Towards a Theory of Recursive Structures*

David Harel**

Dept. of Applied Mathematics and Computer Science
The Weizmann Institute of Science, Rehovot, Israel
harel@wisdom.weizmann.ac.il

Abstract. In computer science, one is interested mainly in finite objects. Insofar as infinite objects are of interest, they must be computable, i.e., recursive, thus admitting an effective finite representation. This leads to the notion of a recursive graph, or, more generally, a recursive structure, model or data base. This paper summarizes recent work on recursive structures and data bases, including (i) the high undecidability of many problems on recursive graphs and structures, (ii) a method for deducing results on the descriptive complexity of finitary NP optimization problems from results on the computational complexity (i.e., the degree of undecidability) of their infinitary analogues, (iii) completeness results for query languages on recursive data bases, (iv) correspondences between descriptive and computational complexity over recursive structures, and (v) zero-one laws for recursive structures.

1 Introduction

This paper provides a summary of work — most of it joint with Tirza Hirst — on infinite recursive (i.e., computable) structures and data bases, and attempts to put it in perspective. The work itself is contained in four papers [H,HH1,HH2,HH3], which are summarized, respectively, in Sections 2, 3, 4 and 5.

When computer scientists become interested in an infinite object, they require it to be computable, i.e., recursive, so that it possesses an effective finite representation. Given the prominence of finite graphs in computer science, and the many results and open questions surrounding them, it is very natural to investigate recursive graphs too. Moreover, insight into finite objects can often be gleaned from results about infinite recursive variants thereof. An infinite recursive graph can be thought of simply as a recursive binary relation over the natural numbers. Recursive graphs can be represented by the (finite) algorithms, or Turing machines, that recognize their edge sets, so that it makes sense to investigate the complexity of problems concerning them.

* Preliminary versions of this paper appeared in STACS '94, *Proc. 11th Ann. Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Vol. 775, Springer-Verlag, Berlin, 1994, pp. 633–645, and in *Computer Science Today*, Lecture Notes in Computer Science, Vol. 1000, Springer-Verlag, 1995, pp. 374–391.

** Incumbent of the William Sussman Chair of Mathematics.

Luboš Prim et al. (Eds.): MFCS'98, LNCS 1450, pp. 36–53, 1998.

Indeed, a significant amount of work has been carried out in recent years regarding the complexity of problems on recursive graphs. Some of the first papers were written in the 1970s by Manaster and Rosenstein [MR] and Bean [B1,B2]. Following that, a variety of problems were considered, including ones that are NP-complete for finite graphs, such as k-colorability and Hamiltonicity [B1,B2,BG2,Bu,GL,MR] and ones that are in P in the finite case, such as Eulerian paths [B2,BG1] In most cases (including the above examples) the problems turned out to be undecidable. This is true even for highly recursive graphs [B1], i.e., ones for which node degree is finite and the set of neighbors of a node is computable. Beigel and Gasarch [BG1] and Gasarch and Lockwood [GL] investigated the precise level of undecidability of many such problems, and showed that they reside on low levels of the arithmetical hierarchy. For example, detecting the existence of an Eulerian path is Π_3^0-complete for recursive graphs and Π_2^0-complete for highly recursive graphs [BG1].

The case of Hamiltonian paths seemed to be more elusive. In 1976, Bean [B2] had shown that the problem is undecidable (even for plana graphs), but the precise characterization was not known. In response to this question, posed by R. Beigel and B. Gasarch, the author was able to show that Hamiltonicity is in fact *highly* undecidable, *viz,* Σ_1^1-complete. The result, proved in [H] and summarized in Section 2, holds even for highly recursive graphs with degree bounded by 3. (It actually holds for planar graphs too.) Hamiltonicity is thus an example of an interesting graph problem that becomes highly undecidable in the infinite case.[1]

The question then arises as to what makes some NP-complete problems highly undecidable in the infinite case, while others (e.g., k-colorability) remain on low levels of the arithmetical hierarchy. This was the starting point of the joint work with T. Hirst. In [HH1], summarized in Section 3, we provide a general definition of infinite recursive versions of NP optimization problems, in such a way that MAX CLIQUE, for example, becomes the question of whether a recursive graph contains an infinite clique. Two main results are proved in [HH1], one enables using knowledge about the infinite case to yield implications to the finite case, and the other enables implications in the other direction. The results establish a connection between the descriptive complexity of (finitary) NP optimization problems, particularly the syntactic class MAX NP, and the computational complexity of their infinite versions, particularly the class Σ_1^1. Taken together, the two results yield many new problems whose infinite versions are highly undecidable and whose finite versions are outside MAX NP. Examples include MAX CLIQUE, MAX INDEPENDENT SET, MAX SUBGRAPH, and MAX TILING.

The next paper, [HH2], summarized in Section 4, puts forward the idea of infinite recursive relational data bases. Such a data base can be defined simply as a finite tuple of recursive relations (not necessarily binary) over some countable domain. We thus obtain a natural generalization of the notion of a finite relational data base. This is not an entirely wild idea: tables of trigonometric

[1] Independent work in [AMS] showed that perfect matching is another such problem.

functions, for example, can be viewed as a recursive data base, since we might be interested in the sines or cosines of infinitely many angles. Instead of keeping them all in a table, which is impossible, we keep rules for computing the values from the angles, and vice versa, which is really just to say that we have an effective way of telling whether an edge is present between nodes i and j in an infinite graph, and this is precisely the notion of a recursive graph.

In [HH2], we investigate the class of computable queries over recursive data bases, the motivation being borrowed from [CH1]. Since the set of computable queries on such data bases is not closed under even simple relational operations, one must either make do with a very humble class of queries or considerably restrict the class of allowed data bases. The main parts of [HH2] are concerned with the completeness of two query languages, one for each of these possibilities. The first is quantifier-free first-order logic, which is shown to be complete for the non-restricted case. The second is an appropriately modified version of the complete language QL of [CH1], which is proved complete for the case of "highly symmetric" data bases. These have the property that their set of automorphisms is of finite index for each tuple-width.

