,z) \wedge E(x,z)) \rightarrow y = z])) \\
\wedge \quad & (\neg E(x,x) \rightarrow (E(y,x) \wedge E(y,y) \wedge [(E(z,z) \wedge E(z,x)) \rightarrow y = z])) \\
\wedge \quad & ((E(x,x) \wedge E(z,z)) \rightarrow [\neg E(y,y) \wedge E(x,y) \wedge (E(x,z) \leftrightarrow E(y,z))])\,\}
\end{aligned}$$

A graph satisfies this formula if the following conditions are all satisfied:

1. The nodes without loops form a complete subgraph.
2. For every node x with a loop, there is exactly one y without a loop such that there is an edge from x to y.
3. For every node y without a loop, there is exactly one x with a loop such that there is an edge from x to y.
4. For all nodes x, z with loops, and y the unique node without a loop such that $E(x,y)$, it holds that $E(x,z)$ iff $E(y,z)$.

Property P_2 below is similar to P_f, except that the isomorphism is not explicitly given.

Definition 8. A graph $G = (V, E)$ has P_2 if it satisfies the following conditions:

1. There is a partition[4] $V_1, V_2 \subseteq V$ such that $|V_1| = |V_2|$, there are loops $(E(x,x))$ on all $x \in V_1$ and no loops $(\neg E(x,x))$ for all $x \in V_2$.
2. The nodes without loops form a complete subgraph.
3. There are no edges from a node with a loop to a node without a loop.
4. There exists a bijection $b\colon V_1 \rightarrow V_2$ such that if x, z have loops, then $E(x,z)$ iff $E(b(x), z)$.

It is not difficult to show that properties P_f and P_2 are indistinguishable.

[4] V_1, V_2 partition V if $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.

Claim 1. *Properties P_f and P_2 are indistinguishable.*

Proof (Claim 1). Let $\varepsilon > 0$ be arbitrary and let $N_\varepsilon = \varepsilon^{-1}$. Assume that G has property P_2 and that $\#(G) > N_\varepsilon$. We will show that $\mathrm{mrdist}(G, P_f) < \varepsilon$.

Graph G has P_2 and so there is a bijection satisfying Condition 4 of Definition 8. We therefore add the edges $E(i, b(i))$ making the isomorphism (from V_1 to V_2) explicit. The resulting graph G_f has P_f.

We have made exactly $n/2$ modifications, all to non-loops, and $n - 1 \geq N_\varepsilon$, so $\mathrm{mrdist}(G, P_f) \leq \mathrm{mrdist}(G, G_f) = 1/2(n-1) < \varepsilon$.

The converse is analogous; given a G that has P_f, simply remove the $n/2$ edges from loops to non-loops after using them to construct a suitable bijection b. □

Properties P_f and P_2 are indistinguishable and so (by Lemma 1), it suffices to show that P_2 is is untestable. Lemma 2 below is stronger than necessary, and actually implies a $\Omega(\sqrt{n})$ lower bound for testing P_f per the discussion following Lemma 1. □

Lemma 2. *Fix $0 < \varepsilon < 1/2$. Any ε-tester for P_2 must perform $\Omega(\sqrt{n})$ queries.*

Proof (Lemma 2). The proof is via Yao's Principle (cf. Principle 1), and so we define two distributions, D_{no} and D_{yes} and show that all deterministic testers have an error-rate greater than 1/3 for property P_2 when the input is chosen randomly from D_{no} with probability 1/2 and from D_{yes} with probability 1/2.

In the following, we consider a distribution over graphs of sufficiently large size $2n$, and an arbitrarily fixed partition of the vertices into V_1 and V_2 such that $|V_1| = |V_2| = n$ (e.g., let the vertices be the integers $V := [2n]$, $V_1 := [n]$, and $V_2 := V \backslash V_1$).

