Indistinguishability and First-Order Logic

Skip Jordan and Thomas Zeugmann

Division of Computer Science
Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan
{skip,thomas}@ist.hokudai.ac.jp

Abstract. The "richness" of properties that are indistinguishable from first-order properties is investigated. Indistinguishability is a concept of equivalence among properties of combinatorial structures that is appropriate in the context of testability. All formulas in a restricted class of second-order logic are shown to be indistinguishable from first-order formulas. Arbitrarily hard properties, including RE-complete properties, that are indistinguishable from first-order formulas are shown to exist. Implications on the search for a logical characterization of the testable properties are discussed.
Key words: property testing, logic, graph theory, descriptive complexity

1 Introduction

In property testing we are interested in efficiently deciding whether a given structure possesses, or is far from possessing, a desired property. Although algorithmic efficiency is often defined as polynomial time, this is not always ideal. For example, we may not have an explicit representation of the input but instead only the ability to query an oracle for individual bits. These queries may be expensive and even a linear-time algorithm, which requires us to compute the entire input explicitly, may be unacceptable. If we are only concerned about distinguishing with high probability between the input *satisfying* and being *far from* satisfying a property, a sub-linear number of queries may be sufficient.

Testers are randomized algorithms that, given the size of the input, are restricted to a certain number of queries which *does not* depend on the input size. We shall give formal definitions below, but mention here that such algorithms are probabilistic approximation algorithms.

In a certain sense, the whole area can be traced back to Freivalds [14] who introduced a program result checker for matrix multiplication over a finite field. Subsequently, the study of property testing originally arose in the context of program verification (see Blum *et al.* [7] and Rubinfeld and Sudan [22]), and the first explicit definition appears to be in [22]. For surveys of the field see e.g., Fischer [12] and Ron [21].

Characterizing the testable properties has been called the most important problem in the area (see Alon and Shapira [5]). The class of regular languages was shown to be testable in Alon *et al.* [4], a result that was extended in Chockler and Kupferman [8]. However, there are context-free languages that are not

testable [12]. It is then perhaps at first surprising that there are many natural, testable properties that are considerably harder. Testers for NP-complete graph properties including k-color were given in Goldreich *et al.* [16].

Restricting ourselves to characterizations of testable graph properties, the first step towards a logical characterization was obtained by Alon *et al.* [2], later extended by Fischer [13]. They show that all properties expressible in first-order logic (FO) with all quantifier alternations of type "$\exists\forall$" are testable, whereas there exists a property expressible with a single quantifier alternation of type "$\forall\exists$" that is not testable. It is useful to note the equivalence of first-order logic with arithmetic and uniform AC^0 (see Barrington *et al.* [6]). Later, a characterization of the graph properties testable with one-sided error by algorithms that are unaware of the input size was given in [5], a result that was extended to hypergraphs by Rödl and Schacht [20]. An exact combinatorial characterization of the graph properties testable with a constant number of queries has been obtained by Alon *et al.* [3].

In the present paper we focus on a question raised in [13]: the expressive power of first-order logic in the context of indistinguishability and testing. The concept of *indistinguishability* was introduced by Alon *et al.* [2] as a suitable form of equivalence in the context of testing. First-order logic with arithmetic is equivalent to uniform AC^0, and so it is strictly contained in NP (see Furst *et al.* [15]). Therefore, properties that are complete for NP under first-order reductions such as k-color cannot be expressed in FO (see Allender *et al.* [1]). However, it has been noted in Alon *et al.* [2] that there are first-order expressible properties that are *indistinguishable* from such properties, including k-color. In this sense, the descriptive power of first-order logic with indistinguishability is surprisingly rich. We examine the set of properties that are indistinguishable from FO-expressible properties and show that this set is larger than was previously known.

The paper is organized as follows. We begin by giving more formal definitions. In Section 3 we show that all graph properties expressible in a restriction of monadic second-order existential logic (MSO$\exists$) are indistinguishable from FO properties. We next prove that there are arbitrarily-hard properties, including RE-complete properties, that are indistinguishable from FO properties (cf. Section 4). In fact, we can construct arbitrarily-hard *testable* properties. Finally, we discuss the implications of the results obtained (cf. Section 5).

2 Preliminaries

We generally restrict our attention to graph properties, and so give the following definitions for graphs. We consider only finite, undirected graphs without loops. We use G and H to refer to graphs, $n = |G|$ as the number of vertices in a graph, and P, Q and R to refer to properties.