While the previous topic involves languages for *computable* queries, our final paper, [HH3], summarized in Section 5, deals with languages that express *non*-computable queries. In the spirit of results for finite structures by Fagin, Immerman and others, we sought to connect the computational complexity of properties of recursive structures with their descriptive complexity, i.e, to capture levels of undecidability syntactically as the properties expressible in various logical formalisms. We consider several formalisms, such as first-order logic, second-order logic and fixpoint logic. One of our results is analogous to that of Fagin [F1]; it states that, for any $k \geq 2$, the properties of recursive structures expressible by Σ^1_k formulas are exactly the generic properties in the complexity class Σ^1_k of the analytical hierarchy.

[HH3] also deals with zero-one laws. It is not too difficult to see that many of the classical theorems of logic that hold for general structures (e.g., compactness and completeness) fail not only for finite models but for recursive ones too. Others, such as Ehrenfeucht–Fraisse games, hold for finite and recursive structures too. Zero-one laws, to the effect that certain properties (such as those expressible in first-order logic) are either almost surely true or almost surely false, are considered unique to finite model theory, since they require counting the number of structures of a given finite size. We introduce a way of extending the definition of these laws to recursive structures, and prove that they hold for first-order logic, strict Σ^1_1 and strict Π^1_1. We then use this fact to show non-expressibility of certain properties of recursive structures in these logics.

While recursive structures and models have been investigated quite widely by logicians (see, e.g., [NR]), the kind of issues that computer scientists are interested in have not been addressed prior to the work mentioned above. We feel that that this is a fertile area for research, and raises theoretical and practical questions concerning the computability and complexity of properties of recursive structures, and the theory of queries and update operations over recursive data

bases. We hope that the work summarized here will stimulate more research on these topics.

2 Hamiltonicity in Recursive Graphs

A *recursive directed graph* is a pair $G = (V, E)$, where V is recursively isomorphic to the set of natural numbers $\mathcal{N}$, and $E \subset V \times V$ is recursive. G is *undirected* if E is symmetric. A *highly recursive graph* is a recursive graph for which there is a recursive function H from V to finite subsets of V, such that $H(v) = \{u \mid \langle v, u\rangle \in E\}$.

A *one-way* (respectively, *two-way*) *Hamiltonian path* in G is a 1-1 mapping p of $\mathcal{N}$ (respectively, $\mathcal{Z}$) onto V, such that $\langle p(x), p(x+1)\rangle \in E$ for all x.

Bean [B2] showed that determining Hamiltonicity in highly recursive graphs is undecidable. His reduction is from non-well-foundedness of recursive trees with finite degree, which can be viewed simply as the halting problem for (nondeterministic) Turing machines. Given such a tree T, the proof in [B2] constructs a graph G, such that infinite paths in T map to Hamiltonian paths in G. The idea is to make the nodes of G correspond to those of T, but with all nodes that are on the same level being connected in a cyclic fashion. In this way, a Hamiltonian path in G simulates moving down an infinite path in T, but at each level it also cycles through all nodes on that level. A fact that is crucial to this construction is the finiteness of T's degree, so that the proof does not generalize to trees with infinite degree, Thus, Bean's proof only establishes that Hamiltonicity is hard for Π_1^0, or co-r.e.

In [H] we have been able to show that the problem is actually Σ_1^1-complete. Hardness is proved by a reduction (that is elementary but not straightforward) from the non-well-foundedness of recursive trees with possibly *infinite* degree, which is well-known to be a Σ_1^1-complete problem [R]:

Theorem: *Detecting (one-way or two-way) Hamiltonicity in a (directed or undirected) highly recursive graph is Σ_1^1-complete, even for graphs with $H(v) \leq 3$ for all v.*

Proof sketch: In Σ_1^1 is easy: With the $\exists f$ quantifying over total functions from $\mathcal{N}$ to $\mathcal{N}$, we write

$$\exists f\ \forall x\ \forall y\ \exists z\ (\langle f(x), f(x+1)\rangle \in E \wedge (x \neq y \rightarrow f(x) \neq f(y)) \wedge f(z) = x).$$

This covers the case of one-way paths. The two-way case is similar.

We now show Σ_1^1-hardness for undirected recursive graphs with one-way paths. (The other cases require more work, especially in removing the infinite branching from the graphs we construct in order to obtain the result for highly recursive graphs. The details can be found in [H].)

Assume a recursive tree T is given, with nodes $\mathcal{N} = 0, 1, 2, 3, \ldots$, and root 0, and whose *parent-of* function is recursive. T can be of infinite degree. We construct an undirected graph G, which has a one-way Hamiltonian path iff T has an infinite path.

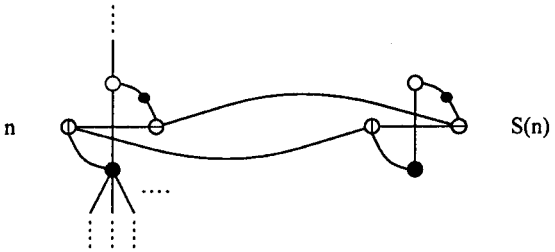

Figure 1

For each element $n \in \mathcal{N}$, G has a cluster of five internal nodes, n^u, n^d, n^r, n^l and n^{ur}, standing, respectively, for *up, down, right, left* and *up-right*. For each such cluster, G has five internal edges:

$$n^l \text{ —— } n^d \text{ —— } n^u \text{ —— } n^{ur} \text{ —— } n^r \text{ —— } n^l$$

For each edge $n \longrightarrow m$ of the tree T, n^d —— m^u is an edge of G. For each node n in T, let $S(n)$ be n's distance from the root in T (its *level*). Since $S(n) \in \mathcal{N}$, we may view $S(n)$ as a node in T. In fact, in G we will think of $S(n)$ as being n's *shadow node*, and the two are connected as follows (see Fig. 1):[2]

$$n^r \text{ —— } S(n)^r \quad \text{and} \quad S(n)^l \text{ —— } n^l$$

To complete the construction, there is one additional root node g in G, with an edge g —— 0^u.

Since T is a recursive tree and S, as a function, is recursive in T, it is easy to see that G is a recursive graph. To complete the proof, we show that T has an infinite path from 0 iff G has a Hamiltonian path.