We begin with D_{no}, defined as the following distribution:

1. Place a loop on each vertex in V_1 and place no loops in V_2.
2. Place each possible edge (except loops) in $V_1 \times V_1$ and $V_2 \times V_1$ uniformly and independently with probability 1/2.

That is, D_{no} is the uniform distribution of graphs (with this particular partition) satisfying the first three conditions of P_2.

Next, we define D_{yes} as the following:

1. Choose uniformly a random bijection $\pi \colon V_1 \to V_2$.
2. Place a loop on each vertex in V_1 and place no loops in V_2.
3. For each possible edge $(i, j \neq i) \in V_1 \times V_1$, uniformly and independently place *both* (i, j) and $(\pi(i), j)$ with probability 1/2 (otherwise place *neither*).

It is easy to see that D_{yes} generates only positive instances. Next, we show that D_{no} generates negative instances with high probability.

Lemma 3. *Fix $0 < \varepsilon < 1/2$ and let n be sufficiently large. Then,*

$$\Pr_{G \sim D_{\mathrm{no}}} [\mathrm{dist}(G, P_2) \leq \varepsilon] = o(1)\,.$$

Proof (Lemma 3). The distribution D_{no} is the uniform distribution over graphs of size $2n$ with a particular partition satisfying the first three conditions of P_2. Let G_ε be the set of graphs G' of size $2n$ satisfying these conditions and such that $\operatorname{dist}(G', P_2) \leq \varepsilon$ (regardless of the partition).

Counting the number of such graphs shows

$$|G_\varepsilon| \;\leq\; \binom{2n}{n} 2^{n(n-1)} n! \sum_{i=0}^{\lceil \varepsilon 2n^2 \rceil} \binom{2n^2}{i} \;\leq\; \binom{2n}{n} 2^{n(n-1)} n! 2^{H(\epsilon) 2n^2},$$

where $H(\varepsilon) := -\varepsilon \log \varepsilon - (1-\varepsilon)\log(1-\varepsilon)$ is the binary entropy function (cf. Lemma 16.19 in Flum and Grohe (2006) for the bound on the summation).

Distribution D_{no} produces each of $2^{n(n-1)} 2^{n^2}$ graphs with equal probability, and so

$$\begin{aligned} \Pr_{G \sim D_{\text{no}}} [\operatorname{dist}(G, P_2) \leq \varepsilon] \;&\leq\; \frac{|G_\varepsilon|}{2^{n(n-1)+n^2}} \;\leq\; \binom{2n}{n} n! 2^{H(\epsilon) 2n^2} / 2^{n^2} \\ &\approx\; \frac{4^n n! 2^{H(\epsilon) 2n^2}}{\sqrt{\pi n} 2^{n^2}} \;=\; o(1)\,. \end{aligned}$$

The approximation is asymptotically tight, which suffices. □

We have shown that D_{yes} generates only positive instances and that (with high probability) D_{no} generates instances that are ε-far from P_2. Next, we show that (again, with high probability) the two distributions look the same to testers making only $o(\sqrt{n})$ queries.

The proof is similar to a proof by Alon and Blais (2010). We begin by defining two random processes, P_{no} and P_{yes}, which answer queries from testers and generate instances according to D_{no} and D_{yes}, respectively.

Process P_{no} is defined in the following way:

1. Choose uniformly a random bijection $\pi\colon V_1 \to V_2$.
2. Intercept all queries from the tester and respond as follows:
   (a) To queries $E(i,i)$ with $i \in V_1$: respond 1.
   (b) To queries $E(i,i)$ with $i \in V_2$: respond 0.
   (c) To queries $E(i,j)$ with $i \in V_1$ and $j \in V_2$: respond 0.
   (d) To queries $E(i,j)$ with $i \neq j \in V_1$: quit if we have queried $E(\pi(i), j)$, otherwise respond 1 or 0 randomly with probability 1/2 in each case.
   (e) To queries $E(i,j)$ with $i \in V_2$ and $j \in V_1$: quit if we have queried $E(\pi^{-1}(i), j)$, otherwise respond 1 or 0 randomly with probability 1/2 in each case.
3. When the process has quit or the tester has finished its queries, complete the generated instance in the following way: First, fix the edges that were queried according to our answer. Next, place loops on each vertex in V_1, no loops in V_2 and no edges from V_1 to V_2. Place each remaining possible edge, place it (uniformly, independently) with probability 1/2, *ignoring* π.

