Computational Model Theory: An Overview

Moshe Y. Vardi*
Department of Computer Science
Rice University
Houston, TX 77005-1892
E-mail: vardi@cs.rice.edu
URL: http://www.cs.rice.edu/~vardi

March 27, 1998

Abstract

The computational complexity of a problem is the amount of resources, such as time or space, required by a machine that solves the problem. The descriptive complexity of problems is the complexity of describing problems in some logical formalism over finite structures. One of the exciting developments in complexity theory is the discovery of a very intimate connection between computational and descriptive complexity. It is this connection between complexity theory and finite-model theory that we term *computational model theory*.

In this overview paper we offer one perspective on computational model theory. Two important observations underly our perspective: (1) while computational devices work on encodings of problems, logic is applied directly to the underlying mathematical structures, and this "mismatch" complicates the relationship between logic and complexity significantly, and (2) first-order logic has severely limited expressive power on finite structures, and one way to increase the limited expressive power of first-order logic is by adding fixpoint constructs. These observations motivated the introduction of two absract formalisms: that of *finite-variable infinitary logic* and that of *relational machines*. This paper is the story of how these formalisms are used in computational model theory and their interrelationship.

*Work supported in part by NSF grants CCR-9628400 and CCR-9700061.

1 Introduction

The *computational complexity* of a problem is the amount of resources, such as time or space, required by a machine that solves the problem. Complexity theory traditionally has focused on the computational complexity of problems. A more recent branch of complexity theory focuses on the *descriptive complexity* of problems, which is the complexity of describing problems in some logical formalism over finite structures [Imm89]. One of the exciting developments in complexity theory is the discovery of a very intimate connection between computational and descriptive complexity. It is this connection between complexity theory [Pap94] and finite-model theory[1] [Fag93] that we term *computational model theory*.

In this overview paper we offer one perspective on computational model theory. One of the themes of this perspective is that fact that although the relationship between descriptive and computational complexity is intimate, it is not without its problems, and the "partners" do have some irreconcilable differences. While computational devices work on *encodings* of problems, logic is applied directly to the underlying mathematical structures. As a result, machines are able to enumerate objects that are logically unordered. For example, while we typically think of the set of nodes in a graph as unordered, it does become ordered when it is encoded on the tape of a Turing machine. This "order mismatch" complicates the relationship between logic and complexity significantly [Var82, Imm86, AV95].

A second theme of our perspective is the fact that first-order logic has severely limited expressive power; for example, graph connectivity is not expressible in first-order logic (see [Fag75, AU79, Gai82, GV85]). In view of that fact, there is an emphasis in database theory and in finite-model theory on logics that go beyond first-order logic. One way to increase the limited expressive power of first-order logic is by adding fixpoint constructs. As fixpoint is a fundamental computational construct [Rog67, Acz77], this extension can be viewed as transforming the logic from an assertional logic to a computational logic. Such extensions of first-order logic have been the subject of extensive study – focusing on their expressive power, their relationship to complexity classes, in particular to P and PSPACE, and their asymptotic probability properties (cf. [AV91a, Cha88, CH82, Imm86, KV87, Var82]). Of particular importance among these extensions are *inflationary-fixpoint logic*, which is obtained from first-order logic by adding fixpoints of inflationary first-order formulas, and *noninflationary-fixpoint logic*, which is obtained from first-order logic by adding fixpoints of general first-order formulas.

Since first-order logic has a finitary syntax, a different way to increase its expressive power is to allow for infinitary formation rules. One of the most powerful logics resulting this way is the *infinitary logic* $L_{\infty\omega}$, which allows for arbitrary disjunctions and conjunctions. This logic has been studied extensively on infinite structures (cf. [BF85]), but it is too powerful to be of interest or use in finite-model theory, because every class of struc-

[1] In this paper we are interested exclusively in finite structures. Thus the terms *model* and *structure* refer to finite models and structures only, unless explicitly stated otherwise.

tures that is closed under isomorphisms is definable in $L_{\infty\omega}$. A more interesting extension from the perspective of finite-model theory is that of *finite-variable* infinitary logic $L^{\omega}_{\infty\omega}$, which consists of all formulas of $L_{\infty\omega}$ with a finite number of distinct variables. More formally, $L^{\omega}_{\infty\omega}$ is the union $\bigcup_{k=1}^{\infty} L^{k}_{\infty\omega}$, where $L^{k}_{\infty\omega}$ is the collection of all formulas of $L_{\infty\omega}$ with k variables. The infinitary logic $L^{\omega}_{\infty\omega}$ was studied originally on infinite structures [Bar77], but it turned out to have several uses in finite-model theory [KV92b, KV95]. In fact, $L^{\omega}_{\infty\omega}$ could be said to underlie much of the work on lower bounds for expressibility on finite structures in [Imm82, dR87, LM89, CFI92], although its use there is rather implicit.

One of the reasons for the importance of $L^{\omega}_{\infty\omega}$ in finite-model theory is the fact that from an expressiveness standpoint $L^{\omega}_{\infty\omega}$ constitutes a proper extension of both inflationary- and noninflationary-fixpoint logics [KV92b]. Thus, $L^{\omega}_{\infty\omega}$ can be used to derive both positive and negative results about fixpoint logics. On the positive side, structural results about $L^{\omega}_{\infty\omega}$ transfer to similar results for fixpoint logics, while, on the negative side, lower-bound results for expressibility in $L^{\omega}_{\infty\omega}$ yield, a fortiori, lower bounds for expressibility in fixpoint logics (but not vice-versa). The main advantage of $L^{\omega}_{\infty\omega}$ is that the expressive power of the infinitary logics $L^{k}_{\infty\omega}$ with k variables, $k \geq 1$, has a clean and precise characterization in terms of simple k-pebble games between two players on a pair of structures (cf. [Bar77, Imm82, KV92b]). In contrast, fixpoint logic is not known to possess a similar property. In this paper we review two such applications of $L^{\omega}_{\infty\omega}$. The first concerns expressibility, while the second concerns 0-1 laws.

While $L^{\omega}_{\infty\omega}$ does provide an elegant framework for studying important phenomena such as 0-1 laws [KV92b] and expressive power of query languages [ACY95, KV95], infinitary formulas have a non-effective syntax and can define non-computable queries. We seek here to understand better the connection between effective query languages and $L^{\omega}_{\infty\omega}$. The main focus is to identify natural *effective* fragments of $L^{\omega}_{\infty\omega}$ and relate them to known computational models for queries.

To formalize natural effective fragments of $L^{\omega}_{\infty\omega}$, we define syntactic and semantic notions of recursively enumerable (r.e.) formulas. The syntactic notion is based on the enumeration of the (syntactic) components of the sentence. In contrast, the semantic notion of r.e. requires the set of models of a sentence to be r.e. We show an intimate connection between the effective fragments of $L^{\omega}_{\infty\omega}$ and a formal computing device that subsumes most database query languages. The device, called *relational machine*, consists of a Turing machine interacting with a relational store via first-order queries. This model is similar in spirit to the computation carried out in practical database languages such as C+SQL, where first-order queries are embedded in a host programming language. Unlike Turing machines, which operate on encodings of problems, relational machines operate directly on the underlying mathematical structures, overcoming the order mismatch.

For Turing machines, the most natural measure of complexity is in terms of the size of the input. This measure is not the most natural one for relational machines, since such machines cannot measure the size of their input. In fact, relational machines have a limited *discerning* power, i.e., the power to distinguish between different pieces of their input, since they are only able to manipulate their input relationally. This suggests that it is natural to measure the size of the input to a relational machine with respect to the discerning power of the machine. The new measure gives rise to a new notion of computational complexity, called *relational complexity*, resulting in classes such as PTIME_r (relational polynomial time) and PSPACE_r (relational deterministic polynomial space) [AV95, AVV97].

Relational complexity can serve as a mediator between logical complexity and standard complexity. On one hand, questions about containments among standard complexity classes can be translated to questions about containments among relational complexity classes. On the other hand, the expressive power of fixpoint logic can be *precisely* characterized in terms of relational complexity classes. This tight, three-way relationship among fixpoint logics, relational complexity and standard complexity yields logical analogs to containments among complexity classes. In particular, this shows that some of the most tantalizing questions in complexity theory – P. vs. PSPACE, NP vs. PSPACE, and PSPACE vs. EXPTIME – boil down to *one fundamental issue*: understanding the relative power of inflationary vs. noninflationary 1st-order operations.

2 Fixpoint Logics and Infinitary Logic

2.1 Fixpoint Logics

Since its introduction in the early 1970s [Cod70], the relational model has become the dominant data model for database management systems [Ull89]. The core query language for the relational model is the relational calculus, which is a first-order logic [Cod72]. It is well known, however, that many natural graph-theoretic queries, such as transitive closure, connectivity, acyclicity, and 2-colorability, are not first-order definable [AU79, Fag75, Gai82]. This motivated the proposal by Aho and Ullman to extend first-order logic with a *fixpoint* construct [AU79].

Let σ be a vocabulary, let S be an n-ary relation symbol not in σ, let $\varphi(x_1, \ldots, x_n, S)$ be a first-order formula over a vocabulary $\sigma \cup \{S\}$, and let $\mathbf{D}$ be a structure over σ. The formula φ gives rise to an operator Φ from n-ary relations on the universe D of $\mathbf{D}$ to n-ary relations on D, where

$$\Phi(T) = \{(a_1, \ldots, a_n) : \mathbf{D} \models \varphi(a_1, \ldots, a_n, T)\},$$

for every n-ary relation T on D.