(*Only-if*) Suppose T has an infinite path p. A Hamiltonian path p' in G starts at the root g, and moves down G's versions of the nodes in p, taking detours to the right to visit n's shadow node $S(n)$ whenever $S(n) \notin p$. The way this is done can be seen in Fig. 2. Since p is infinite, we will eventually reach a node of any desired level in T, so that any $n \notin p$ will eventually show up as a shadow of some node along p and will be visited in due time. It is then easy to see that p' is Hamiltonian.

[2] Clearly, given T, the function $S : \mathcal{N} \to \mathcal{N}$ is not necessarily one-one. In fact, Fig. 1 is somewhat misleading, since there may be infinitely many nodes with the same shadow, so that the degree of both up-nodes and down-nodes can be infinite. Moreover, $S(n)$ itself is a node somewhere else in the tree, and hence has its own T-edges, perhaps infinitely many of them.

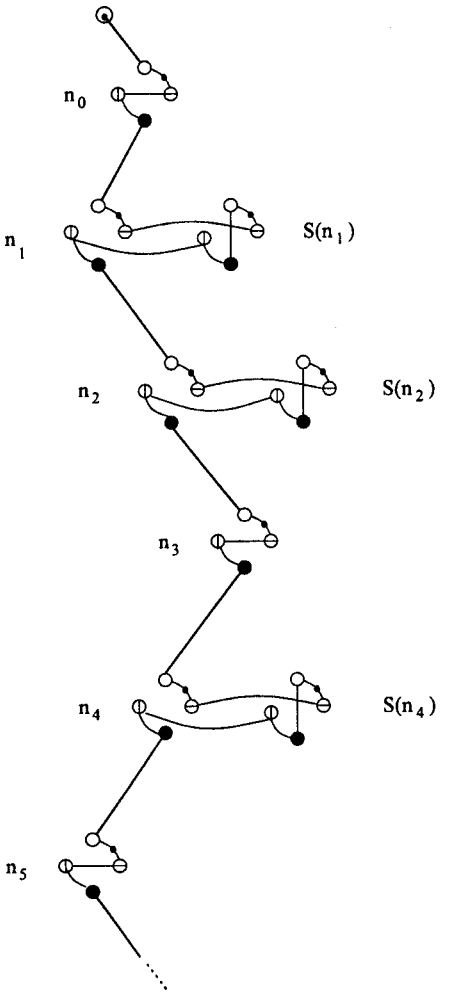

Figure 2

(*If*) Suppose G has a Hamiltonian path p. It helps to view the path p as containing not only the nodes, but also the edges connecting them. Thus, with the exception of the root g, each node in G must contribute to p exactly two incident edges, one incoming and one outgoing.

We now claim that for any n, if p contains the T-edge incident to the up-node n^u, or, when $n = 0$, if it contains the edge between g and 0^u, then it must also contain a T-edge incident to the down node n^d.

To see why this is true, assume p contains the T-edge incident to n^u (this is the edge leading upwards at the top left of Fig. 1). Consider n^{ur} (the small black node in the figure). It has exactly two incident edges, both of which must therefore be in p. But since one of them connects it to n^u, we already have in p the two required edges for n^u, so that the one between n^u and n^d cannot be in p. Now, the only remaining edges incident to n^d are the internal one connecting

it to n^l, and its T-edges, if any. However, since p must contain exactly two edges incident to n^d, one of them must be one of the T-edges. $\triangle$

In fact, Hamiltonicity is Σ_1^1-complete even for planar graphs [HH1].

3 From the Finite to the Infinite and Back

Our approach to optimization problems focuses on their descriptive complexity, an idea that started with Fagin's [F1] characterization of NP in terms of definability in existential second-order logic on finite structures. Fagin's theorem asserts that a collection C of finite structures is NP-computable if and only if there is a quantifier-free formula $\psi(\overline{x}, \overline{y}, S)$, such that for any finite structure A:

$$A \in C \Leftrightarrow A \models (\exists S)(\forall \overline{x})(\exists \overline{y})\psi(\overline{x}, \overline{y}, S).$$

Papadimitriou and Yannakakis [PY] introduced the class MAX NP of maximization problems that can be defined by

$$\max_S |\{\overline{x}\colon\ A \models (\exists \overline{y})\psi(\overline{x}, \overline{y}, S)\}|,$$

for quantifier-free ψ. MAX SAT is the canonical example of a problem in MAX NP. The authors of [PY] also considered the subclass MAX SNP of MAX NP, consisting of those maximization problems in which the existential quantifier above is not needed. (Actually, the classes MAX NP and MAX SNP of [PY] contain also their closures under L-reductions, which preserve polynomial-time approximation schemes. To avoid confusion, we use the names MAX Σ_0 and MAX Σ_1, introduced in [KT], rather than MAX SNP and MAX NP, for the 'pure' syntactic classes.)

Kolaitis and Thakur [KT] then examined the class of all maximization problems whose optimum is definable using first-order formulas, i.e., by

$$\max_S |\{\overline{w}\colon\ A \models \psi(\overline{w}, S)\}|,$$

where $\psi(\overline{w}, S)$ is an arbitrary first-order formula. They first showed that this class coincides with the collection of polynomially-bounded NP-maximization problems on finite structures, i.e., those problems whose optimum value is bounded by a polynomial in the input size. They then proved that these problems form a proper hierarchy, with exactly four levels:

$$\text{MAX } \Sigma_0 \subset \text{MAX } \Sigma_1 \subset \text{MAX } \Pi_1 \subset \text{MAX } \Pi_2 = \bigcup_{i \geq 2} \text{MAX } \Pi_i$$

Here, MAX Π_1 is defined just like MAX Σ_1 (i.e., MAX NP), but with a universal quantifier, and MAX Π_2 uses a universal followed by an existential quantifier, and corresponds to Fagin's general result stated above. The three containments are known to be strict. For example, MAX CLIQUE is in MAX Π_1 but not in MAX Σ_1.