We define P_{yes} in the same way, except for the final step. When P_{yes} quits or the tester finishes, it fixes the edges that were queried according to its answers, and *also* fixes the corresponding edges (when relevant) according to π. More precisely, for each fixed $E(i,j)$ with $i \neq j \in V_1$, we also fix $E(\pi(i),j)$ and for fixed $E(i,j)$ with $i \in V_2, j \in V_1$, we also fix $E(\pi^{-1}(i),j)$, in both cases the same as our response to $E(i,j)$ (not randomly). The remaining edges are placed as in P_{no}.

Note that P_{no} generates instances according to D_{no} and P_{yes} generates instances according to D_{yes}. In addition, P_{yes} and P_{no} behave identically until they quit or answer all queries. In particular, if a tester does not cause the process to quit, the distribution of responses of its queries is identical for the two processes. We show that, with high probability, a tester that makes $o(\sqrt{n})$ queries does not cause either process to quit.

Lemma 4. *Let T be a deterministic tester which makes $o(\sqrt{n})$ queries, and let T interact with P_{yes} or P_{no}. In both cases,*

$$\Pr[T \text{ causes the process to quit}] = o(1)\,.$$

Proof (Lemma 4). The condition causing the process to quit is identical in P_{yes} and P_{no}. The probability that any pair of queries $E(i,j)$ and $E(i',j')$ cause the process to quit is at most

$$\Pr[i' = \pi(i) \text{ or } i = \pi(i')] \leq \frac{(n-1)!}{n!} = 1/n\,.$$

The tester makes at most $o(\sqrt{n})$ queries and so

$$\Pr[T \text{ causes the process to quit}] \leq o(\sqrt{n})^2 O(1/n) = o(1)\,.$$

□

Any deterministic tester T which makes $o(\sqrt{n})$ queries can only distinguish between D_{yes} and D_{no} with probability $o(1)$, but it must accept D_{yes} with probability $2/3$, and reject D_{no} with probability $2/3 - o(1)$. It is impossible for T to satisfy both conditions, and the lemma follows from Principle 1. □

4 The Case without Equality

In Section 3, we proved that $[\forall\exists\forall, (0,1)]_=$ is untestable. The formula proved untestable contains equality, and so we now consider the class *without* equality. The main result in this section is Theorem 2, stating that $[\forall\exists\forall, (0,1)]$ is not mr-testable. Although this seems to be a tradeoff between the presence of equality and the "degree" of testability, we suspect that this class is not testable under either definition. The proof is very similar to the proof above.

Theorem 2. *There are properties in $[\forall\exists\forall, (0,1)]$ that are not mr-testable, even given $o(\sqrt{n})$ queries.*

Proof (Theorem 2). The proof is similar to the proof of Theorem 1. We will begin by defining a property P_f that is expressible in our class. We will then define a property P which is indistinguishable from P_f, and use Yao's Principle to show that P is not mr-testable.

A graph has property P_f if it satisfies the following conditions:

1. For every x with a loop, there is an outgoing edge to at least one y without a loop.
2. For every x without a loop, there is an incoming edge from at least one y without a loop.
3. There are no edges between vertices without loops.
4. For every x with a loop, there is an edge to at least one y without a loop such that for all z with loops, the following holds: There is a directed edge from y to z iff there are an odd number[5] of directed edges between x and z.
5. For every x without a loop, there is an incoming edge from at least one y with a loop such that for all z with loops, the following holds: There is a directed edge from x to z iff there are an odd number of directed edges between y and z.