Let G and G' be two graphs having the same vertex set and let $\varepsilon > 0$. If G' can be constructed from G by adding and removing no more than εn^2 edges then we say that G and G' *differ in no more than* εn^2 *places.*

Definition 1 (Alon *et al.* [2]). *Let P be a property of graphs and let $\varepsilon > 0$.*

(1) *A graph G with n vertices is called* ε-far from satisfying *P if no graph G' with the same vertex set, which differs from G in no more than εn^2 places, satisfies P.*
(2) *An* ε-test *for P is a randomized algorithm which, given the quantity n and the ability to make queries whether or not a desired pair of vertices of an input graph G with n vertices are adjacent, distinguishes with probability at least $\frac{2}{3}$ between the case of G satisfying P and the case of G being ε-far from satisfying P.*

Note that in Definition 1 the choice of $\frac{2}{3}$ is of course traditional and arbitrary. Any probability strictly greater than $\frac{1}{2}$ can be chosen and the resulting test can be iterated a constant number of times to achieve any desired accuracy strictly less than one, see e.g., Hromkovič [19].

Definition 2 (Alon *et al.* [2]). *The property P is called* testable *if for every fixed $\varepsilon > 0$ there exists an ε-test for P whose total number of queries is bounded only by a function of ε, which is independent of the size of the input graph.*

We allow the tester to know the size of the input, and to make its queries in any computable fashion. In [17] this was shown to be equivalent to the "non-adaptive" model, where a tester uniformly chooses a set of vertices, receives the induced subgraph, and makes a decision based on whether that subgraph has some fixed property. We note that this definition of testability is not an $o(n)$ number of queries and that "ε-far" clearly depends on n.

In general we do not require a uniformity condition. However, it is very natural to require the ε-tests for P in the above definition to be computable given ε. We refer to properties satisfying this additional condition as *uniformly testable* where the uniform and non-uniform cases differ (Proposition 1).

We note that testers given in the literature are generally presented as a single algorithm that takes ε as a parameter and therefore uniform.

Definition 3 (Alon *et al.* [2]). *Two graph properties P and Q are called* indistinguishable *if for every $\varepsilon > 0$ there exists $N = N(\varepsilon)$ satisfying the following. For every graph G having $n > N$ vertices that satisfies P there exists a graph G' with the same vertex set, differing from G in no more than εn^2 places, which satisfies Q; and for every graph H with $n > N$ vertices satisfying Q there exists a graph H' with the same vertex set, differing from H in no more than εn^2 places, which satisfies P.*

As notation, we use ϕ, ψ and γ to refer to logical formulas. Whether these formulas are first- or second-order will be clear from context. We use Φ to denote a logical interpretation of variables. First-order variables are denoted by x_i, y_i and t_i while second-order variables are denoted by C_i. Members of the universe (nodes in the graph) are referred to as u_i when we wish to distinguish between variables and the nodes bound to them. We write $E(x, y)$ to denote the predicate "there is an edge between x and y." Since we consider only finite, undirected

graphs without loops, it follows that $E(x, y)$ implies $E(y, x)$ for all x, y and that $E(x, x)$ is false for all x.

Logical structures such as graphs allow us to interpret predicate symbols. The combination of structures and interpretations, e.g., (G, Φ), then allows us to interpret formulas. We write $G \models \phi$, read G models ϕ, if formula ϕ holds when interpreting the edge predicate according to G. Inductively we use interpretations to interpret bound variables, and write $(G, \Phi) \models \phi$ if formula ϕ holds when interpreting bound variables according to Φ and the edge predicate (assuming it isn't bound by a second-order quantifier) according to G. For a more formal introduction to the logics used, see e.g., Enderton [9].

We assume that ordering and arithmetic are not present in the logics considered, however ordering and arithmetic can be defined in logics containing existential second-order. In proofs we treat the universal quantifier ($\forall$) as the dual of the existential quantifier ($\exists$). Although properties are often implicitly assumed to be computable, we shall see that there are implications to this assumption, and so explicitly state that we do not make this assumption.

Finally, we define DTIME($f(n)$) in the usual manner as the set of decision problems computable on a deterministic Turing machine in $f(n)$ steps.

3 Monadic Second-Order Existential Logic

In this section we show that all graph properties expressible in a restriction of monadic second-order existential logic (see Definition 4 below) are indistinguishable from FO properties.