We now define a little more precisely the class of optimization problems we deal with[3]:

Definition: (See [PR]) An NPM problem is a tuple $F = (\mathcal{I}_F, S_F, m_F)$, where

- $\mathcal{I}_F$, the set of *input instances*, consists of finite structures over some vocabulary σ, and is recognizable in polynomial time.
- $S_F(I)$ is the space of *feasible solutions* on input $I \in \mathcal{I}_F$. The only requirement on S_F is that there exists a polynomial q and a polynomial time computable predicate p, both depending only on F, such that $\forall I \in \mathcal{I}_F$, $S_F(I) = \{S\colon |S| \leq q(|I|) \wedge p(I,S)\}$.
- $m_F\colon \mathcal{I}_F \times \Sigma^* \to \mathcal{N}$, the *objective function*, is a polynomial time computable function. $m_F(I,S)$ is defined only when $S \in S_F(I)$.
- The following decision problem is required to be in NP: Given $I \in \mathcal{I}_F$ and an integer k, is there a feasible solution $S \in S_F(I)$, such that $m_F(I,S) \geq k$?

This definition (with an additional technical restriction that we omit here; see [HH1]) is broad enough to encompass most known optimization problems arising in the theory of NP-completeness.

We now define infinitary versions of NPM problems, by evaluating them over infinite recursive structures and asking about the existence of an infinite solution:

Definition: For an NPM problem $F = (\mathcal{I}_F, S_F, m_F)$, let $F^\infty = (\mathcal{I}_F^\infty, S_F^\infty, m_F^\infty)$ be defined as follows:

- $\mathcal{I}_F^\infty$ is the set of *input instances*, which are infinite recursive structures over the vocabulary σ.
- $S_F^\infty(I^\infty)$ is the set of *feasible solutions* on input $I^\infty \in \mathcal{I}_F^\infty$.
- $m_F^\infty\colon \mathcal{I}^\infty \times S_F \to \mathcal{N} \cup \{\infty\}$ is the *objective function*, satisfying

$$\forall I^\infty \in \mathcal{I}_F^\infty, \forall S \in S_F^\infty(I^\infty)\ (m_F^\infty(I^\infty, S) = |\{\overline{x}\colon\ \psi_F(I^\infty, S, \overline{x})\}|).$$

- The decision problem is: Given $I^\infty \in \mathcal{I}_F^\infty$, does there exist $S \in S_F^\infty(I^\infty)$, such that $m_F^\infty(I^\infty, S) = \infty$? Put another way:

$$F^\infty(I^\infty) = \text{TRUE} \quad \text{iff} \quad \exists S(|\{\overline{x}\colon \dot{\psi}_F(I^\infty, S, \overline{x})\}| = \infty).$$

Due to the conditions on NPM problems, F^∞ can be shown not to depend on the Π_2-formula representing m_F. This is important, since, if some finite problem F could be defined by two different formulas ψ_1 and ψ_2 that satisfy the condition but yield different infinite problems, we could construct a finite structure for which ψ_1 and ψ_2 determine different solutions.

Here is the first main result of [HH1]:

Theorem: If $F \in$ MAX Σ_1 then $F^\infty \in \Pi_2^0$.

[3] We concentrate here on maximization problems, though the results can be proved for appropriate minimization ones too.

A special case of this is:

Corollary: For any NPM problem F, if F^∞ is Σ_1^1-hard then F is not in MAX Σ_1.

It follows that since the infinite version of Hamiltonicity is Σ_1^1-complete and thus completely outside the arithmetical hierarchy, an appropriately defined finitary version cannot be in MAX Σ_1. Obviously, the corollary is valid not only for such problems but for all problems that are above Π_2^0 in the arithmetical hierarchy. For example, since detecting the existence of an Eulerian path in a recursive graph is Π_3^0-complete [BG1], its finite variant cannot be in MAX Σ_1 either.

In order to be able to state the second main result of [HH1], we define a special kind of *monotonic* reduction between finitary NPM problems, an *M-reduction*:

Definition: Let $\mathcal{A}$ and $\mathcal{B}$ be sets of structures. A function f: $\mathcal{A} \to \mathcal{B}$ is *monotonic* if $\forall A, B \in \mathcal{A}$ $(A \leq B \Rightarrow f(A) \leq f(B))$. (Here, $\leq$ denotes the substructure relation.) Given two NPM problems: $F = (\mathcal{I}_F, S_F, m_F)$ and $G = (\mathcal{I}_G, S_G, m_G)$, an *M-reduction* g from F to G is a tuple $g = (t_1, t_2, t_3)$, such that:

- $t_1 : \mathcal{I}_F \to \mathcal{I}_G$, $t_2 : \mathcal{I}_F \times S_F \to S_G$, and $t_3 : \mathcal{I}_G \times S_G \to S_F$, are all monotonic, polynomial time computable functions..
- m_F and m_G grow monotonically with respect to t_1, t_2 and t_3 (see [HH1] for a more precise formulation).

We denote the existence of an M-reduction from F to G by $F \propto_M G$. The second main result of [HH1] shows that M-reductions preserve the Σ_1^1-hardness of the corresponding infinitary problems:

Theorem: Let F and G be two NPM problems, with $F \propto_M G$. If F^∞ is Σ_1^1-hard, then G^∞ is Σ_1^1-hard too.

The final part of [HH1] applies these two results to many examples of NPM problems, some of which we now list with their infinitary versions. It is shown in [HH1] that for each of these the infinitary version is Σ_1^1-complete. Mostly, this is done by establishing monotonic reductions on the finite level, and applying the second theorem above. From the first theorem it then follows that the finitary versions must be outside MAX Σ_1.

Here are some of the examples:

1. MAX CLIQUE: I is an undirected graph, $G = (V, E)$.

$$S(G) = \{Y\colon\ Y \subseteq V,\ \forall y, z \in Y\ y \neq z \Rightarrow (y, z) \in E\}$$
$$m(G, Y) = |Y|$$

The maximization version is:

$$\max_{Y \subseteq V} |\{x\colon\ x \in Y \wedge \forall y, z \in Y\ y \neq z \Rightarrow (y, z) \in E\}|$$

MAX CLIQUE$^\infty$: I^∞ is a recursive graph G. Does G contain an infinite clique?