More formally, P_f is the set of graphs that satisfy the following formula:

$$\begin{aligned}
\forall x \exists y \forall z : \quad & \{\, (E(x,x) \rightarrow (\neg E(y,y) \wedge E(x,y))) \\
\wedge \quad & (\neg E(x,x) \rightarrow (E(y,y) \wedge E(y,x))) \\
\wedge \quad & ((\neg E(x,x) \wedge \neg E(z,z)) \rightarrow \neg E(x,z)) \\
\wedge \quad & ((E(x,x) \wedge E(z,z)) \rightarrow ((E(x,z) \oplus E(z,x)) \leftrightarrow E(y,z))) \\
\wedge \quad & ((\neg E(x,x) \wedge E(z,z)) \rightarrow ((E(y,z) \oplus E(z,y)) \leftrightarrow E(x,z))) \,\}
\end{aligned}$$

Next, we define a property P that we will show to be indistinguishable from P_f. A graph has property P if it satisfies the following conditions:

1. There is a partition of the vertices into (non-empty) V_1, V_2.
2. All vertices in V_1 have loops and no vertices in V_2 have loops.
3. There are no edges in $V_2 \times V_2$.
4. There exist functions $f\colon V_1 \rightarrow V_2$ and $g\colon V_2 \rightarrow V_1$ satisfying the following: For all $x, z \in V_1$, there is an edge from $f(x)$ to z iff there are an odd number of directed edges between x and z. For all $x \in V_2$ and $z \in V_1$, there is an edge from x to z iff there are an odd number of directed edges between $g(x)$ and z.

It is not difficult to show that P and P_f are indistinguishable.

Lemma 5. *Properties P and P_f are indistinguishable.*

[5] There is an odd number of edges between x and z if there is a directed edge from x to z or from z to x, but not both. Note that a loop is counted as an even number of edges.

Proof (Lemma 5). Let G be graph with property P_f and let $\varepsilon > 0$ be arbitrarily fixed. Then, G also has property P, that is $\text{mrdist}(G, P) = 0$. In the other direction, if the graph G has property P, then we can satisfy property P_f by adding at most $O(n)$ (non-loop) edges from x to $f(x)$ and $g^{-1}(y)$ to y. Thus, $\text{mrdist}(G, P_f) \leq O(n)/\Theta(n^2) = o(1) < \varepsilon$ for sufficiently large graphs. □

Indistinguishability preserves testability (cf. Lemma 1) and so it suffices to show that P is untestable. Lemma 6 below is stronger than necessary and actually implies a $\Omega(\sqrt{n})$ lower bound for testing P_f per the discussion following Lemma 1. □

Lemma 6. *There is an $0 < \varepsilon < 1/2$ such that any mr-style ε-tester for P must perform $\Omega(\sqrt{n})$ queries.*

Proof (Lemma 6). The proof is via Yao's Principle (cf. Principle 1) and so we must define a distribution of inputs and show that all deterministic ε-testers have an error rate greater than $1/3$ for P on inputs from the distribution. For our distribution, we will draw from a distribution D_{no} with probability $1/2$ and from a distribution D_{yes} with probability $1/2$.

In the following, we consider distributions over graphs with sufficiently large vertex set $[2n]$ and an arbitrarily fixed partition of the vertices into V_1 and V_2 such that $|V_1| = |V_2| = n$.

We begin with D_{no}, defined as the following distribution:

1. Place loops on all vertices in V_1 and no loops in V_2.
2. For each ordered pair in $V_1 \times V_2$, place a directed edge with probability $1/2$.
3. For each unordered pair $\{i, j\}$, $i, j \in V_1$ and $i \neq j$, with probability $1/2$ place no edge, and with probability $1/2$ place a single directed edge, from i to j if $i \leq j$ and from j to i if $j \leq i$.
4. For each ordered pair in $V_2 \times V_1$, with probability $1/2$ place the directed edge and with probability $1/2$ do not.