Definition 4. *Let rMSO$\exists$ denote the graph properties expressible in monadic second-order existential logic that satisfy the following. For all $P \in$rMSO$\exists$ expressible with r second-order quantifiers, there exists an N such that in all graphs G satisfying P with $n > N$ vertices*

(1) *there exists a set of r vertices such that removing them does not affect P, and*
(2) *adding r disconnected vertices to the graph does not affect P.*

Note that rMSO$\exists$ contains natural problems such as k-color. The first restriction is similar to but weaker than hereditariness. The proof of the following theorem is a generalization of a result regarding the indistinguishability of k-color from FO properties (see, e.g., Alon *et al.* [2]).

Theorem 1. *All properties expressible in rMSO$\exists$ are indistinguishable from properties expressible in FO.*

Proof. Let P be the rMSO$\exists$ property expressed by formula $\phi := \exists C_1 C_2 \dots C_r \psi$, where ψ is first-order. We construct a first-order formula ϕ' expressing property Q such that P and Q are indistinguishable.

$$\phi' := \exists t_1 t_2 \dots t_r \psi',$$

where the symbols t_i do not occur in ψ (if they do occur, simply rename them). Formula ψ' is derived from ψ with the following changes:

1. Replace all occurrences of $C_i(x)$ with $E(t_i, x)$.
2. Replace all quantifiers with restricted quantifiers:
   $\exists x : \gamma$ with $\exists x : (x \neq t_1 \wedge \ldots \wedge x \neq t_r) \wedge \gamma$ and
   $\forall x : \gamma$ with $\forall x : (x \neq t_1 \wedge \ldots \wedge x \neq t_r) \rightarrow \gamma$.

First, let $\varepsilon > 0$ be arbitrary and let G model ϕ. Assume $|G| = n > N$. Choose the set of r vertices guaranteed to exist by Restriction (1) of Definition 4, call them $u_1 \ldots u_r$ and remove them. Call this graph G^-. Take an interpretation Φ under which G^- models ϕ. Replace the u_i as disconnected vertices. Connect u_i and x iff $C_i(x)$ holds in Φ. Call the resulting graph G'. Note that we have changed at most $r(n-1)$ edges, which is less than εn^2 for sufficiently large n.

Claim 1. Graph G' models ϕ'.

Proof. We construct a satisfying interpretation Φ' from Φ. The only change is to bind t_i to u_i. Recall that the t_i appear only where we added them above and thus the u_i are only referred to in these contexts.

Note that:

- $(G^-, \Phi) \models x = y \iff (G', \Phi') \models x = y$,
  because the u_i cannot appear here and all other members are retained.
- $(G^-, \Phi) \models C_i(x) \iff (G', \Phi') \models E(t_i, x)$, by construction.
- Logical operators $\wedge$, $\neg$ are preserved inductively.
- $(G^-, \Phi) \models \exists x : \gamma \iff (G', \Phi') \models \exists x : (x \neq t_1 \wedge \ldots \wedge x \neq t_r) \wedge \gamma$,
  because G^- does not contain the vertices referred to by the t_i.

Therefore, G' models ϕ' and thus has the first-order property Q. □ Claim 1

Next, let $\varepsilon > 0$ be arbitrary and assume that graph H models ϕ'. Assume further that $|H| > N$. Let Φ' be an interpretation satisfying ψ'. Let the vertices bound to the t_i be called u_i. Recall that because of the restricted quantifiers in ϕ', the u_i are referred to only as t_i, and the t_i only occur where we explicitly added them.

Remove the u_i from H and call this graph H^-. Next, re-add the u_i as r isolated vertices and call this graph H'. We claim that both H^- and H' model ϕ, and construct a satisfying interpretation Φ from Φ'. Because of Restriction (2) in Definition 4, it is sufficient to prove that H^- models ϕ as adding r isolated vertices will not affect property P.

We set $C_i(x)$ to be true in Φ iff there is an edge between u_i and x in H.

Claim 2. $(H^-, \Phi) \models \phi$.

Proof. Note that:

- $(H^-, \Phi) \models x = y \iff (H, \Phi') \models x = y$,
  because x and y cannot be bound to u_i on the right.
- $(H^-, \Phi) \models C_i(x) \iff (H, \Phi') \models E(t_i, x)$, by construction.
- Logical operators $\wedge$, $\neg$ are preserved inductively.