Every such operator Φ generates a sequence of *stages* that are obtained by iterating Φ. We will be interested here in the relationship between the stages of the operator and its *fixpoints*.

Definition 2.1: Let $\mathbf{D}$ be a structure over a vocabulary σ.

- The stages Φ^m, $m \geq 1$, of Φ on $\mathbf{D}$, are defined by the induction:

$$\Phi^1 = \Phi(\emptyset), \ \Phi^{m+1} = \Phi(\Phi^m).$$

- We say that a relation T on D is a *fixpoint* of the operator Φ (or, of the formula φ) if $\Phi(T) = T$.

∎

Intuitively, one would like to associate with an operator Φ the "limit" of its stages. This is possible only when the sequence Φ^m, $m \geq 1$, of the stages "converges", i.e., when there is an integer m_0 such that $\Phi^{m_0} = \Phi^{m_0+1}$ and, hence, $\Phi^{m_0} = \Phi^m$, for all $m \geq m_0$. Notice that in this case Φ^{m_0} is a fixpoint of Φ, since $\Phi^{m_0} = \Phi^{m_0+1} = \Phi(\Phi^{m_0})$. The sequence of stages, however, may not converge. In particular, this will happen if the formula $\varphi(\mathbf{x}, S)$ has no fixpoints. Thus, additional conditions have to be imposed on the formulas considered in order to ensure that the sequence of stages converges.

A formula $\varphi(x_1, \ldots, x_n, S)$ is *inflationary in S* if it is of the form $S(x_1, \ldots, x_n) \vee \psi(x_1, \ldots, x_n, S)$. This guarantees that $T \subseteq \Phi(T)$ for every n-ary relation T. Inflationariness is a natural syntactic condition that guarantees convergence. as the sequence Φ^m, $m \geq 1$, of stages is increasing. If $\mathbf{D}$ is a structure with s elements, then every stage Φ^m has at most s^n elements and, consequently, there is an integer $m_0 \leq s^n$ such that $\Phi^{m_0} = \Phi^m$ for every $m \geq m_0$. Thus, the sequence of stages of $\varphi(\mathbf{x}, S)$ converges to Φ^{m_0}. Φ^{m_0} is the *inflationary fixpoint of* $\varphi(\mathbf{x}, S)$. We write φ^∞ or Φ^∞ to denote the inflationary fixpoint of φ.

Inflationary-fixpoint logic, denoted IFP, is first-order logic augmented with the *inflationary fixpoint* formation rule for *inflationary* formulas. The canonical example of a formula of IFP is provided by the inflationary fixpoint $\varphi^\infty(x, y)$ of the first-order formula

$$S(x, y) \vee E(x, y) \vee (\exists z)(S(x, z) \wedge S(z, y)).$$

In this case $\varphi^\infty(x, y)$ defines the *transitive closure* of the *edge* relation E. It follows that *connectivity* is a property expressible in IFP, but, as noted above not in first-order logic.

Remark 2.2: Fixpoints can also be guaranteed to exist when the formula $\varphi(x_1, \ldots, x_n, S)$ is *positive in S*, namely, when every occurrence of S is in the scope of an even number of negations. In that case, we have that Φ is *monotone*, i.e., $\Phi(T_1) \subseteq \Phi(T_2)$ if $T_1 \subseteq T_2$, and consequently, the sequence Φ^m, $m \geq 1$, of stages is increasing and has a limit that is a fixpoint. In fact, that limit is the least fixpoint of Φ. *Positive-fixpoint logic* is 1st-order logic augmented with the positive-fixpoint formation rule for positive formulas. It is easy to see that IFP is at least as expressive as positive-fixpoint logic. Gurevich and Shelah showed that in fact the two logics have the same expressive power [GS86] (see also [Lei90]). ∎

The complexity-theoretic aspects of IFP were studied in [CH82, Imm86, Var82]. (These papers actually focused on positive fixpoint logic, but, as observed above, positive fixpoint logic and IFP have the same expressive power, which implies that they have the same data complexity in the sense of [Var82].) It is known that IFP captures the complexity class PTIME, where a logic L is said to *capture* a complexity class C if the following holds:

1. All problems expressible in L are in C, and

2. on ordered structures L can express all problems in C.

A more natural definition would be to require that a problem is in C iff it is expressible in L. There are, however, problems in PTIME (e.g., the *evenness* property, checking whether the cardinality of the structure is even) that are not expressible in IFP [CH82]. Thus, requiring L to express precisely the problems in C would be too stringent.

How can we obtain logics with fixpoint constructs that are more expressive than inflationary fixpoint logic? A more powerful logic results if one iterates general first-order operators, until a fixpoint is reached (which may never happen). In this case, we may have non-terminating computations, unlike inflationary fixpoint logic, where the iteration was guaranteed to converge.

Definition 2.3: Let Φ^m, $m \geq 1$, be the sequence of stages of the operator Φ associated with a formula $\varphi(x_1, \ldots, x_n, S)$ over a structure $\mathbf{D}$. If there is an integer m_0 such that $\Phi^{m_0} = \Phi^{m_0+1}$, then we put $\varphi^\infty = \Phi^\infty = \Phi^{m_0}$; otherwise, we set $\varphi^\infty = \Phi^\infty = \emptyset$.[2] We call φ^∞ the *noninflationary fixpoint* of φ on $\mathbf{D}$. *Noninflationary-fixpoint logic* (NFP) is 1st-order logic augmented with the noninflationary fixpoint formation rule for arbitrary 1st-order formulas. ∎

Notice that for every first-order formula $\varphi(x_1, \ldots, x_n, S)$, if $\mathbf{D}$ is a structure with s elements, then either the sequence Φ^m, $m \geq 1$, of stages converges or it cycles. Which of the two is the case can be determined by carrying out at most 2^{s^n} iterations of Φ (as there are at most 2^{s^n} distinct n-ary relations over s elements). Thus, the computation of the noninflationary fixpoint requires space polynomial in the sizefootnote The size of a structure is the length of a reasonable encoding of the structure. For example, the size of a n-ary relation with m tuples over a domain with s elements is $O(mn \log s)$. of the structure $\mathbf{D}$, since we only have to store one stage at a time and compute the next stage, while making sure that the current level of iteration has not exceeded 2^{s^n}. Notice also that if the formula $\varphi(x_1, \ldots, x_n, S)$ is inflationary in S, then the noninflationary fixpoint of φ coincides with the inflationary fixpoint of φ, because the sequence of stages converges. It follows that NFP is an extension of IFP. While IFP formulas can be evaluated in

[2] Actually, it is shown in [AV91a] that there is no loss of expressiveness in restricting attention to converging 1st-order operators, i.e., to operators such that $\Phi^{m_0} = \Phi^{m_0+1}$ for some integer m_0.

polynomial time, NFP formulas can be evaluated in polynomial space and can express PSPACE-complete problems, such as the *network convergence* problem [Fed91].

Noninflationary-fixpoint logic was introduced by Abiteboul and Vianu [AV91a] (who called it *partial*-fixpoint logic). In particular, they observed that noninflationary-fixpoint logic coincides with the query language RQL introduced in [CH82] and studied further in [Var82]. It follows that NFP captures the complexity class PSPACE [Var82], but the evenness property is not expressible in NFP [CH82].

As IFP captures PTIME and as NFP captures PSPACE, if IFP and NFP have the same expressive power, then PTIME=PSPACE. It is not clear, however, that the converse holds because IFP and NFP need order to "fully capture" PTIME and PSPACE, respectively. This phenomenon is called the *order mismatch.* Complexity classes are defined in terms of Turing machines. Turing machines always receive ordered inputs, since the encoding of the input always impose a order (e.g., the nodes of an input graph have to be numbered). In contrast, logic operates at a more abstract level, which does not require the imposition of order (consider, as an example, how the transitive-closure query is expressed in IFP). We will return to the order mismatch in the sequel.[3]

2.2 Finite-Variable Infinitary Logic

We consider the infinitary logic $L_{\infty\omega}$, which is the extension of first-order logic that results by allowing infinite disjunctions and conjunctions in the syntax, while keeping the quantifier strings finite (cf. [BF85]). To illustrate the gain in expressive power, recall the evenness property, which as we noted in not expressible in NFP. Let ρ_n be a first-order sentence stating that there are exactly n elements. Then the infinitary sentence $\bigvee_{n=1}^{\infty} \rho_{2n}$ asserts that "there is an even number of elements".

We now define formally the syntax of the infinitary logic $L_{\infty\omega}$.

Definition 2.4: Let σ be a vocabulary consisting of finitely many relational and constant symbols and let $\{v_1, \ldots, v_n, \ldots\}$ be an infinite set of variables. The class $L_{\infty\omega}$ of *infinitary formulas* over σ is the smallest collection of expressions such that

- it contains all first-order fomulas over σ;
- if φ is an infinitary formula, then so is $\neg\varphi$;
- if φ is an infinitary formula and v_i is a variable, then $(\forall v_i)\varphi$ and $(\exists v_i)\varphi$ are also infinitary formulas;
- if Ψ is a set of infinitary formulas, then $\bigvee \Psi$ and $\bigwedge \Psi$ are also infinitary formulas.

[3]It is interesing to note that this mismatch does not arise in Fagin's result that existential second-order logic expresses precisely all NP-complete problem [Fag74], since in that logic it is possible to require the existence of a total order.

∎

The semantics of infinitary formulas is a direct extension of the semantics of first-order logic, with $\bigvee \Psi$ interpreted as a disjunction over all formulas in Ψ and $\bigwedge \Psi$ interpreted as a conjunction.