Note that D_{no} is the uniform distribution of graphs that satisfy the following conditions:

1. The vertex set is $[2n]$ and the vertices with loops follow the given partition.
2. There are no undirected edges between vertices with loops (when a loop is not considered an undirected edge).
3. There are no edges between vertices without loops.
4. All directed edges (i, j) between vertices with loops satisfy $i \leq j$.

Next, we define D_{yes}.

1. Place loops on all vertices in V_1 and no loops in V_2.
2. For each ordered pair in $V_1 \times V_2$, place a directed edge with probability $1/2$.
3. Choose uniformly a random bijection $\pi \colon V_1 \to V_2$.
4. For each unordered pair in $(i, j \neq i) \in V_1 \times V_1$, with probability $1/2$ do the following: Place directed edges $(\pi(i), j)$ and $(\pi(j), i)$, and then place either (i, j) (if $i \leq j$) *or* (j, i) (if $j \leq i$). Otherwise, do not place any edges right now.

Distribution D_{yes} generates only positive instances for P. Now, we show that with high probability, D_{no} generates instances that are ε-far.

Lemma 7. *Let $\varepsilon > 0$ be sufficiently small and n be sufficiently large. Then,*

$$\Pr_{G \sim D_{\text{no}}}[\text{mrdist}(G, P) \leq \varepsilon] = o(1)\,.$$

Proof (Lemma 7). Distribution D_{no} is the uniform distribution over graphs of size $2n$ with a fixed partition V_1, V_2 satisfying the following:

1. All vertices in V_1 have loops and no vertices in V_2 have loops.
2. There are no undirected edges between vertices with loops.
3. There are no edges between vertices without loops.
4. All directed edges (x, y) between vertices with loops satisfy $x \leq y$.

We want a small upper-bound on the probability of a graph being drawn from D_{no} that is not ε-far from P. Since D_{no} is the uniform distribution over a certain class of graphs, this probability is

$$\frac{\left|\{G \mid G \sim D_{\text{no}},\, \text{mrdist}(G, P) \leq \varepsilon\}\right|}{\left|\{G \mid G \sim D_{\text{no}}\}\right|}\,.$$

The number of distinct graphs produced by D_{no} is $2^{\binom{n}{2}} 2^{2n^2} = 2^{2.5n^2 - n/2}$.

Let G_{2n} be the set of graphs with vertices $[2n]$ that have property P and are not ε-far from all graphs in D_{no}. Then,

$$\Pr_{G \sim D_{\text{no}}}[\text{mrdist}(G, P) \leq \varepsilon] \leq \frac{|G_{2n}| \sum_{i=0}^{\lfloor 4\varepsilon n^2 \rfloor} \binom{4n^2}{i}}{2^{2.5n^2 - n/2}}\,. \tag{1}$$

Note that any graph G that is not ε-far from all graphs in D_{no} must have loops on $n - \varepsilon n \leq j \leq n + \varepsilon n$ vertices. Therefore,

$$\begin{aligned} |G_{2n}| &\leq \sum_{j=n-\varepsilon n}^{n+\varepsilon n} \binom{2n}{j} 4^{\binom{j}{2}} 2^{j(2n-j)} j^{2n-j} \\ &\leq (2\varepsilon n + 1)\binom{2n}{n} 2^{2\binom{n+\varepsilon n}{2} + (n+\varepsilon n)^2 + (n+\varepsilon n)\log(n+\varepsilon n)}\,. \end{aligned} \tag{2}$$

Using the (asymptotically tight) estimate $\binom{2n}{n} \approx 4^n/\sqrt{\pi n}$, we see that (2) is approximately

$$\frac{(2\varepsilon n + 1)}{\sqrt{\pi n}} 2^{2n + (n+\varepsilon n)^2 + (n+\varepsilon n)(n+\varepsilon n - 1) + (n+\varepsilon n)\log(n+\varepsilon n)}\,.$$