– $(H^-, \Phi) \models \exists x : \gamma \iff (H, \Phi') \models \exists x : (x \neq t_1 \wedge \ldots \wedge x \neq t_r) \wedge \gamma$,
because H^- does not contain the vertices referred to by the t_i.

□ Claim 2

Therefore, H^- has property P and by Restriction (2) of Definition 4, H' does too. We have changed at most $r(n-1)$ edges, which is less than εn^2 for sufficiently large n.

Properties P and Q are therefore indistinguishable. □

4 Hard Properties

In the previous section, we showed that every property expressible in a restriction of monadic second-order existential logic is indistinguishable from some FO-expressible property. All properties expressible in this logic are contained in NP by Fagin's [10] theorem. For graphs it is known that MSO∃ is strictly less expressive than SO∃: there are graph properties in NP that are not expressible in MSO∃ (see, Fagin *et al.* [11]). In this section we continue our study of the set of properties indistinguishable from FO-expressible properties, and show that it contains *much harder* properties. We show that this set contains uncomputable properties, and also, for every $f(n)$, computable (and testable) properties that are not computable in time $f(n)$. In this sense, the power of first-order logic in the context of indistinguishability is even larger than previously known. However, we also show that there exist computable properties that are *distinguishable* from every first-order expressible property.

In the next proof we use the concept of RE-completeness. A decision problem is in RE (recursively enumerable) iff there exists a Turing machine that halts and accepts all positive instances and does not accept negative instances. The machine is not required to halt on negative instances. A decision problem D is *RE-complete* iff it is in RE and all other problems in RE can be *decided* by machines given an oracle for D. The halting problem is the canonical RE-complete problem.

Theorem 2. *There exists an RE-complete graph property that is indistinguishable from a FO-expressible property.*

Proof. We define property P such that graph G satisfies it iff

(1) there is a vertex i such that all edges are incident to i, and
(2) taking the degree d of vertex i as the number of a Turing machine M_d in some canonical enumeration $(M_i)_{i \in \mathbb{N}}$ of Turing machines where every Turing machine appears at least once, provided M_d halts on the empty string.

If M_d does not halt on the empty string, then graph G does not have property P. For convenience, we require that machine M_0 in the enumeration halts on the empty string. We shall show that P is an RE-complete property that is indistinguishable from the FO-expressible property of being an empty graph.

Lemma 1. *P is RE-complete.*

Proof. Let HALT=$\{a \mid M_a$ halts on the empty string$\}$. We show that P is RE-complete by reducing HALT to it. On input a, representing Turing machine M_a in the enumeration, we output a graph on $a+1$ vertices. There is an edge between vertices i and j iff exactly one of them is zero. All edges are then incident to the zero vertex, and as the degree of vertex zero is $n-1=a$, the graph has property P iff machine M_a halts on the empty string. □ Lemma 1

Lemma 2. *P is indistinguishable from the FO property of being an empty graph $(\forall x, y : \neg E(x, y))$.*

Proof. Assume graph G satisfies property P and let $\varepsilon > 0$ be arbitrary. Let i denote the vertex that is mentioned in Property (1) of the Theorem, and let d be its degree. Remove the $d \leq n-1$ edges incident to i. By assumption, all edges in G were incident to i, and so the resulting graph is empty. For n sufficiently large, $n-1 < \varepsilon n^2$.

Now assume graph G is an empty graph. By assumption, M_0 halts. Choose an arbitrary vertex i and note that all zero edges are incident to it. Consequently, G has property P. We have not changed any edges, and therefore $0 < \varepsilon n^2$ for all non-zero n. □ Lemma 2

Lemma 1 and 2 directly yield the theorem. □

The following proposition is essentially obvious: an ε-tester is a probabilistic machine. It remains only to mention that we can, by choosing $\varepsilon > 0$ appropriately as a function of n, remove the "approximation" in "probabilistic approximation algorithm." Of course, we then make a number of queries that depends on the input size.

It is important to note that this proposition does not hold in the non-uniform case.