In the context of finite-model theory, $L_{\infty\omega}$ is too powerful to be of interest, as it can express any class of structure that is closed under isomoprhism (since any structure can be described up to isomoprhism by a first-order sentence). In general, formulas of $L_{\infty\omega}$ may have an infinite number of distinct variables. We now focus attention on fragments of $L_{\infty\omega}$ in which the total number of variables is required to be finite. Variables, however, may have an infinite number of occurrences in such formulas.

Definition 2.5: Let k be a positive integer.

- The *infinitary logic with k variables*, denoted by $L^k_{\infty\omega}$, consists of all formulas of $L_{\infty\omega}$ with at most k distinct variables.
- The infinitary logic $L^\omega_{\infty\omega}$ consists of all formulas of $L_{\infty\omega}$ with a finite number of distinct variables. Thus, $L^\omega_{\infty\omega} = \bigcup_{k=1}^{\infty} L^k_{\infty\omega}$.
- We write $L^k_{\omega\omega}$ for the collection of all first-order formulas with at most k variables.

∎

The family $L^\omega_{\infty\omega}$ of the infinitary languages $L^k_{\infty\omega}$, $k \geq 1$, was introduced first by Barwise [Rub75, Bar77], as a tool for studying positive-fixpoint logic on infinite structures. Since that time, however, these languages have had numerous uses and applications in theoretical computer sciences. Indeed, they underlie much of the work in [Imm82, dR87, LM89, CFI92] and they have also been studied in their own right in [Kol85, KV92a, KV92b, Lyn93, Tys94, DLW95, KV95, LT95, KV96]. See [KV92c] for a survey.

We now give some examples that illustrate the expressive power of infinitary logic with a fixed number of variables.

Example 2.6: Paths and Connectivity
Assume that the vocabulary σ consists of a single binary relation symbol E and let $p_n(x, y)$ be a first-order formula over σ asserting that there is a path of length n from x to y. The obvious way to write $p_n(x, y)$ requires $n + 1$ variables, namely

$$(\exists x_1 \ldots \exists x_{n-1})(E(x, x_1) \wedge E(x_1, x_2) \wedge \ldots \wedge E(x_{n-1}, y)).$$

It is well known, however, that each $p_n(x, y)$ is equivalent to a formula in $L^3_{\omega\omega}$, i.e. a first-order formula with at most three distinct variables x, y, z. To see this, put

$$p_1(x, y) \equiv E(x, y)$$

and assume, by induction on n, that $p_{n-1}(x,y)$ is equivalent to a formula in $L^3_{\omega\omega}$. Then

$$p_n(x,y) \equiv (\exists z)[E(x,z) \wedge (\exists x)(x = z \wedge p_{n-1}(x,y))].$$

It follows that "connectivity" is a property of directed graphs expressible in $L^3_{\infty\omega}$, since it is given by the formula $(\forall x \forall y)(\bigvee_{n=1}^{\infty} p_n(x,y))$. Similarly, the property "there is no cycle" is also in $L^3_{\infty\omega}$, since it is definable by $(\forall x)(\bigwedge_{n=1}^{\infty} \neg p_n(x,x))$. Notice that if P is *any* set of positive integers, then the property "x and y are connected by a path whose length is a number in P" is expressible in $L^3_{\infty\omega}$ via the formula $\bigvee_{n\in P} p_n(x,y)$. It follows that $L^\omega_{\infty\omega}$ can express non-recursive properties. ∎

Properties such as "connectivity" and "there is no cycle" are also known to be expressible in IFP [CH82]. We consider next extensions of first-order logic with fixpoint formation rules and compare the resulting logics to $L^\omega_{\infty\omega}$. A key observation underlying this paper is that IFP and NFP can be viewed as fragments of $L^\omega_{\infty\omega}$.

Theorem 2.7: [KV92a] *Let $\varphi(x_1, \ldots, x_m, S)$ be an $L^k_{\omega\omega}$- formula over a vocabulary $\sigma \cup \{S\}$, where S is an m-ary relation, $m \leq k$. Let Φ be the operator associated with φ. Then, for every $l \geq 1$, the stage $\Phi^l(x_1, \ldots, x_k)$ of Φ is definable by an $L^k_{\omega\omega}$-formula on all structures over σ. As a result, the noninflationary fixpoint $\varphi^\infty(x_1, \ldots, x_k)$ of $\varphi(x_1, \ldots, x_k, S)$ is definable by a formula of $L^k_{\infty\omega}$ on all structures over σ.*

Note, however, that $L^\omega_{\infty\omega}$ is more expressive than NFP, since, as observed above, $L^\omega_{\infty\omega}$ can express non-recursive properties.

3 Applications

One of the reasons for the importance of $L^\omega_{\infty\omega}$ in finite-model theory is the fact that from the expressive power standpoint $L^\omega_{\infty\omega}$ constitutes an extension of fixpoint logics (Theorem 2.7). Thus, $L^\omega_{\infty\omega}$ can be used to derive both positive and negative results about fixpoint logics. On the positive side, structural results about $L^\omega_{\infty\omega}$ transfer to similar results for fixpoint logics, while, on the negative side, lower-bound results for expressibility in $L^\omega_{\infty\omega}$ yield, a fortiori, lower bounds for expressibility in fixpoint logics (but not vice-versa). The main advantage of $L^\omega_{\infty\omega}$ is that the expressive power of the infinitary logics $L^k_{\infty\omega}$ with k variables, $k \geq 1$, has a clean and precise characterization in terms of simple k-pebble games between two players on a pair of structures (cf. [Bar77, Imm82, KV92b]). In contrast, fixpoint logic is not known to possess a similar property.

3.1 $L^k_{\infty\omega}$-Equivalence and Pebble Games

Two structures are *equivalent* in some logic L if no sentence of L distinguishes them. This is a central concept in every logic and plays an important role in model theory. We discuss here equivalence in the infinitary logics with a fixed number of variables.

Definition 3.1: Let $\mathbf{A}$ and $\mathbf{B}$ be two structures over a vocabulary σ and let k be a positive integer.

- We say that $\mathbf{A}$ is $L^k_{\infty\omega}$-*equivalent* to $\mathbf{B}$, and we write $\mathbf{A} \equiv^k_{\infty\omega} \mathbf{B}$, if $\mathbf{A}$ and $\mathbf{B}$ satisfy the same sentences of $L^k_{\infty\omega}$.
- We say that $\mathbf{A}$ is $L^k_{\omega\omega}$-*equivalent* to $\mathbf{B}$, and we write $\mathbf{A} \equiv^k_{\omega\omega} \mathbf{B}$, if $\mathbf{A}$ and $\mathbf{B}$ satisfy the same sentences of $L^k_{\omega\omega}$.

∎

The connection between definability in $L^k_{\infty\omega}$ and the equivalence relation $\equiv^k_{\infty\omega}$ is described by the following two propositions.

Proposition 3.2: [KV92b] *Let σ be a vocabulary, let $\mathcal{F}$ be the class of all structures over σ, and let k be a positive integer. Then every equivalence class of the equivalence relation $\equiv^k_{\infty\omega}$ on $\mathcal{F}$ is definable by a sentence of $L^k_{\infty\omega}$.*

Proposition 3.3: [KV92b] *Let $\mathcal{C}$ be a class of structures over a vocabulary σ and let k be a positive integer. Then the following statements are equivalent:*

1. *The class $\mathcal{C}$ is $L^k_{\infty\omega}$-definable, i.e. there is a sentence φ of $L^k_{\infty\omega}$ such that for any structure $\mathbf{A}$ over σ we have that*

$$\mathbf{A} \in \mathcal{C} \Longleftrightarrow \mathbf{A} \models \varphi.$$

2. *If $\mathbf{A}$ and $\mathbf{B}$ are structures over σ such that $\mathbf{A} \in \mathcal{C}$ and $\mathbf{A} \equiv^k_{\infty\omega} \mathbf{B}$, then $\mathbf{B} \in \mathcal{C}$.*

It is known that $\equiv^k_{\infty\omega}$-equivalence can be characterized in terms of an *infinitary k-pebble game*. This game was implicit in Barwise [Bar77] and was described in detail in Immerman [Imm82].

Definition 3.4: Assume that $\mathbf{A}$ and $\mathbf{B}$ are two structures over the vocabulary σ and let $c_1, \ldots, c_l$ and $d_1, \ldots, d_l$ be the interpretations of the constant symbols of σ on $\mathbf{A}$ and $\mathbf{B}$, respectively, such that $c_i = c_j$ iff $d_i = d_j$ for $1 \leq i < j \leq l$. The *k-pebble game* between two players, known as the *Spoiler* and the *Duplicator*, on the structures $\mathbf{A}$ and $\mathbf{B}$ proceeds in rounds according to the following rules:

In each round the Spoiler chooses one of the two structures $\mathbf{A}$ and $\mathbf{B}$ and places a pebble on one of its elements. The Duplicator responds by placing a pebble on an element of the other structure. The Spoiler chooses again one of the two structures and the game continues this way until k pebbles have been placed on each structure.