2. MAX IND SET: I is an undirected graph $G = (V, E)$.

$$S(G) = \{Y\colon\ Y \subseteq V,\ \forall y, z \in Y\ (y, z) \notin E\}$$
$$m(G, Y) = |Y|$$
$$\max_{Y \subseteq V} |\{x\colon\ x \in Y\ \wedge\ \forall y, z \in Y\ (y, z) \notin E\}|$$

MAX IND SET$^\infty$: I^∞ is a recursive graph G. Does G contain an infinite independent set?

3. MAX SET PACKING: I is a collection C of finite sets, represented by pairs (i, j), where the set i contains j.

$$S(C) = \{Y \subseteq C\colon\ \forall A, B \in Y\ A \neq B \Rightarrow A \cap B = \emptyset\}$$
$$m(C, Y) = |Y|$$

MAX SET PACKING$^\infty$: I^∞ is a recursive collection of infinite sets C. Does C contains infinitely many disjoint sets?

4. MAX SUBGRAPH: I is a pair of graphs, $G = (V_1, E_1)$ and $H = (V_2, E_2)$, with $V_2 = \{v_1, \ldots, v_n\}$.

$$S(G, H) = \{Y\colon\ Y \subseteq V_1 \times V_2,\ \forall (u, v), (x, y) \in Y,\ u \neq x \wedge v \neq y\ \wedge\ (u, x) \in E_1 \Leftrightarrow (v, y) \in E_2\}$$
$$m((G, H), Y) = k \text{ iff } v_1, \ldots, v_k \text{ appear in } Y, \text{ but } v_{k+1} \text{ does not appear in } Y.$$

MAX SUBGRAPH$^\infty$: I^∞ is a pair of recursive graphs, H and G. Is H a subgraph of G?

5. MAX TILING: I is a grid D of size $n \times n$, and a set of tiles $T = \{t_1, \ldots, t_m\}$. (We assume the reader is familiar with the rules of tiling problems.)

$$S(D, T) = \{Y\colon\ Y \text{ is a legal tiling of some portion of } D \text{with tiles from } T\}$$
$$m(\{D, T\}, Y) = k \text{ iff } Y \text{ contains a tiling of a full } k \times k \text{ subgrid of } D.$$

MAX TILING$^\infty$: I^∞ is a recursive set of tiles T.
Q: Can T tile the positive quadrant of the infinite integer grid?

We thus establish closely related facts about the level of undecidability of many infinitary problems and the descriptive complexity of their finitary counterparts. More examples appear in [HH1].

Two additional graph problems of interest are mentioned in [HH1], planarity and graph isomorphism. The problem of detecting whether a recursive graph is planar can be shown to be co-r.e. Determining whether two recursive graphs

are isomorphic is arithmetical for graphs that have finite degree and contain only finitely many connected components. More precisely, this problem is in Π_1^0 for highly recursive trees; in Π_3^0 for recursive trees with finite degree; in Σ_2^0 for highly recursive graphs; and in Σ_4^0 for recursive graphs with finite degree. As to the isomorphism problem for general recursive graphs, Morozov [Mo] has recently proved, using different techniques, that the problem is Σ_1^1-complete.

4 Completeness for Recursive Data Bases

It is easy to see that recursive relations are not closed under some of the simplest accepted relational operators. For example, if $R(x, y, z)$ means that the yth Turing machine halts on input z after x steps (a primitive-recursive relation), then the projection of R on columns 2 and 3 is the nonrecursive halting predicate. This means that even very simple queries, when applied to general recursive relations, do not preserve computability. Thus, a naive definition of a recursive data base as a finite set of recursive relations will cause many extremely simple queries to be non-computable.

This difficulty can be overcome in essentially two ways (and possibly other intermediate ways that we haven't investigated). The first is to accept the situation as is; that is, to resign ourselves to the fact that on recursive data bases the class of computable queries will necessarily be very humble, and then to try to capture that class in a (correspondingly humble) complete query language. The second is to restrict the data bases, so that the standard kinds of queries *will* preserve computability, and then to try to establish a reasonable completeness result for these restricted inputs. The first case will give rise to a rich class of data bases but a poor class of queries, and the second to a rich class of queries but a poor class of data bases. In both cases, of course, in addition to being Turing computable, the queries will also have to satisfy the consistency criterion of [CH1], more recently termed *genericity*, whereby queries must preserve isomorphisms.

The first result of [HH2] shows that the class of computable queries on recursive data bases is indeed extremely poor. First we need some preparation.

Definition: Let D be a countable set, and let $R_1, \ldots, R_k$, for $k > 0$, be relations, such that for all $1 \leq i \leq k$, $R_i \subseteq D^{a_i}$. $B = (D, R_1, \ldots, R_k)$ is a *recursive relational data base* (or an r-db for short) *of type* $a = (a_1, \ldots a_k)$, if each R_i, considered as a set of tuples, is recursive.

Definition: Let $B_1 = (D_1, R_1, \ldots, R_k)$ and $B_2 = (D_2, R'_1, \ldots, R'_k)$ be two r-db's of the same type, and let $u \in D_1^n$ and $v \in D_2^n$, for some n. Then (B_1, u) and (B_2, v) are *isomorphic*, written $(B_1, u) \cong (B_2, v)$, if there is an isomorphism between B_1 and B_2 taking u to v. (B_1, u) and (B_2, v) are *locally isomorphic*, written $(B_1, u) \cong_l (B_2, v)$, if the restriction of B_1 to the elements of u and the restriction of B_2 to the elements of v are isomorphic.

Definition: An *r-query* Q (i.e., a partial function yielding, for each r-db B of type a, an output (if any) which is a recursive relation over $D(B)$) is *generic*,

if it preserves isomorphisms; i.e. for all B_1, B_2, u, v, if $(B_1, u) \cong (B_2, v)$ then $u \in Q(B_1)$ iff $v \in Q(B_2)$. It is *locally generic* if it preserves local isomorphisms; i.e., for all B_1, B_2, u, v, if $(B_1, u) \cong_l (B_2, v)$ then $u \in Q(B_1)$ iff $v \in Q(B_2)$.

The following is a key lemma in the first result:

Lemma: If Q is a recursive r-query, then Q is generic iff Q is locally generic.

Definition: A query language is *r-complete* if it expresses precisely the class of recursive generic r-queries.