Combining this with Inequality (1) and using that $\sum_{i=0}^{\lfloor \varepsilon 4n^2 \rfloor} \binom{4n^2}{i} \leq 2^{H(\varepsilon) 4n^2}$, where $H(\varepsilon) = -\varepsilon \log \varepsilon - (1-\varepsilon)\log(1-\varepsilon)$ is the binary entropy function (cf. Lemma 16.19 in Flum and Grohe (2006)), we get

$$\begin{aligned} &\Pr_{G \sim D_{\text{no}}}[\text{mrdist}(G, P) \leq \varepsilon] \\ &\leq \frac{2\varepsilon n + 1}{\sqrt{\pi n}} 2^{-n^2/2 + 4H(\varepsilon)n^2 + 3/2n + 2(\varepsilon^2 + \varepsilon)n^2 + \varepsilon n + (n+\varepsilon n)\log(n+\varepsilon n)} = o(1)\,, \end{aligned}$$

because the $-n^2/2$ in the exponent dominates when ε is sufficiently small. □

We have shown that D_{yes} generates only positive instances and, with high probability, D_{no} generates ε-far instances. Next, we show that, with high probability, the two distributions look identical to testers making only $o(\sqrt{n})$ queries. The proof is similar to a proof by Alon and Blais (2010).

We begin by defining two random processes, P_{no} and P_{yes}, which answer queries from testers and generate instances according to D_{no} and D_{yes}, respectively.

Process P_{no} is defined in the following way:

1. Choose uniformly a random bijection $\pi\colon V_1 \to V_2$.
2. Intercept all queries from the tester and respond as follows:
   (a) To queries $E(i,i)$ with $i \in V_1$, respond 1.
   (b) To queries $E(i,i)$ with $i \in V_2$, respond 0.
   (c) To queries $E(i,j)$ with $i \in V_2, j \in V_2$, respond 0.
   (d) To queries $E(i,j)$ with $i \in V_1, j \in V_2$, randomly respond 1 or 0 with probability $1/2$ in each case.
   (e) To queries $E(i,j)$ with $i > j \in V_1$, respond 0.
   (f) To queries $E(i,j)$ with $i < j \in V_1$, quit if we have queried $E(\pi(i),j)$ or $E(\pi(j),i)$. Otherwise randomly respond 1 or 0 with probability $1/2$ in each case.
   (g) To queries $E(i,j)$ with $i \in V_2, j \in V_1$, quit if we have queried $E(\pi^{-1}(i),j)$ or $E(j,\pi^{-1}(i))$. Otherwise randomly respond 1 or 0 with probability $1/2$ in each case.
3. When the process has quit, or the tester has finished its queries, complete the generated instance in the following way:
   First, fix the edges that were queried according to our answers. Next, place loops on all vertices in V_1, no loops in V_2 and no edges internal to V_2. Place each edge in $V_1 \times V_2$ uniformly and independently with probability $1/2$. For each remaining possible edge $(i,j) \in V_1 \times V_1$, place the edge uniformly and independently with probability $1/2$ if $i < j$ and do not place the edge if $i > j$. For each remaining possible edge in $V_2 \times V_1$, place the edge uniformly and independently with probability $1/2$ (ignoring π).

We define P_{yes} in the same way, except for the final step. When P_{yes} quits or the tester finishes, it fixes the edges that were queried according to its answers, and also fixes the corresponding edges (when relevant) according to π. More precisely, for each fixed $E(i,j)$ with $i \neq j \in V_1$, we also fix $E(\pi(i),j)$ and $E(j,\pi(i))$, and for fixed $E(i,j)$ such that $i \in V_2, j \neq \pi^{-1}(i) \in V_1$, we also fix $E(\pi^{-1}(i),j)$ and $E(j,\pi^{-1}(i))$, in both cases according to our previous decision. The remaining edges are placed as in P_{no}.