Let a_i and b_i, $1 \leq i \leq k$, be the elements of $\mathbf{A}$ and $\mathbf{B}$ respectively picked by the two players in the i-th move. The Spoiler wins the game at this point if one of the following two conditions holds:

- Two of the pebbles are on the same element of one of the structures, while the corresponding two pebbles are on different elements of the other structure, i.e., $a_i = a_j$ and $b_i \neq bj$ (or $a_i \neq a_j$ and $b_i = b_j$), for some i and j such that $1 \leq i < j \leq k$.
- The previous condition fails and the mapping h with

  $$h(a_i) = b_i,\ 1 \leq i \leq k,$$

  and

  $$h(c_j) = d_j,\ 1 \leq j \leq l,$$

  is *not* an isomorphism between the substructures of $\mathbf{A}$ and $\mathbf{B}$ with universes

  $$\{a_1, \ldots, a_k\} \cup \{c_1, \ldots, c_l\}$$

  and

  $$\{b_1, \ldots, b_k\} \cup \{d_1, \ldots, d_l\}$$

  respectively.

If both conditions fail, then the Spoiler removes some corresponding pairs of pebbles and the game proceeds to the next round. ∎

The Duplicator wins the game if he can continue playing "forever", i.e. if the Spoiler can never win a round of the game. The concept of a *winning strategy* for the Duplicator in the k-pebble game is formalized in terms a family of partial isomorphisms; see [KV92b] for details.

The crucial connection between k-pebble games and $L^k_{\infty\omega}$-equivalence is provided by the following result.

Theorem 3.5: [Bar77, Imm82, KV92b] *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two structures over a vocabulary* σ*, and let* k *be a positive integer. The following are equivalent:*

1. $\mathbf{A} \equiv^k_{\omega\omega} \mathbf{B}$.
2. $\mathbf{A} \equiv^k_{\infty\omega} \mathbf{B}$.
3. *The Duplicator has a winning strategy for the* k*-pebble game on* $\mathbf{A}$ *and* $\mathbf{B}$.

We note that the fact that the k-pebble game captures both $L^k_{\infty\omega}$-equivalence and $L^k_{\omega\omega}$-equivalence holds only for finite structures; for general structures the k-pebble game captures $L^k_{\infty\omega}$-equivalence, which differs from $L^k_{\omega\omega}$-equivalence [KV92b].

As a consequence of Proposition 3.3 and Theorem 3.5, we get a game-theoretic characterization of definability in the logics $L^k_{\infty\omega}$, $k \geq 1$.

Proposition 3.6: [KV92b] *Let $\mathcal{C}$ be a class of structures over a vocabulary σ and let k be a positive integer. Then the following statements are equivalent:*

1. *The class $\mathcal{C}$ is $L^k_{\infty\omega}$-definable.*
2. *If $\mathbf{A}$ and $\mathbf{B}$ are structures over σ such that $\mathbf{A} \in \mathcal{C}$ and the Duplicator has a winning strategy for the k-pebble game on $\mathbf{A}$ and $\mathbf{B}$, then $\mathbf{B} \in \mathcal{C}$.*

3.2 Expressiveness

Proposition 3.6 provides a method for establishing that certain properties are not expressible in infinitary logic with a finite number of variables. More specifically, in order to establish that a property Q is not expressible by any formula of $L^\omega_{\infty\omega}$ it is enough to show that for any $k \geq 1$ there are structures $\mathbf{A}_k$ and $\mathbf{B}_k$ such that $\mathbf{A}_k \models Q$, $\mathbf{B}_k \not\models Q$, and the Duplicator has a winning strategy for the k-pebble game on $\mathbf{A}_k$ and $\mathbf{B}_k$. Moreover, Proposition 3.6 guarantees that this method is also *complete*, i.e., if Q is not expressible in $L^k_{\infty\omega}$, then such structures $\mathbf{A}_k$ and $\mathbf{B}_k$ must exist.

As a demonstration of this method we show that the evenness property is not expressible in $L^\omega_{\infty\omega}$.

Proposition 3.7: [KV92b] *The evenness property is not expressible in $L^\omega_{\infty\omega}$.*

Proof sketch: We describe the proof for a vocabulary that consists of one binary predicate symbol E; the proof for arbitrary vocabularies is similar. It suffices to show that the evenness property is not expressible in $L^k_{\infty\omega}$, for arbitrary k. Consider $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$, the complete directed graphs on k nodes and $k+1$ nodes, respectively. Obviously, precisely one of these structures has even cardinality. Note, however, that every l-node substructure of $\mathbf{K}_k$ is isomoprhic to $\mathbf{K}_l$, which is in turn isomorphic to every l-node substructure of $\mathbf{K}_{k+1}$, for each $l \leq k$. Thus, the Duplicator clearly has a winning strategy for the k-pebble game on $\mathbf{K}$ and $\mathbf{K}_{k+1}$. The claim follows. ∎

3.3 0-1 Laws for Infinitary Logics

Let σ be a vocabulary consisting of finitely many relation symbols (but not proposition, function, or constant symbols) and let $\mathcal{C}$ be the class of all structures over σ with universe an initial segment $\{1, 2, \ldots, n\}$ of the positive integers for some $n \geq 1$. If P is a property of (some) structures in $\mathcal{C}$, then the *(labeled) asymptotic probability* $\mu(P)$ on $\mathcal{C}$ is defined to be equal to the limit as $n \to \infty$ of the fraction of structures in $\mathcal{C}$ of cardinality n which satisfy P, provided this limit exists. We say that P *is true almost everywhere* on $\mathcal{C}$ in case $\mu(P)$ is equal to 1. If $\mu(P) = 0$, then we say that P *is false almost everywhere*. It turns out that many interesting properties on the class $\mathcal{G}$ of all graphs are either true almost everywhere or false almost everywhere. It is, for example, well known and easy to prove that μ(connectivity)=1, μ(rigidity)=1, while μ(planarity)=0 and μ(l-colorability)=0, for $l \geq 2$ [Bol79]. On the other hand, evenness does not have an asymptotic probability. The study of asymptotic probabilities can be viewed as the foundations of the theory of average-case complexity, since it focuses on the average-case properties of mathematical structures (cf. [ACV92]).

Fagin [Fag76] and Glebskii et al. [GKLT69] were the first to establish a fascinating connection between logical definability and asymptotic probabilities. More specifically, they showed that if C is the class of all structures over some relational vocabulary and if P is any property expressible in first-order logic, then $\mu(P)$ exists and is either 0 or 1. This result, which is known as *the 0-1 law for first-order logic*, became the starting point of a series of investigations aiming in discovering the relationship between expressibility in a logic and asymptotic probabilities. The survey by Compton [Com88] contains an eloquent account of developments in this area (see also [Gur92] for an expository introduction).

If L is a logic, we say that the 0-1 *law holds for* L in case $\mu(P)$ exists and is equal to 0 or 1 for every property P expressible in the logic L. An important direction of investigation in finite-model theory pursued the study of 0-1 laws for fixpoint extensions of first-order logic. Talanov [Tal81] showed that the 0-1 holds for first-order logic augmented with a *transitive closure* operator. This result was extended by Talanov and Knyazev [TK86], and, independently, by Blass, Gurevich and Kozen [BGK85] who proved that a 0-1 law holds for IFP. The latter results were extended further in [KV87], where a 0-1 law was established for NFP. It turns out that these results are subsumed by a 0-1 law for $L^{\omega}_{\infty\omega}$.

In the past, 0-1 laws for various logics L were proved by establishing first a *transfer theorem for* L of the following kind:

> *There is an infinite structure* $\mathbf{R}$ *over the vocabulary* σ *such that for any property* P *expressible in* L *we have:*
>
> $$\mathbf{R} \textit{ satisfies } P \quad \Longleftrightarrow \quad \mu(P) = 1 \textit{ on } \mathcal{C}.$$

This method was discovered by Fagin [Fag76] in his proof of the 0-1 law for first-order logic. It was also used later in [BGK85] to establish the 0-1 law for IFP and in [KV87] to show that the 0-1 law holds for NFP.

It turns out that there is a countable structure $\mathbf{R}$ over the vocabulary σ that satisfies the above equivalence for all these logics. Moreover, this structure $\mathbf{R}$ is unique up to isomorphism. We call $\mathbf{R}$ the *countable random structure over the vocabulary* σ. The random structure $\mathbf{R}$ is characterized by an infinite set of *extension axioms*, which, intuitively, assert that every *type* can be *extended* to any other possible type. The precise definitions are as follows.

Definition 3.8: Let σ be a vocabulary consisting of relation symbols only.

- If $\mathbf{x} = (x_1, ..., x_m)$ is a sequence of distinct variables, then a *type* $t(\mathbf{x})$ *in the variables* $\mathbf{x}$ *over* σ is the conjunction of all the formulas in a *maximally consistent* set S of equalities $x_i = x_j$, inequalities $x_i \neq x_j$, atomic formulas in the variables $\mathbf{x}$, and negated atomic formulas in the variables $\mathbf{x}$.

- Let z be a variable that is different from all the variables in $\mathbf{x}$. We say that *a type* $t(\mathbf{x}, z)$ *extends the type* $s(\mathbf{x})$ if every conjunct of $s(\mathbf{x})$ is also a conjunct of $t(\mathbf{x}, z)$.

- With each pair $s(\mathbf{x})$ and $t(\mathbf{x}, z)$ of types such that t extends s we associate a first-order *extension axiom* $\tau_{s,t}$ stating that

$$(\forall \mathbf{x})(s(\mathbf{x}) \rightarrow (\exists z) t(\mathbf{x}, z)).$$

∎

Let T be the set of all extension axioms. The theory T was studied by Gaifman [Gai64], who showed, using a *back and forth* argument, that any two countable models of T are isomorphic (*T is an ω-categorical theory*). Fagin [Fag76] realized that the extension axioms are relevant to the study of asymptotic probabilities and proved that on the class $\mathcal{C}$ of all structures over a finite vocabulary σ

$$\mu(\tau_{s,t}) = 1$$

for any extension axiom $\tau_{s,t}$. The equivalence between truth on $\mathbf{R}$ and almost sure truth on $\mathcal{C}$ (and consequently the 0-1 law for first-order logic) follows from these two results by an application of the Compactness Theorem of mathematical logic.