Theorem: The language of first-order logic without quantifiers is r-complete.

We now prepare for the second result of [HH2], which insists on the full set of computable queries of [CH1], but drastically reduces the allowed data bases in order to achieve completeness.

Definition: Let $B = (D, R_1, \ldots, R_k)$ be a fixed r-db. For each $u, v \in D^n$, u and v are *equivalent*, written $u \cong_B v$, if $(B, u) \cong (B, v)$. B is *highly symmetric* if for each $n > 0$, the relation $\cong_B$ induces only a finite number of equivalence classes of rank n.

Highly symmetric graphs consist of a finite or infinite number of connected components, where each component is highly symmetric, and there are only finitely many pairwise non-isomorphic components. In a highly symmetric graph, the finite degrees, the distances between points and the lengths of the induced paths are bounded. A grid or an infinite straight line, for instance, are not highly symmetric, but the full infinite clique is highly symmetric. Fig. 3 shows an example of another highly symmetric graph.

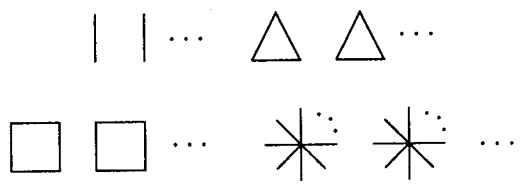

Figure 3

A *characteristic tree* for B is defined as follows. Its root is Λ, and the rest of the vertices are labeled with elements from D, such that the labels along each path from the root form a tuple that is a representative of an equivalence class of $\cong_B$. The whole tree covers representatives of all such classes. No two paths are allowed to form representatives of the same class. We represent a highly symmetric data base B by a tuple

$$C_B = (T_B, \cong_B, C_1, \ldots, C_k),$$

where T_B is some characteristic tree for B, and each C_i is a finite set of representatives of the equivalence classes constituting the relation R_i. We also require that $\cong_B$ be recursive, and that T_B be highly recursive (in the sense of Section 2).

We say that a query Q on a highly symmetric data base is *recursive* if the following version of it, which is applied to the representation C_B rather than to the data base B itself, is partial recursive: whenever $Q(C_B)$ is defined, it yields a finite set of representatives of the equivalence classes representing the relation $Q(B)$.

We now describe the query language QL_s. Its syntax is like that of the QL language of Chandra and Harel [CH1], with the following addition: the test in a **while** loop can be for whether a relation has a single representative, and not only for a relation's emptiness. The semantics of QL_s is the same as the semantics of QL, except for some minor technical adaptations that are omitted here. As in [CH1], the result of applying a program P to C_B is undefined if P does not halt; otherwise it is the contents of some fixed variable, say X_1.

Definition: A query language is *hs-r-complete* if it expresses precisely the class of recursive generic queries over highly symmetric recursive data bases.

Theorem: QL_s is hs-r-complete.

The proof follows four main steps, which are analogous to those given in the completeness proof for QL in [CH1]. The details, however, are more intricate.

In [HH2] a number of additional issues are considered, including the restriction of recursive data bases to finite/co-finite recursive relations, completeness of the generic machines of [AV], and BP-completeness.

5 Expressibility vs. Complexity, and Zero-One Laws

One part of [HH3] proves results that relate the expressive power of various logics over recursive structures to the computational complexity (i.e., the level of undecidability) of the properties expressible therein. We summarize some of these, without providing all of the relevant definitions. In the previous section, we mentioned the result from [HH2] to the effect that the very restricted language of quantifier-free first-order relational calculus is r-complete; i.e., it expresses precisely the recursive and generic r-queries. Here we deal with languages that have stronger expressive power, and hence express also non-recursive queries.

There are many results over *finite* structures that characterize complexity classes in terms of logic. One of the most important of these is Fagin's theorem [F1], mentioned in section 2 above, which establishes that the properties of finite structures expressible by Σ_1^1 formulas are exactly the ones that are in NP. This kind of correspondence also holds between each level of the quantifier hierarchy of second-order logic and the properties computable in the corresponding level of the polynomial-time hierarchy.

In order to talk about recursive structures it is convenient to use the following definition, which we adapt to recursive structures from Vardi [V]

Definition: The *data complexity* of a language L is the level of difficulty of computing the sets $Gr(Q_e) = \{(B,u) | u \in Q(B)\}$ for an expression e in L, where Q_e is the query expressed by e, and B denotes a recursive data base

(i.e., structure). A language L is *data-complete* (or D-*complete* for short) for a computational class C if for every expression e in L, $Gr(Q_e)$ is in C, and there is an expression e_0 in L such that $Gr(Q_{e_0})$ is hard for C.

Here we restrict ourselves to the consistent, or generic, queries, which are the ones that preserve isomorphisms. In fact, we require that they preserve the isomorphisms of *all* structures, not only recursive ones, under the assumption that there exist oracles for their relations. That is, Q is consiedered here to be generic if for all B_1, B_2, if $B_1 \cong B_2$ then $Q(B_1) \cong Q(B_2)$, where $Q(B)$ is the result of applying Q to oracles for the relations in B.

We now provide a very brief description of the main results of this part of [HH3]:

1. First-order logic expresses generic queries from the entire arithmetical hierarchy, but it does not express all of them. For example, the connectivity of recursive graphs is arithmetical, but is not expressible by a first-order formula.
2. The logical formalism E-Σ^1_1, which consists of existential second-order formulas, is D-complete for the complexity class Σ^1_1 of the analytical hierarchy, but there are queries, even arithmetical ones, that are not expressible in E-Σ^1_1. However, over ordered structures (that is, if a built-in total order is added to the vocabulary), all Σ^1_1 properties are expressible in E-Σ^1_1.
3. For $k \geq 2$, a stronger result is proved, analogous to Fagin's result for finite structures: the logical formalism E-Σ^1_k expresses precisely the generic properties of the complexity class Σ^1_k. This means that every generic query over some vocabulary σ that is expressible by a Σ^1_k formula over interpreted recursive predicates, is also expressible by an uninterpreted E-Σ^1_k formula over σ.[4]
4. Monadic E-Σ^1_1, where the second-order quantifiers are restricted to range over unary relations (sets), is D-complete for Σ^1_1, and strict E-Σ^1_1 is D-complete for Σ^0_2.
5. Consider fixpoint logic, which is obtained by adding least fixpoint operators to first-order formulas [CH2, I, Mos]. Denote by FP_1 positive fixpoint logic, in which the least fixpoint operator is restricted to positive formulas, and by FP the hierarchy obtained by alternating the least fixpoint operator with the first-order constructs. In finite structures, the FP hierarchy collapses, and a single fixpoint operator suffices [I]. In contrast, for recursive structures FP_1 is D-complete for Π^1_1, and hence $\neg\mathrm{FP}_1$ (negations of formulas in FP_1) is D-complete for Σ^1_1. The data complexity of FP is exactly Δ^1_2, and an example is shown of a query expressible in FP that is hard for both Σ^1_1 and Π^1_1.