Note that P_{no} generates instances according to D_{no} and P_{yes} generates instances according to D_{no}. In addition, P_{yes} and P_{no} behave identically until they quit or answer all queries. In particular, if a tester does not cause the process to quit, the distribution of responses to queries is identical for the two processes. We show that, with high probability, a tester that makes $o(\sqrt{n})$ queries does not cause either process to quit.

Lemma 8. *Let T be a deterministic tester which makes $o(\sqrt{n})$ queries, and let T interact with P_{yes} or P_{no}. In both cases,*

$$\Pr[T \text{ causes the process to quit}] = o(1)\,.$$

Proof (Lemma 8). The condition causing the process to quit is identical in P_{yes} and P_{no}. The probability that any fixed pair of queries $E(i,j)$ and $E(i',j')$ cause the process to quit is at most

$$\Pr[i' = \pi(i) \text{ or } i' = \pi(j)] \leq \frac{2(n-1)!}{n!} = 2/n\,.$$

The tester makes at most $o(\sqrt{n})$ queries and so

$$\Pr[T \text{ causes the process to quit}] \leq o(\sqrt{n})^2 O(1/n) = o(1)\,.$$

□

Any deterministic tester T which makes $o(\sqrt{n})$ queries can only distinguish between D_{yes} and D_{no} with probability $o(1)$, but it must accept D_{yes} with probability at least $2/3$ and reject D_{no} with probability at least $2/3 - o(1)$. It is impossible for T to satisfy both conditions, so the lemma follows from Principle 1. □

5 Conclusions

Property testing is a kind of randomized approximation, where we take a small, random sample of a structure and seek to determine whether the structure has a desired property or is far from having the property. We focused on the classification problem for testability, wherein we seek to determine exactly which prefix vocabulary classes are testable and which are not.

In particular, we focused on the the testability of first-order properties expressible with quantifier prefix $\forall\exists\forall$. In Section 3, we showed that this prefix can express untestable (directed) graph properties when equality is available. Then, in Section 4 we considered the class without equality. There, we showed that this class remains mr-untestable, however testability using dist remains open. We suspect that this class remains untestable using dist. These results sharpen some of the results of Jordan and Zeugmann (2012).

As mentioned in Subsection 1.1, the current classification for testability closely resembles several other classifications (e.g., those for the finite model property, docility and associated second-order 0-1 laws) and it would be interesting to determine whether it coincides with one of these. In particular, determining the testability of variants of the Gödel class would complete the classification for the special case of predicate logic with equality.

Acknowledgments

We are very thankful to Rūsiņš Freivalds for hosting us as visitors and for the many enlightening discussions on probabilistic algorithms we enjoyed during these stays in Riga. We're also grateful to Neil Immerman for pointing out that removing equality in our untestable properties does not really change the "spirit" of why they are untestable. Section 4 formalizes this for prefix $\forall\exists\forall$. We suspect that the mr-untestable property given there is also untestable when using dist, and that a similar argument can be used to remove equality from the untestable property with prefix $\forall^3\exists$ given in Jordan and Zeugmann (2012). Finally, we thank Hiro Ito for pointing out an omission in a previous version of the proof of Lemma 2 (cf. Jordan and Zeugmann (2011)).

References

Alon, N. and Blais, E. (2010). Testing Boolean function isomorphism. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 2010, Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 394–405. Springer.

Alon, N., Fischer, E., Krivelevich, M., and Szegedy, M. (2000). Efficient testing of large graphs. *Combinatorica*, 20(4):451–476.

Alon, N., Krivelevich, M., Newman, I., and Szegedy, M. (2001). Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862.