Proposition 1. *All uniformly testable graph properties can be decided by a probabilistic Turing machine with success probability at least $\frac{2}{3}$.*

Proof. Assume graph property P is testable. We construct a probabilistic machine deciding P. On input G of size n, choose ε such that $\varepsilon n^2 < 1$, for example, $\varepsilon = \frac{1}{n^2+1}$. Run the ε-test on G and output the result. Because $\varepsilon n^2 < 1$, being *at most* ε-far from G implies that we may not change *any* edges. We therefore distinguish with probability at least $\frac{2}{3}$ between the case of G satisfying P and G not satisfying P. □

Corollary 1. *All uniformly testable properties are decidable by probabilistic machines.*

Proof. Simply modify εn^2 in the proof to the appropriate definition of ε-far for the vocabulary in question. □

The following also follows immediately, as randomization does not allow us to compute uncomputable functions. We provide a proof sketch for completeness.

Corollary 2. *All uniformly testable properties are recursive.*

Proof. We convert the probabilistic machine into a deterministic machine using the following generic construction.

All probabilistic machines can be modified such that their randomness is taken from a special binary "random tape" that is randomly fixed when the machine is started, in which each digit is 0 or 1 with equal probability.

All halting probabilistic machines must eventually halt, regardless of the random choices made. We can then simulate the machine over all initial segments of increasing lengths, keeping track of "accepting," "rejecting" and "still running" states. Once any given segment has halted, all random strings beginning with that initial segment must also halt. Therefore, the percentage of halting paths is increasing, and we shall eventually reach a length such that at least 70% of the paths have halted. Our error probability is at most $\frac{1}{3}$, strictly less than half of 70% and so we can output the decision of the majority of the halting paths. □

Theorem 3 (Alon *et al.* [2]). *Every first-order property P of the form*

$$\exists x_1, \ldots, x_t \forall y_1, \ldots, y_s : A(x_1, \ldots, x_t, y_1, \ldots, y_s),$$

where A is quantifier-free, is testable.

Note that our RE-complete property P defined in the proof to Theorem 2, is indistinguishable from a first-order property of this form ($t = 0$).

Theorem 4 (Alon *et al.* [2]). *If P and Q are indistinguishable graph properties, then P is testable if and only if Q is testable.*

However, we now see a contradiction in the uniform case. Our RE-complete property P, defined in the proof to Theorem 2, is indistinguishable from a FO-property that, according to Theorem 3, is testable (which it is). Theorem 4 then implies that this RE-complete property P is also testable, which contradicts Corollary 2. Therefore, Theorem 4 is not strictly correct in the uniform case. The proof given in [2] assumes that if input G has strictly less than $N = N(\varepsilon)$ vertices, there exists a decision procedure that gives "accurate output according to whether it satisfies" the property in question. Of course, no such (uniform) procedure exists for RE-complete properties. Theorem 4 holds in the uniform case when restricted to recursive properties.

We can however use a similar construction to Theorem 2 to obtain the following, restricting ourselves to recursive properties. By the time hierarchy theorem, for every computable $f(n)$ there exist computable properties that cannot be decided in time $f(n)$, see Hartmanis and Stearns [18].

We define *arbitrarily-hard properties* to be any set of *computable* properties that "for each computable $f(n)$, contains properties that cannot be computed in DTIME($f(n)$)."

It is of course possible to use other complexity measures. We have restricted these sets to computable properties and so they are obviously infinite.

The reduction in the following proof increases the input length by an exponential factor. However, because we are interested in *arbitrarily-hard* properties and by the time hierarchy theorem, we can choose Q such that it is not computable in, e.g., DTIME($2^{f(n)}$). Then, after the input length is increased exponentially, we have a property that cannot be computed in DTIME($f(n)$). This can be done for all $f(n)$.

Theorem 5. *There are arbitrarily-hard testable properties.*

Proof. Let Q be an arbitrarily-hard property. We define property R such that Q is reducible to R. Let $q(x)$ be the characteristic function for Q with an appropriate encoding of the input. Similar to Theorem 2, we define R to hold in graph G of size n iff

1. there is a vertex i such that all edges are incident to i,
2. and either the degree of i is zero or $q(n) = 1$.

We can obviously reduce Q to R by computing the encoding x, and outputting a graph on x vertices with one edge. We can therefore construct arbitrarily hard properties R.

Using the same proof as that used for Theorem 2, we see that all such R are also indistinguishable from the empty graph. The property of being the empty graph is testable by Theorem 3, and so R is testable by Theorem 4, if it is decidable. We can therefore construct arbitrarily hard, testable properties. □

Computable graph properties that are distinguishable from all first-order properties do exist however. We show the following by a simple diagonalization argument. We define *distinguishable* as "not indistinguishable."