[4] In the direction going from expressibility in E-Σ^1_k to computability in Σ^1_k, the second-order quantifiers are used to define a total order and predicates $+$ and $*$, which, in turn, are used to define the needed elementary arithmetic expression. Each subset of elements must contain a minimum in the defined order, which requires for its definition a universal second-order quantifier. This explains why the result requires $k \geq 2$.

The second part of [HH3] deals with 0–1 laws on recursive structures.

If C is a class of finite structures over some vocabulary σ and if P is a property of some structures in C, then the *asymptotic probability* $\mu(P)$ on C is the limit as $n \to \infty$ of the fraction of the structures in C with n elements that satisfy P, provided that the limit exists. Fagin [F2] and Glebskii et al. [GKLT] were the first to discover the connection between logical definability and asymptotic probabilities. They showed that if C is the class of all finite 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, known as the *0–1 law for first-order logic*, became the starting point of a series of investigations aimed at discovering the relationship between expressibility in a logic and asymptotic probabilities. Several additional logics, such as fixpoint logic, iterative logic and strict E-Σ^1_1, have been shown by various authors to satisfy the 0–1 law too.

A standard method for establishing 0–1 laws on finite structures, originating in Fagin [F2], is to prove that the following *transfer theorem* holds: there is an infinite structure $\mathbf{A}$ over σ such that for any property P expressible in L:

$$\mathbf{A} \models P \text{ iff } \mu(P) = 1 \text{ on } C.$$

It turns out that there is a single countable structure $\mathbf{A}$ that satisfies this equivalence for all the logics mentioned above. Moreover, $\mathbf{A}$ is characterized by an infinite set of *extension axioms*, which, intuitively, assert that every type can be extended to any other possible type. More specifically, for each finite set X of points, and each possible way that a new point $y \notin X$ could relate to X in terms of atomic formulas over the appropriate vocabulary, there is an extension axiom that asserts that there is indeed such a point. For example, here is an extension axiom over a vocabulary containing one binary relation symbol R:

$$\forall x_1 \forall x_2 \Big(x_1 \neq x_2 \Rightarrow \exists y \, (y \neq x_1 \wedge y \neq x_2 \wedge$$

$$(y, x_1) \in R \wedge (x_1, y) \notin R \wedge (y, x_2) \notin R \wedge (x_2, y) \in R) \Big).$$

Fagin realized that the extension axioms are relevant to the study of probabilities on finite structures and proved that on the class C of all finite structures of vocabulary σ, $\mu(\tau) = 1$ for any extension axiom τ. The theory of all extension axioms, denoted T, is known to be ω-categorical (that is, every two countable models are isomorphic), so that $\mathbf{A}$, which is a model for T, is unique up to isomorphism. This unique structure is called the *random countable structure*, since it is generated, with probability 1, by a random process in which each possible tuple appears with probability 1/2, independently of the other tuples. The random graph was studied by Rado [Ra], and is sometimes called the *Rado graph*.

Now, since all countable structures are isomorphic to $\mathbf{A}$ with probability 1, the asymptotic probability of each (generic) property P on countable structures is trivially 0 or 1, since this depends only on whether $\mathbf{A}$ satisfies P or not. Hence, the subject of 0–1 laws over the class of all countable structures is not interesting.

As to recursive structures, which are what we are interested in here, one is faced with the difficulty of defining asymptotic probabilities, since structure size is no longer applicable.

The heart of this part of [HH3] is a proposal for a definition of 0–1 laws for recursive structures.

Definition: Let $\mathcal{F} = \{F_i\}_{i=1}^{\infty}$ be a sequence of recursive structures over some vocabulary, and let P be a property defined over the structures in $\mathcal{F}$. Then the *asymptotic probability* $\mu_{\mathcal{F}}(P)$ is defined to be

$$\mu_{\mathcal{F}}(P) = \lim_{n \to \infty} \frac{|\{F_i |\ 1 \leq i \leq n,\ F_i \models P\}|}{n}.$$

Definition: Let $\mathcal{F} = \{F_i\}_{i=1}^{\infty}$ be a sequence of recursive structures over some vocabulary σ. We say that $\mathcal{F}$ is a *T-sequence* if $\mu_{\mathcal{F}}(\tau) = 1$ for every extension axiom τ over σ.

As an example, a sequence of graphs that are all isomorphic to the countable random graph $\mathbf{A}$ is a T-sequence. We shall use U to denote one such sequence. Here is another example of a T-sequence: take $\mathcal{F} = \{F_n\}_{n=1}^{\infty}$, where each F_n is a graph satisfying all the n-extension axioms and is built in stages. First take n distinct and disconnected points. Then, at each stage add a new point z for every set $\{x_1, \ldots, x_n\}$ from previous stages and for every possible extension axiom for it, and connect z accordingly.

Definition: Let P be a property of recursive structures. We say that the *0–1 law holds for P* if for every T-sequence $\mathcal{F}$ the limit $\mu_{\mathcal{F}}(P)$ exists and is equal to 0 or 1. The *0–1 law holds for a logic L on recursive structures* if it holds for every property expressible in L.

Here are some of the results proved in [HH3] for this definition of 0–1 laws over recursive structures.