We now show that the 0-1 law holds for the infinitary logic $L^{\omega}_{\infty\omega}$ using the game-theoretic characterization of $L^{\omega}_{\infty\omega}$. Unlike the proofs that are based on the transfer property, our proof here does note employ any "infinitistic" methods; the proofs does not assume the 0-1 law for first-order logic, it does not involve the random structure $\mathbf{R}$ or any other infinite structure, and it does not make use of compactness[4] or of any of its consequences. Moreover, the 0-1 law is derived directly without establishing a transfer theorem first. The 0-1 law for $L^{\omega}_{\infty\omega}$ on the one hand subsumes all earlier ones in [BGK85, Fag76, KV87, Tal81, TK86], and on the other hand provides a unifying treatment of 0-1 laws for first-order logic and its extensions with fixpoint operators or infinitary syntax.

Let k be a fixed positive integer. If $\mathbf{A}$ is a structure, then we write $[\mathbf{A}]$ for *the equivalence class* of $\mathbf{A}$ with respect to the equivalence relation $\equiv^{k}_{\infty\omega}$. In what follows we will show that there is a tight connection between 0-1 laws and the asymptotic probabilities of equivalence classes $[\mathbf{A}]$. Actually, this turns out to be a general fact that holds for arbitrary probability measures.

So far all the results discussed here are about the *uniform* probability measures on $\mathcal{C}$, i.e., all structures with n elements carry the same probability. There is, however, a well

[4]Note that the Compactness Theorem is known to fail for finite structures.

developed study of random structures under *variable* probability measures. This started with the work of Erdös and Rényi [ER60] and is presented in detail in Bollobás [Bol85]. In general, for each $n \geq 1$ one has a probability measure pr_n on all structures in $\mathcal{C}$ with n elements, where pr_n may be a non-uniform distribution. The *asymptotic probability* $pr(P)$ of a property P (*relative to the probability measures* pr_n, $n \geq 1$) is defined by $pr(P) = \lim_{n\to\infty} pr_n(P)$, provided this limit exists. Note that the asymptotic probability μ defined earlier for the uniform measure is simply a special case of this definition. If L is a logic, then we say that a *0-1 law holds for L relative to the measure pr* if for every sentence ψ of L the asymptotic probability $pr(\psi)$ exists and is either 0 or 1. Notice that, strictly speaking, pr is not a probability measure, because it is not countably additive (it is, however, finitely additive).

We now describe necessary and sufficient condition for the existence of 0-1 laws for $L^k_{\infty\omega}$ under arbitrary probability measures.

Theorem 3.9: [KV92b] *Let K be a class of structures over a vocabulary σ, let k be a positive integer, and let pr_n, $n \geq 1$, be a sequence of probability measures on the structures in K with n elements. Then the following are equivalent:*

1. *The 0-1 law holds for the infinitary logic $L^k_{\infty\omega}$ relative to the measure pr.*
2. *There is an equivalence class C of the equivalence relation $\equiv^k_{\infty\omega}$ such that $pr(C) = 1$.*

We now return to the uniform measure μ on $L^k_{\infty\omega}$ and give a proof of the 0-1 law for $L^k_{\infty\omega}$ using the preceding Theorem 3.9 and the characterization of $\equiv^k_{\infty\omega}$ in terms of pebble games.

Theorem 3.10: [KV92b] *Let $\mathcal{C}$ be the class of all structures over a vocabulary σ, let k be a positive integer, and let θ_k be the conjunction of all extension axioms with at most k variables. If $\mathbf{A}$ is a structure that is a model of θ_k, then $\mu([\mathbf{A}]) = 1$. As a result, the 0-1 law holds for $L^k_{\infty\omega}$ relative to the uniform measure on $\mathcal{C}$.*

Proof Sketch: If $\mathbf{A}$ and $\mathbf{B}$ are both models of θ_k, then it is easy to verify that the Duplicator has a winning strategy in the k-pebble game on $\mathbf{A}$ and $\mathbf{B}$. Intuitively, the winning strategy for the Duplicator is provided by the elements of $\mathbf{A}$ and $\mathbf{B}$ witnessing the extension axioms with at most k variables.

It follows from Theorem 3.5 that if $\mathbf{A}$ and $\mathbf{B}$ are both models of θ_k, then $\mathbf{A} \equiv^k_{\infty\omega} \mathbf{B}$. Since $\mu(\theta_k) = 1$ [Fag76], it follows that $\mu([\mathbf{A}]) = 1$, for any structure $\mathbf{A}$ that is a model of θ_k. The 0-1 law now follows by Theorem 3.9. ∎

Corollary 3.11: *Let $\mathcal{C}$ be the class of all structures over a vocabulary σ. The 0-1 law holds for IFP and NFP relative to the uniform measure on $\mathcal{C}$.*

4 Effective Fragments of $L^{\omega}_{\infty\omega}$

While $L^{\omega}_{\infty\omega}$ is an elegant theoretical tool, its formulas have a non-effective syntax, and can define non-computable queries. Thus, $L^{\omega}_{\infty\omega}$ is not practical as a query language in the context of databases. The purpose of this section is identify natural *effective* fragments of $L^{\omega}_{\infty\omega}$, and relate them to known computational models for queries.

4.1 Relational Machines

The language consisting of IFP and NFP queries can be viewed as a programming language with simple control interacting with the databases via a set of first-order queries. A natural extension of this paradigm is to consider computationally complete computing devices interacting with the database via given first-order queries [AV95].

A relational machine is a Turing machine augmented with a *relational store*. The relational store consists of a set of relations of certain arities. Some of these relations are designated as input relations and some are designated as output relations. The *type* of a relational machine is the sequence of arities of its relations. The *arity* of the machine is the maximal arity of the relations in its store. For example, the relational store may consists of two unary relations and one binary relation, in which case the type is $\langle 1, 1, 2\rangle$ and the arity is 2. The tape of the machine is a work tape and is initially empty. Transitions depend on the current state, the content of the current tape cell, and a test of emptiness of a relation of the relational store. Transitions involve the following actions: move right or left on the tape, overwrite the current tape cell with some tape symbol, and replace a relation of the store with the result of an first-order operation on relations of the store. The first-order operations are: boolean operations $(\cap, \cup, -)$, $\pi_{i_1 \ldots i_m}$ (project the tuples in a relation on the coordinates $i_1 \ldots i_m$ in the specified order), $\times$ (cross-product of two relations), and $\sigma_{i=j}$, $\sigma_{i=a}$ (resp., select from relation tuples whose i-th coordinate coincides with j-th one, or select those tuples whose i-th coordinate coincides with domain element a), see [Ull89]. For example, the machine can have instructions such as:

> If the machine is in state s_3, the head is reading the symbol 1, and relation R_1 is empty, then change the state to s_4, replace the symbol 1 by 0, move the head to the right, and replace R_2 by $R_2 \cap R_3$.

Relational machines model the embedding of a relational database query language [Cod72] in a general purpose programming language. They are essentially the *loosely coupled generic machines* (GM^{loose}), introduced in [AV91b]. Unlike Turing machines, which get their input spread on the input tape, relational machines get their input in a more natural way. For example, if the input is a graph, then the input to the machine is the set of nodes (a unary relation) and the set of edges (a binary relation). Thus, the order mismatch does not come up between relational machines and fixpoint logics. We will come back later to the connection between relational machines and fixpoint logics.

Relational machines give rise to three types of computational devices. First, we can think of a relational machine as an acceptor of a *relational language*, i.e., a set of structures. In that case, we assume that there is a single 0-ary output relation; the machine accepts if the output relation consists of the 0-ary tuple and rejects if the output relation is empty. We can also think of a relational machine as a *relational function*, computing an output structure for a given input structure. Finally, we can think of relational machine as a *mixed function*, i.e., a function from structures to strings, where the output is written on the machine's tape.

Since relational machines have a Turing machine component, they can perform arbitrarily complex computations on the tape. Thus, relational machines are Turing complete[5] on ordered inputs, since they can first compute an encoding of the ordered input on the tape and then proceed to emulate any standard Turing machine. Relational machines are not Turing complete, however, in general. Indeed, as we shall see, they cannot compute evenness. This wide variation in expressive power generalizes the situation for the IFP and NFP languages and has similar causes. Like IFP and NFP, a relational machine interacts with the databases via a finite set of first-order queries, which limits its ability to distinguish among tuples in the input. Essentially, relational machine can perform *any* computation on equivalence classes of tuples it can distinguish. We will return to this point shortly.

4.2 Infinitary Logic and Relational Machines

As noted, neither the syntax nor the semantics of $L^{\omega}_{\infty\omega}$ are effective. To isolate natural effective fragments of $L^{\omega}_{\infty\omega}$, we consider both the syntactic and the semantic aspects. This leads to two notions of recursively enumerable (r.e.) fragments of $L^{\omega}_{\infty\omega}$:

- Syntactic: one can effectively enumerate the syntactic "components" of the formula.
- Semantic: the set of models of the formula is r.e.

As we shall see, the two approaches do not lead to the same classes of formulas (nor the same sets of models).