Theorem 6. *There exist computable graph properties that are distinguishable from all first-order properties.*

Proof. We let first-order formula ϕ_i denote the i'th formula in some enumeration of first-order formulas on graphs, in which all such formulas occur infinitely often. We define property Q such that G has property Q iff $\phi_{|G|}$ does not hold on any graph with $|G| = n$ vertices.

We show that Q is distinguishable from all first-order properties by contradiction. Assume that first-order ψ expresses a property that is indistinguishable from Q. Find the first $i > N$ such that $\psi = \phi_i$. There are two cases. First, assume that there is a graph G on i vertices such that G satisfies ψ. Then, there is no graph on i vertices with property Q, by construction. We therefore cannot obtain a graph G' on i vertices with property Q by changing at most εn^2 edges in G, because no such graph exists.

There must then be no such graph G on i vertices satisfying ψ. In this case, by definition all graphs on i vertices have property Q. Taking any of them, we see that we cannot obtain a graph on i vertices satisfying ψ by modifying at most εn^2 edges, because again no such graph exists. There must then be no first-order ψ expressing a property indistinguishable from Q. □

5 Discussion

The descriptive power of first-order logic with indistinguishability is surprisingly large. As we have seen, we can construct testable properties of arbitrary hardness. However, as seen in [2], there are problems in FO with quantifier alternations "$\forall\exists$" that are not testable. So, although the class of testable properties contains arbitrarily hard properties, it does not strictly contain uniform AC^0 or even the context-free languages. In this sense, it is a rather odd class.

A complete, logical characterization of the testable properties must then not contain FO entirely, but must contain arbitrarily hard properties. However, in the uniform case we have seen that all testable properties are recursive, and so a characterization of the uniformly testable properties must not be able to express e.g. RE-complete properties.

We also believe that the distinction between the *uniformly* testable and *non-uniformly* testable properties is important. In particular, given our motivation of searching for a class that is very efficiently computable, it seems undesirable to admit uncomputable properties. In this sense, the uniform case is preferable (Proposition 1). It would also be worthwhile to consider other possible definitions of uniformity.

Acknowledgements. We wish to thank Osamu Watanabe for an inspiring discussion regarding the importance of uniformity conditions.

References

[1] Eric Allender, José L. Balcázar, and Neil Immerman. A first-order isomorphism theorem. *SIAM J. Comput.*, 26(2):539–556, 1997.

[2] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20(4):451–476, 2000.

[3] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. In *STOC '06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 251–260, New York, NY, USA, 2006. ACM.

[4] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862, 2001.

[5] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proceedings, 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005*, pages 429–438, Washington, DC, USA, 2005. IEEE Computer Society.

[6] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC1. *J. of Comput. Syst. Sci.*, 41(3):274–306, 1990.

[7] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. of Comput. Syst. Sci.*, 47(3):549–595, 1993.

[8] Hana Chockler and Orna Kupferman. ω-regular languages are testable with a constant number of queries. *Theoret. Comput. Sci.*, 329(1-3):71–92, 2004.

[9] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, second edition, 2000.
[10] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R.M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings*, volume VII, pages 43 – 73. Amer. Mathematical Soc., 1974.
[11] Ronald Fagin, Larry J. Stockmeyer, and Moshe Y. Vardi. On monadic NP vs. monadic co-NP. *Inform. Comput.*, 120(1):78–92, 1995.
[12] Eldar Fischer. The art of uninformed decisions. *Bulletin of the European Association for Theoretical Computer Science*, 75:97, October 2001. Columns: Computational Complexity.
[13] Eldar Fischer. Testing graphs for colorability properties. *Random Struct. Algorithms*, 26(3):289–309, 2005.
[14] Rūsiņš Freivalds. 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, 1979.
[15] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
[16] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
[17] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Struct. Algorithms*, 23(1):23–57, 2003.
[18] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.
[19] Juraj Hromkovič. *Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms*. Springer, 2005.
[20] Vojtěch Rödl and Mathias Schacht. Property testing in hypergraphs and the removal lemma. In *STOC '07: Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 488–495, New York, NY, USA, 2007. ACM.
[21] Dana Ron. Property testing. In Sanguthevar Rajasekaran, Panos M. Pardalos, John H. Reif, and José Rolim, editors, *Handbook of Randomized Computing*, volume II, chapter 15, pages 597–649. Kluwer Academic Publishers, 2001.
[22] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.