Theorem: The 0–1 law holds for all properties of recursive structures definable in first-order logic, strict E-Σ_1^1 and strict E-Π_1^1. Moreover, if $\mathbf{A}$ is the countable random structure, P is such a property and $\mathcal{F}$ is a T-sequence, then $\mathbf{A} \models P$ iff $\mu_{\mathcal{F}}(P) = 1$.

However, the property of a graph having an infinite clique, for example, is shown not to satisfy the 0–1 law, so that the law does not hold in general for E-Σ_1^1-properties.

As a result of the theorem, a property for which the 0–1 law does not hold is not expressible in first-order logic, strict E-Σ_1^1 or strict E-Π_1^1. In fact, we have the following:

Theorem: Every property on recursive structures that is true in $\mathbf{A}$, but does not have probability 1 on some T-sequence, is not expressible by an E-Π_1^1 sentence or by a strict E-Σ_1^1 sentence.

In way of applying the techniques, we show in [HH3] that the following properties are not expressible by an E-Π_1^1 sentence or by a strict E-Σ_1^1 sentence:

a recursive graph having an infinite clique, a recursive graph having an infinite independent set, a recursive graph satisfying all the extension axioms, and a pair of recursive graphs being isomorphic.

Acknowledgements: I would like to thank Richard Beigel, who by asking the question addressed in Section 2, introduced me to this area. His work with Bill Gasarch has been a great inspiration. Very special thanks go to Tirza Hirst, without whom this paper couldn't have been written. Apart from Section 2, the results are all joint with her, and form her outstanding PhD thesis.

References

[AV] S. Abiteboul and V. Vianu, "Generic Computation and Its Complexity", *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pp. 209–219, ACM Press, New York, 1991.

[AMS] R. Aharoni, M. Magidor and R. A. Shore, "On the Strength of König's Duality Theorem", *J. of Combinatorial Theory (Series B)* **54**:2 (1992), 257–290.

[B1] D.R. Bean, "Effective Coloration", *J. Sym. Logic* **41** (1976), 469–480.

[B2] D.R. Bean, "Recursive Euler and Hamiltonian Paths", *Proc. Amer. Math. Soc.* **55** (1976), 385–394.

[BG1] R. Beigel and W. I. Gasarch, unpublished results, 1986-1990.

[BG2] R. Beigel and W. I. Gasarch, "On the Complexity of Finding the Chromatic Number of a Recursive Graph", Parts I & II, *Ann. Pure and Appl. Logic* **45** (1989), 1–38, 227–247.

[Bu] S. A. Burr, "Some Undecidable Problems Involving the Edge-Coloring and Vertex Coloring of Graphs", *Disc. Math.* **50** (1984), 171–177.

[CH1] A. K. Chandra and D. Harel, "Computable Queries for Relational Data Bases", *J. Comp. Syst. Sci.* **21**, (1980), 156–178.

[CH2] A.K. Chandra and D. Harel, "Structure and Complexity of Relational Queries", *J. Comput. Syst. Sci.* **25** (1982), 99–128.

[F1] R. Fagin, "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets", In *Complexity of Computations* (R. Karp, ed.), SIAM-AMS Proceedings, Vol. 7, 1974, pp. 43–73.

[F2] R. Fagin, "Probabilities on Finite Models", *J. of Symbolic Logic*, **41**, (1976), 50 – 58.

[GL] W. I. Gasarch and M. Lockwood, "The Existence of Matchings for Recursive and Highly Recursive Bipartite Graphs", Technical Report 2029, Univ. of Maryland, May 1988.

[GKLT] 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**, (1969), 142–154.

[H] D. Harel, "Hamiltonian Paths in Infinite Graphs", *Israel J. Math.* **76**:3 (1991), 317–336. (Also, *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, New Orleans, pp. 220–229, 1991.)

[HH1] T. Hirst and D. Harel, "Taking it to the Limit: On Infinite Variants of NP-Complete Problems", *J. Comput. Syst. Sci.*, to appear. (Also, *Proc. 8th IEEE Conf. on Structure in Complexity Theory*, IEEE Press, New York, 1993, pp. 292–304.)

[HH2] T. Hirst and D. Harel, "Completeness Results for Recursive Data Bases", *J. Comput. Syst. Sci.*, to appear. (Also, *12th ACM Ann. Symp. on Principles of Database Systems*, ACM Press, New York, 1993, 244–252.)

[HH3] T. Hirst and D. Harel, "More about Recursive Structures: Zero-One Laws and Expressibility vs. Complexity", in preparation.

[I] N. Immerman, "Relational Queries Computable in Polynomial Time", *Inf. and Cont.* **68** (1986), 86–104.

[KT] P. G. Kolaitis and M. N. Thakur, "Logical definability of NP optimization problems", *6th IEEE Conf. on Structure in Complexity Theory*, pp. 353–366, 1991.

[MR] A. Manaster and J. Rosenstein, "Effective Matchmaking (Recursion Theoretic Aspects of a Theorem of Philip Hall)", *Proc. London Math. Soc.* **3** (1972), 615–654.

[Mo] A. S. Morozov, "Functional Trees and Automorphisms of Models", *Algebra and Logic* **32** (1993), 28–38.

[Mos] Y. N. Moschovakis, *Elementary Induction on Abstract Structures*, North Holland, 1974.

[NR] A. Nerode and J. Remmel, "A Survey of Lattices of R. E. Substructures", In *Recursion Theory*, Proc. Symp. in Pure Math. Vol. 42 (A. Nerode and R. A. Shore, eds.), Amer. Math. Soc., Providence, R. I., 1985, pp. 323–375.

[PR] A. Panconesi and D. Ranjan, "Quantifiers and Approximation", *Theor. Comp. Sci.* **107** (1993), 145–163.

[PY] C. H. Papadimitriou and M. Yannakakis, "Optimization, Approximation, and Complexity Classes", *J. Comp. Syst. Sci.* **43**, (1991), 425–440.

[Ra] R. Rado, "Universal Graphs and Universal Functions", *Acta Arith.*, **9**, (1964), 331–340.

[R] H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.

[V] M. Y. Vardi, "The Complexity of Relational Query Languages", *Proc. 14th ACM Ann. Symp. on Theory of Computing*, 1982, pp. 137–146.