We start with the syntactic approach. Consider the syntax tree of a given $L^{\omega}_{\infty\omega}$-formula φ. By definition, each path in the tree is finite. Suppose that each node in the tree has countably many children. Then each position in the tree is identified by a finite sequence $i_1 \ldots i_k$ of integers. The notion of recursive enumeration of the components of the formula is therefore captured by:

Definition 4.1: A formula φ in $L^{\omega}_{\infty\omega}$ is *r.e.* if:

1. each of its disjunctions and conjunctions is countable; and

[5] I.e., they have the same expressive power as Turing machines.

2. there exists a computable function which, given a finite sequence of integers $i_1 ... i_k$, returns the symbol in the position denoted by $i_1 ... i_k$ in the syntax tree of φ.

∎

Remark 4.2: The above definition is the analogue of the definition of the Church-Kleene fragment $L^{CK}_{\omega_1\omega}$, which is the syntactically effective fragment of $L_{\omega_1\omega}$; see [Bar75]. ∎

It turns out that syntactic effectiveness does not guarantee semantic effectiveness. Indeed, we show that the r.e. formulas yield the analog of the arithmetic hierarchy defined using relational machines.

The arithmetic hierarchy based on relational machines is defined in analogy to the classical arithmetic hierarchy [Rog67]. More precisely, add to relational machines oracles for sets of structures accepted by relational machines to obtain the first level of the hierarchy; next, add oracles on the set of structures that can thereby be accepted; and so on. Let Σ^{rel} denote the collection of sets of structures in this hierarchy.

Proposition 4.3: [AVV95] *$S = models(\varphi)$ for some r.e. sentence φ in $L^{\omega}_{\infty\omega}$ iff S is in Σ^{rel}.*

It is easily seen that Σ^{rel} coincides on ordered inputs with the classical arithmetic hierarchy. Thus, r.e. $L^{\omega}_{\infty\omega}$-formulas can define non-computable queries. We therefore try to attack the problem of finding effective fragments of $L^{\omega}_{\infty\omega}$ using the semantic approach. The semantic notion of an effective $L^{\omega}_{\infty\omega}$-formula simply requires that the set of models of the formula be r.e.

Definition 4.4: A set S of structures over a vocabulary σ is *recursively enumerable* (r.e.) if there exists a recursive enumeration $I_1, ..., I_i, ...$ of structures over σ such that I is in S iff some structure isomorphic to I belongs to the enumeration. ∎

We next characterize the $L^{\omega}_{\infty\omega}$ sentences whose set of models is r.e. The result shows that these are semantically equivalent to relational machines. This also yields a normal form for such sentences. To prove this result, we use a normal form for $L^{\omega}_{\infty\omega}$ in the style of [AV95, DLW95]. The normal form says essentially that any $L^{\omega}_{\infty\omega}$-query can be reduced by a relational machine to an $L^{\omega}_{\infty\omega}$-query over ordered structures.

A structure is *ordered* if it contains a binary relation *succ* consisting of a successor function over the domain of the structure. The normal form reduces query evaluation on arbitrary structures to query evaluation on ordered structures. In the next lemma, the fact that the resulting structure is ordered is used to encode the structure on the tape of a relational machine.

Lemma 4.5: [AVV95] *For vocabulary σ and an $L^{\omega}_{\infty\omega}$ formula φ over σ, there exists a vocabulary $\sigma_<$ and relational machine M_{order} such that:*

1. *given a structure I over σ, the machine M_{order} computes on the tape an encoding of an ordered structure $I_<$ over $\sigma_<$;*

2. *there exists an $L^{\omega}_{\infty\omega}$ formula ψ over $\sigma_<$ such that for each structure I over σ we have that $I \models \varphi$ iff $I_< \models \psi$.*

We now have:

Theorem 4.6: [AVV95] *Let S be a set of structures over a vocabulary σ. The following are equivalent:*

1. *S is accepted by some relational machine;*

2. *$S = models(\varphi)$ for some sentence $\varphi \in L^{\omega}_{\infty\omega}$ and S is r.e.;*

Proof Sketch: Suppose that S is accepted by some relational machine M of arity k. Clearly, S is r.e., since M can be simulated by a standard Turing machine. It can be shown that each accepting computation of M can be described by an $L^k_{\omega\omega}$-formula. Now let A be the set of $L^k_{\omega\omega}$-formulas describing accepting computations of M, i.e.,

$$A = \{\varphi_\alpha \mid \alpha \text{ is an accepting computation of } M\}.$$

Since an input is accepted by M iff it yields some accepting computation of M, we have $S = models(\bigvee A)$.

Suppose that $S = models(\varphi)$ is r.e., where $\varphi \in L^{\omega}_{\infty\omega}$. Since φ is in $L^{\omega}_{\infty\omega}$, we can use Lemma 4.5. Note that, since S is r.e., $M_{order}(S)$ is also r.e. A relational machine M accepting S can be constructed as follows. Given input I, the machine M first computes $M_{order}(I)$ on the tape. Next, M enumerates the encodings of instances in $M_{order}(S)$ on the tape, which is possible since $M_{order}(S)$ is r.e. M accepts I if $M_{order}(I)$ is generated. Let us verify that M accepts precisely S. Suppose I is accepted by M. Then $M_{order}(I)$ is in $M_{order}(S)$, so $M_{order}(I) = M_{order}(J)$ for some J satisfying φ. Also, $I_<$ and $J_<$ are isomorphic. By Lemma 4.5, $J_<$ satisfies ψ, so $I_<$ also satisfies ψ. Finally, again by Lemma 4.5, I satisfies φ, so I is in S. Conversely, suppose I is in S. Then $M_{order}(I)$ occurs in $M_{order}(S)$, so M accepts I. ∎

Thus, relational machines can be viewed as a syntactic and semantic effective fragment of $L^{\omega}_{\infty\omega}$.

4.3 Relational Complexity

Now that we have identified relational machines as an effective fragment of $L^{\omega}_{\infty\omega}$, we would like to define complexity classes using relational machines. As noted in [AV95], using as measure of the input its size is not appropriate for relational machines. This is because, unlike Turing machines, relational machines have limited access to their input, and in particular cannot generally compute its size. Indeed, relational machines are less "discerning" than Turing machines. We say that a relational machine M cannot *discern* between k-tuples $\mathbf{u}$ and $\mathbf{v}$ over an input $\mathbf{D}$, if for each k-ary relation R in its store and throughout the computation of M over $\mathbf{D}$, we have that $\mathbf{u}$ is in R precisely when $\mathbf{v}$ is in R. For example, if there is an automorphism on $\mathbf{D}$ that maps $\mathbf{u}$ to $\mathbf{v}$, then $\mathbf{u}$ and $\mathbf{v}$ cannot be discerned by M over $\mathbf{D}$. The discerning power of relational machines is best understood in terms of k-pebble games.

The k-pebble game between the Spoiler and the Duplicator on the l-tuples ($l \leq k$) $\mathbf{u}$ and $\mathbf{v}$ of a structure $\mathbf{D}$ proceeds as described above on the pair $(\mathbf{D}, \mathbf{D})$ of structures, with the following initial rule. The game starts with the Spoiler choosing one of the tuples $\mathbf{u}$ or $\mathbf{v}$, placing l pebbles on the elements of the chosen tuple (i.e., on $u_1, \ldots, u_l$ or $v_1, \ldots, v_l$), and placing the other pebbles on chosen elements of $\mathbf{D}$. The Duplicator responds by placing l pebbles on the elements of the other tuple, and placing the other pebbles on chosen elements of $\mathbf{D}$.

If the Duplicator wins the k-pebble game on the tuple $\mathbf{u}$ and $\mathbf{v}$ of a structure $\mathbf{D}$, then we say that $\mathbf{u}$ and $\mathbf{v}$ are k-equivalent, denoted $\mathbf{u} \equiv_k \mathbf{v}$, over $\mathbf{D}$. The relation $\equiv_k$ characterizes the discerning power of k-ary relational machines.

Proposition 4.7: [AVV97] *The tuples* $\mathbf{u}$ *and* $\mathbf{v}$ *are k-equivalent over a structure* $\mathbf{D}$ *if and only if no k-ary relational machine M can discern between* $\mathbf{u}$ *and* $\mathbf{v}$ *over* $\mathbf{D}$.

Since k-ary relational machines cannot discern among k-equivalent tuples, they cannot measure the size of their input. Intuitively, all that a k-ary relational machine "sees" in the input is the k-equivalence classes of tuples. This is best understood by looking at the extremes. As shown in [AV95], relational machines are Turing complete on ordered inputs (where all distinct tuples can be distinguished from each other), but they collapse to first-order logic on unordered sets (where the number of k-equivalence classes is bounded by a constant independent of the size of the input). Since all a k-ary relational machine "sees" in the input is the k-equivalence classes of tuples, it seems natural to measure the input size with respect to $\equiv_k$. Let the *k-size* of a structure $\mathbf{D}$, denoted $size_k(\mathbf{D})$, be the number of $\equiv_k$-classes of k-tuples over $\mathbf{D}$. Thus, is is reasonable to measure the time or space complexity of k-ary relational machines as a function of the k-size of their input. This measure, however, can reasonably serve as a basis for measuring complexity only if it can be calculated by relational machines. We first state an important technical result.