Blum, M., Luby, M., and Rubinfeld, R. (1993). Self-testing/correcting with applications to numerical problems. *J. of Comput. Syst. Sci.*, 47(3):549–595.

Börger, E., Grädel, E., and Gurevich, Y. (1997). *The Classical Decision Problem.* Springer-Verlag.

Büchi, J. R. (1960). Weak second-order arithmetic and finite-automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92.

Fischer, E. (2001). The art of uninformed decisions. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126. Columns: Computational Complexity.

Fischer, E. and Matsliah, A. (2008). Testing graph isomorphism. *SIAM J. Comput.*, 38(1):207–225.

Flum, J. and Grohe, M. (2006). *Parametrized Complexity Theory.* Springer.

Freivalds, R. (1977). Probabilistic machines can use less running time. In Gilchrist, B., editor, *Information Processing 77, Proceedings of the IFIP Congress 77, Toronto, Canada, August 8-12, 1977*, pages 839–842, North-Holland.

Freivalds, R. (1979). Fast probabilistic algorithms. In *Mathematical Foundations of Computer Science 1979, Proceedings, 8th Symposium, Olomouc, Czechoslovakia, September 3-7, 1979*, volume 74 of *Lecture Notes in Computer Science*, pages 57–69. Springer-Verlag.

Gill, J. (1977). Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675–695.

Goldreich, O. (2010). Introduction to testing graph properties. Technical Report TR10-082, Electronic Colloquium on Computational Complexity (ECCC).

Goldreich, O., Goldwasser, S., and Ron, D. (1998). Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750.

Jordan, C. and Zeugmann, T. (2011). Untestable properties in the Kahr–Moore–Wang class. In Beklemishev, L. D. and de Queiroz, R., editors, *Logic, Language, Information and Computation, 18th International Workshop, WoLLIC 2011, Philadelphia, PA, USA, May 18-21, 2011, Proceedings*, volume 6642 of *Lecture Notes in Artificial Intelligence*, pages 176–186. Springer, 2011.

Jordan, C. and Zeugmann, T. (2012). Testable and untestable classes of first-order formulae. *J. of Comput. Syst. Sci.*, 78(5):1557–1578.

Kahr, A. S., Moore, E. F., and Wang, H. (1962). Entscheidungsproblem reduced to the ∀∃∀ case. *Proc. Nat. Acad. Sci. U.S.A.*, 48:365–377.

Kolaitis, P. G. and Vardi, M. Y. (2000). 0-1 laws for fragments of existential second-order logic: A survey. In Nielsen, M. and Rovan, B., editors, *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August/September 2000, Proceedings*, volume 1893 of *Lecture Notes in Computer Science*, pages 84–98. Springer.

de Leeuw, K., Moore, E.F., Shannon, C. E., and Shapiro, N. (1965). Computability by probabilistic machines. In Shannon, C. E. and McCarthy, J., editors, *Automata Studies*, pages 183–212. Princeton University Press, Princeton, NJ.

McNaughton, R. and Papert, S. (1971). *Counter-Free Automata*. M.I.T. Press.

Ron, D. (2001). Property testing. In Rajasekaran, S., Pardalos, P. M., Reif, J. H., and Rolim, J., editors, *Handbook of Randomized Computing*, volume II, chapter 15, pages 597–649. Kluwer Academic Publishers.

Ron, D. (2008). Property testing: A learning theory perspective. *Found. Trends Mach. Learn.*, 1(3):307–402.

Ron, D. (2009). Algorithmic and analysis techniques in property testing. *Found. Trends Theor. Comput. Sci.*, 5(2):73–205.

Rubinfeld, R. and Sudan, M. (1996). Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271.

Vedø, A. (1997). Asymptotic probabilities for second-order existential Kahr-Moore-Wang sentences. *J. Symbolic Logic*, 62(1):304–319.

Yao, A. C.-C. (1977). Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE Computer Society.

Received August 25, 2016 , accepted October 4, 2016