Lemma 4.8: [AV95, DLW95] *For each input type σ and $k > 0$ there exists an IFP formula φ over σ, with $2k$ free variables $x_1, \ldots, x_k, y_1, \ldots, y_k$ such that for each input* $\mathbf{D}$ *of type σ, the formula φ defines a partial order $\prec$ over k-tuples, and $x_1, \ldots, x_k$ is incomparable (via $\prec$) with $y_1, \ldots, y_k$ iff $x_1, \ldots, x_k \equiv_k y_1, \ldots, y_k$.*

In other words, there exists an IFP query computing an order on the equivalence classes of $\equiv_k$. Once we have an order, counting the number of classes is easy.

Proposition 4.9: [AVV97] *For each $k > 0$ and input type σ, there is a relational machine that outputs on its tape, for an input structure $\mathbf{D}$ of type σ, a string of length $size_k(\mathbf{D})$ in time polynomial in $size_k(\mathbf{D})$.*

From now on we measure the complexity of k-ary relational machines in terms of the k-size of their input. We can now define relational complexity classes in terms of deterministic or nondeterministic relational time or relational space, e.g., $\mathrm{DTIME}_r(f(n))$, $\mathrm{NSPACE}_r(f(n))$, and so on. Thus, we can define the relational complexity class PTIME_r (relational polynomial time), $\mathrm{NPSPACE}_r$, etc. Proposition 4.9 guarantees the effective enumerability of these classes. It is not obvious whether known relationships between deterministic, nondeterministic, and alternating complexity classes, such as

$$\mathrm{PSPACE}=\mathrm{NPSPACE}=\mathrm{APTIME},$$

hold also for relational complexity classes, since these relationships are typically the result of simulations that use order (although we show below that they do hold)

To simplify references to complexity classes, we use the notation *Class*(*Resource*, *Control*, *Bound*) and $Class_r$(*Resource*, *Control*, *Bound*), where *Resource* can be time or space, *Control* can be deterministic, nondeterministic, or alternating, and *Bound* is the bounding function or family of functions. Thus, $Class(time, nondeterministic, poly)$ is NP, and $Class_r(space, deterministic, poly)$ is PSPACE_r. We will always assume that our bounds are at least linear. Typically, *Bound* will be a *polynomially closed* set of functions, i.e., a set of functions that contains the linear functions and contains $p(f(n))$ whenever it contains $f(n)$, for all polynomials $p(x)$. Furthermore, we assume that all bounds are fully time and space constructible (see [HU79]). Note that most commonly used functions, including the logarithmic, linear, polynomial and exponential bounds, are fully time and space constructible; also, the fully time and space constructible functions are polynomially closed.

What is the relationship between relational and standard complexity? It is easy to see that the k-size of a structure $\mathbf{D}$ is always bounded above by a polynomial ($O(n^k)$) in the size of $\mathbf{D}$, for all $k > 0$. Furthermore, relational machines are strictly weaker than Turing machines because they cannot check that the cardinality of their input is even. The latter follows from Theorems 3.7 and 4.6.

Proposition 4.10: [AVV97] *Let Φ be a polynomially closed set of functions. Then*

$$Class_r(resource, control, \Phi) \subset Class(resource, control, \Phi),$$

for any resource and control.

While relational machines are in some sense weaker than standard machines, the weakness is due only to the lack of order. This weakness disappears in the presence of order. Let w be string, say over the alphabet $\{0, 1\}$, of length n. We can encode w

by a structure $rel(w)$ consisting of a total order on the elements $\{0, \ldots, n-1\}$ and a unary predicate on these elements, giving the positions of 1's in the string. Note than the k-size of $rel(w)$ is bounded by n^k. For a language L, let $rel(L) = \{rel(w) \mid w \in L\}$. Since $rel(w)$ is ordered for each w, a relational machine can simulate a machine for L.

Proposition 4.11: [AVV97] *For each resource, control, polynomially closed bound and a language L in $Class(resource, control, bound)$, the relational language $rel(L)$ is in $Class_r(resource, control, bound)$.*

Combining Propositions 4.10 and 4.11, we get:

Corollary 4.12: [AVV97] *Let Φ_1 and Φ_2 be polynomially closed sets of functions and let $resource_1$, $resource_2$, $control_1$, $control_2$ be resources and controls, respectively. Then we have that*

$$Class(resource_1, control_1, \Phi_1) \subseteq Class(resource_2, control_2, \Phi_2)$$

if

$$Class_r(resource_1, control_1, \Phi_1) \subseteq Class_r(resource_2, control_2, \Phi_2).$$

Proof: Suppose

$$Class_r(resource_1, control_1, \Phi_1) \subseteq Class_r(resource_2, control_2, \Phi_2).$$

Let L be in $Class(resource_1, control_1, \Phi_1)$. By Proposition 4.11, $rel(L)$ is in the relational class $Class_r(resource_1, control_1, \Phi_1)$, so $rel(L)$ is also in $Class_r(resource_2, control_2, \Phi_2)$. By Proposition 4.10,

$$Class_r(resource_2, control_2, \Phi_2) \subset Class(resource_2, control_2, \Phi_2).$$

It follows that there exists a Turing machine $M_{rel(L)}$ that accepts a standard encoding of $rel(w)$ with control $control_2$ and with the resource $resource_2$ bounded by some function in Φ_2 iff $w \in L$. Since a standard encoding of $rel(w)$ can obviously be obtained from w in time/space polynomial in $|w|$, it follows that there exists a Turing machine M_L which accepts L, with the same control and resource bound as $M_{rel(L)}$. Thus, $Class(resource_2, control_2, \Phi_2)$ contains L. ∎

It follows from Corollary 4.12 that separation results among standard complexity classes translate into separation results among relational complexity classes. For example, it follows that PTIME_r is strictly contained in EXPTIME_r.

To further understand the relationship between standard and relational complexity, we introduce the notion of *reduction* from a relational language to a standard language. We say that a relational language L of type σ is *relationally reducible in polynomial time* to a standard language L' if there exists a deterministic polynomial-time relational machine T acting as a mixed function from σ-structures to strings, such that for each structure $\mathbf{D}$ of type σ we have that $\mathbf{D} \in L$ if and only if $T(\mathbf{D}) \in L'$. It turns out that every relational language can be reduced to a standard language. This reduction uses the partial order constructed in Lemma 4.8.

Theorem 4.13: [AVV97] *Let Φ be a polynomially closed set of fully time/space constructible functions and let L be a relational language in $Class_r(\rho, \kappa, \Phi)$ for some resource ρ, control κ, and bound Φ. Then L is relationally reducible in polynomial time to a language L' in $Class(\rho, \kappa, \Phi)$.*

According to the proof of the theorem, the reduction from L to L' depends only on the type σ of L and the arity k of its relational acceptor M. We denote the relational machine that does the reduction for type σ and arity k by $T_{\sigma,k}$, and call it the *relational reduction machine* of input type σ and arity k.

Combining Propositions 4.10 and 4.11, Corollary 4.12, and Theorem 4.13, we get that the relationships among relational complexity classes are analogous to the relationships among standard complexity classes.

Corollary 4.14: [AVV97] *Let Φ_1 and Φ_2 be polynomially closed sets of fully time/space constructible functions and let $resource_1$, $resource_2$, $control_1$, $control_2$ be kinds of resources and controls, respectively. Then*

$$Class(resource_1, control_1, \Phi_1) \subseteq Class(resource_2, control_2, \Phi_2)$$

if and only if

$$Class_r(resource_1, control_1, \Phi_1) \subseteq Class_r(resource_2, control_2, \Phi_2).$$

Proof: The if part is Corollary 4.12. For the only-if part, suppose

$$Class(resource_1, control_1, \Phi_1) \subseteq Class(resource_2, control_2, \Phi_2)$$

and let L be a relational language of type σ in $Class_r(resource_1, control_1, \Phi_1)$, accepted by some relational machine of arity k.

Consider the relational reduction machine $T_{\sigma,k}$. By Theorem 4.13, there exists a language L' in $Class(resource_1, control_1, \Phi_1)$ such that for each $\mathbf{D}$ of type σ, $\mathbf{D} \in L$ iff $T_{\sigma,k}(\mathbf{D}) \in L'$. Since L' is in $Class(resource_1, control_1, \Phi_1)$, it is in $Class(resource_2, control_2, \Phi_2)$. Let M' be a relational machine that, on input $\mathbf{D}$ computes $T_{\sigma,k}(\mathbf{D})$ (in time polynomial in $size_k(\mathbf{D})$), then runs on $T_{\sigma,k}(\mathbf{D})$ the Turing machine that accepts L' with control $control_2$ using resource $resource_2$ bounded by some function in Φ_2. Then M' accepts L, uses control $control_2$ and resource $resource_2$ bounded by some function in Φ_2. Thus, L is in $Class_r(resource_2, control_2, \Phi_2)$. ∎

In particular, it follows from Corollary 4.14, that the known relationships between deterministic, nondeterministic, and alternating complexity classes, such as

$$\text{PSPACE=NPSPACE=APTIME},$$

do hold for relational complexity classes, i.e. PSPACE$_r$=NPSPACE$_r$=APTIME$_r$. Also, the open questions about standard complexity classes translate to questions about relational complexity classes, e.g., PTIME=PSPACE if and only if PTIME$_r$=PSPACE$_r$.

4.4 Relational Machines and Fixpoint Logics

Fixpoint logics involve iterations of 1st-order formulas. Since relational algebra has the expressive power of 1st-order logic, it is clear that relational machines with the appropriate control can simulate all fixpoint logics. For example, IFP$\subseteq$ PTIME$_r$ and NFP$\subseteq$ PSPACE$_r$. To find the precise relationship between fixpoint logics and relational machines consider again Theorem 4.13. According to that theorem, every relational language of a certain relational complexity can be reduced in relational polynomial time to a standard language of the analogous standard complexity. For example, a relational language in PTIME$_r$ can be reduced in relational polynomial time to a language in PTIME. As discussed earlier, IFP captures PTIME. Thus, the only gap now between PTIME$_r$ and IFP is the relational polynomial time reduction. This gap is now bridged by the following theorem, which uses the normal-form theorem of [AV95].

Theorem 4.15: *Let $T_{\sigma,k}$ be the relational reduction machine of input type σ and arity k. There is an IFP formula $\varphi_{\sigma,k}$ such that $\varphi_{\sigma,k}(\mathbf{D}) = rel(T_{\sigma,k}(\mathbf{D}))$.*

Theorem 4.15 supplies the missing link between relational machines and fixpoint logics. Combining this with Theorem 4.13 we get:

Theorem 4.16: [AV95]

1. *IFP=PTIME$_r$*
2. *NFP=PSPACE$_r$*

Proof: To simulate a relational machine by a fixpoint logic, one first uses $\varphi_{\sigma,k}$ to obtain $rel(T_{\sigma,k}(\mathbf{D}))$, and then uses the logic to simulate a standard machine. This is possible because $rel(T_{\sigma,k}(\mathbf{D}))$ is an ordered structure and because the fixpoint logics express the corresponding standard complexity classes on ordered structures. ∎

Theorem 4.16 should be contrasted with our earlier discussion on the complexity of IFP and NFP. In that discussion, we talked about IFP and NFP *capturing* complexity classes, Theorem 4.16 provides a *precise* characterization of the expressive power of fixpoint logics in terms of relational complexity classes. We can now use relational complexity as a mediator between fixpoint logic and standard complexity. Corollary 4.14 relates standard complexity to relational complexity, while Theorem 4.16 relates relational complexity to fixpoint logic. Together, they bridge the gap between standard complexity and fixpoint logic.

Corollary 4.17: [AV95] *P = PSPACE iff IFP=NFP*

Thus, one of the most tantalizing questions in complexity theory boil down to one fundamental issue: the relative power of inflationary vs. noninflationary 1st-order operators. As is shown in [AVV97], the questions of NP vs. PSPACE and PSPACE vs. EXPTIME can also be expressed in terms of the relative power of inflationary vs. noninflationary 1st-order operators.

Acknowledgements. I am grateful to Ron Fagin for numerous comments on a previous draft of this article.

References

[ACV92] S. Abiteboul, K. Compton, and V. Vianu. Queries are easier than you thought (probably). In *Proc. 11th ACM Symp. on Principles of Database Systems*, pages 23–32, 1992.

[ACY95] F. Afrati, S.S. Cosmadakis, and M. Yannakakis. On Datalog vs. polynomial time. *J. of Computer and System Sciences*, 51:177–196, 1995.

[Acz77] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 739–782. North Holland, 1977.

[AU79] A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Proc. 6th ACM Symp. on Principles of Programming Languages*, pages 110–117, 1979.

[AV91a] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43:62–124, 1991.

[AV91b] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 209–219, 1991.

[AV95] S. Abiteboul and V. Vianu. Computing with first-order logic. *Journal of Computer and System Sciences*, 50(2):309–335, April 1995.

[AVV95] S. Abiteboul, M.Y. Vardi, and V. Vianu. Computing with infinitary logic. *Theoretical Computer Science*, 149:101–128, 1995.

[AVV97] S. Abiteboul, M.Y. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. *J. ACM*, 44:30–56, 1997.

[Bar75] J. Barwise. *Admissible Sets and Structures*. Springer-Verlag, 1975.

[Bar77] J. Barwise. On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.

[BF85] J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Springer-Verlag, 1985.

[BGK85] A. Blass, Y. Gurevich, and D. Kozen. A zero–one law for logic with a fixed point operator. *Information and Control*, 67:70–90, 1985.

[Bol79] B. Bollobas. *Graph Theory*. Springer-Verlag, 1979.

[Bol85] B. Bollobas. *Random Graphs*. Academic Press, 1985.

[CFI92] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.

[CH82] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[Cha88] A. Chandra. Theory of database queries. In *Proc. 7th ACM Symp. on Principles of Database Systems*, pages 1–9, 1988.

[Cod70] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13:377–387, 1970.

[Cod72] E. F. Codd. Relational completeness of data base sublanguages. In R. Rustin, editor, *Database Systems*, pages 33–64. Prentice-Hall, 1972.

[Com88] K. J. Compton. 0-1 laws in logic and combinatorics. In I. Rival, editor, *NATO Adv. Study Inst. on Algorithms and Order*, pages 353–383. D. Reidel, 1988.

[DLW95] A. Dawar, S. Lindell, and S. Weinstein. Infinitary logic and inductive definability over finite structures. *Information and Computation*, 119:160–175, 1995.

[dR87] M. de Rougemont. Second-order and inductive definability on finite structures. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 33:47–63, 1987.

[ER60] P. Erdös and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 7:17–61, 1960.

[Fag74] R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, pages 43–73, 1974.

[Fag75] R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[Fag76] R. Fagin. Probabilities on finite models. *Journal of Symbolic Logic*, 41:50–58, 1976.

[Fag93] R. Fagin. Finite-model theory—a personal perspective. *Theoretical Computer Science*, 116(1):3–31, 1993.

[Fed91] T. Feder. *Stable networks and product graphs*. PhD thesis, Stanford University, 1991.

[Gai64] H. Gaifman. Concerning measures in first-order calculi. *Israel Journal of Mathematics*, 2:1–18, 1964.

[Gai82] H. Gaifman. On local and nonlocal properties. In J. Stern, editor, *Logic Colloquium '81*, pages 105–135. North Holland, 1982.

[GKLT69] Y. V. Glebskii, D. I. Kogan, M. I. Liogonki, and V. A. Talanov. Range and degree of realizability of formulas in the restricted predicate calculus. *Cybernetics*, 5:142–154, 1969.

[GS86] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[Gur92] Y. Gurevich. Zero-one laws: The logic in computer science column. *Bulletin of the European Association for Theoretical Computer Science*, 1992.

[GV85] H. Gaifman and M. Y. Vardi. A simple proof that connectivity is not first-order. *Bulletin of the European Association for Theoretical Computer Science*, 26:43–45, June 1985.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Imm82] N. Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[Imm89] N. Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Computational Complexity Theory, Proc. Symp. Applied Math., Vol. 38*, pages 75–91. American Mathematical Society, 1989.

[Kol85] Ph. G. Kolaitis. On asymptotic probabilities of inductive queries and their decision problem. In R. Parikh, editor, *Logics of Programs '85, Lecture Notes in Computer Science 193*, pages 153–166. Springer-Verlag, 1985.

[KV87] Ph. G. Kolaitis and M. Y. Vardi. The decision problem for the probabilities of higher-order properties. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 425–435, 1987.

[KV92a] Ph. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *Proc. 6th IEEE Symp. on Logic in Comp. Sci.*, pages 46–57, 1992.

[KV92b] Ph. G. Kolaitis and M. Y. Vardi. Infinitary logic and 0-1 laws. *Information and Computation*, 98:258–294, 1992.

[KV92c] Ph. G. Kolaitis and M. Y. Vardi. Infinitary logic for computer science. In *Proc. 19th International Colloq. on Automata, Languages, and Programming*, pages 450–473. Springer-Verlag, Lecture Notes in Computer Science 623, 1992.

[KV95] Ph. G. Kolaitis and M. Y. Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, August 1995.

[KV96] Ph. G. Kolaitis and M. Y. Vardi. On the expressive power of variable-confined logics. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 348–359, 1996.

[Lei90] D. Leivant. Inductive definitions over finite structures. *Information and Computation*, 89:95–108, 1990.

[LM89] V. S. Lakshmanan and A. O. Mendelzon. Inductive pebble games and the expressive power of DATALOG. In *Proc. 8th ACM Symposium on Principles of Database Systems*, pages 301–310, 1989.

[LT95] J. Lynch and J. Tyszkiewicz. The infinitary logic of sparse random graphs. In *Proc. 10th IEEE Symp. on Logic in Comp. Sci.*, pages 46–53, 1995.

[Lyn93] J. Lynch. Infinitary logics and very sparse random graphs. In *Proc. 8th IEEE Symp. on Logic in Computer Science*, pages 191–198, 1993.

[Pap94] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Rog67] H. Rogers. *Theory of recursive functions and effective computability.* McGraw-Hill, 1967.

[Rub75] A. Rubin. *Free Algebras in von Neumann-Bernays-Godel Set Theory and Positive Elementary Inductions in Reasonable Structures.* PhD thesis, California Institute of Technology, 1975.

[Tal81] V. A. Talanov. Asymptotic solvability of logical formulas. *Combinatorial-Algebraic Methods in Applied Mathematics*, pages 118–126, 1981. MR 85i:03081, ZBL 538.03007.

[TK86] V. A. Talanov and V. V. Knyazev. The asymptotic truth of infinite formulas. In *Proceedings of All-Union Seminar on Discrete and Applied Mathematics and it Applications*, pages 56–61, 1986. MR 89g:03054.

[Tys94] J. Tyszkiewicz. Infinitary queries and their asymptotic probabilities II: properties definable in least fixed point logic. *Random Structures & Algorithms*, 5:215–234, 1994.

[Ull89] J. D. Ullman. *Database and Knowledge-Base Systems, Volumes I and II.* Computer Science Press, 1989.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 137–146, 1982.