ISRAEL JOURNAL OF MATHEMATICS 76 (1991), 317–336

HAMILTONIAN PATHS IN INFINITE GRAPHS*

BY

DAVID HAREL

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

ABSTRACT

A tight connection is exhibited between infinite paths in recursive trees and Hamiltonian paths in recursive graphs. A corollary is that determining Hamiltonicity in recursive graphs is highly undecidable, viz, Σ_1^1-complete. This is shown to hold even for highly recursive graphs with degree bounded by 3. Hamiltonicity is thus an example of an interesting graph problem that is outside the arithmetic hierarchy in the infinite case.

1. Introduction

Computer scientists are interested predominantly in finite objects. Insofar as they are interested in *infinite* objects, these must be computable, i.e., recursive, thus admitting an effective finite representation. In fact, insight into finite objects can often be gleaned from results about (infinite) recursive variants thereof. Given the central place that finite graphs occupy in theoretical computer science, and the many results and open questions surrounding them, it would seem appropriate to investigate recursive graphs too. Indeed, a significant amount of work has been carried out recently regarding the complexity of problems on recursive graphs. Some of the first papers were written by Manaster and Rosenstein [10] and Bean [2, 3]. Since then, a variety of problems have been considered, including ones that are NP-complete for finite graphs, such as k-colorability and Hamiltonicity

* Parts of this research were carried out during a visit to IBM T.J. Watson Research Center, Hawthorne, NY, in the Summer of 1990. The author holds the William Sussman Professorial Chair in Mathematics.
Received March 26, 1991 and in revised form October 10, 1991

[1, 3, 5, 6, 7, 10], and ones that are in P in the finite case, such as Eulerian paths [3, 4].

In most cases (including the above examples) the problems have turned out to be undecidable. This is true even for highly recursive graphs [2], i.e., ones for which node degree is finite (but not necessarily bounded) and the set of neighbors of a node is computable. Beigel and Gasarch [4] and Gasarch and Lockwood [7] have investigated the precise level of undecidablity of many such problems, and have shown that they reside on low levels of the arithmetic hierarchy. For example, it is shown in [4] that detecting the existence of an Eulerian path is Π_3^0-complete for recursive graphs and Π_2^0-complete for highly recursive graphs.

The case of Hamiltonian paths seems to have hitherto defied satisfactory classification. Bean showed in [3] that the problem is undecidable (even for planar graphs), but the precise characterization was not known. In this paper we prove that Hamiltonicity is in fact *highly* undecidable, *viz,* Σ_1^1-complete. In Section 2 the result is proved for recursive graphs, and in Section 4 it is shown to hold even for highly recursive graphs with degree bounded by 3.[1] Hamiltonicity thus becomes an example of an interesting graph problem that becomes highly undecidable in the infinite case. Another example can be found in the (independent) work of Aharoni, Magidor and Shore [1], from which it follows that perfect matching in recursive graphs is also Σ_1^1-complete.

These two results give rise to the far more general question of trying to characterize graph problems whose infinite version is outside the arithmetic hierarchy.

In Section 3 we provide a more subtle version of the construction of Section 2, obtaining a stronger connection between infinite paths in recursive trees and Hamiltonian paths in recursive graphs: When transforming a tree T into a graph G, or vice versa, the mappings between infinite paths in T and Hamiltonian paths in G are not only recursive, but yield a tight synchronization; making progress in one path depends only on fixed portions of the other.

2. 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 **one-way** (respectively, **two-way**) **Hamilto-**

1 Using a different reduction, T. Hirst has recently managed to strengthen this result, showing that it holds even for planar graphs.

nian 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 [3] has shown that determining Hamiltonicity in highly recursive graphs is undecidable. It is worth sketching the proof here.[2] The reduction is from non-well-foundedness of recursive trees with finite outdegree. Given such a tree T, a graph G is constructed, such that infinite paths in T map to Hamiltonian paths in G. The idea is that as a path in T moves down the tree, the Hamiltonian path in G cycles through all nodes of T that reside at the current level, ending the cycle at a point from which the process of moving down T can be resumed. See Fig. 1.

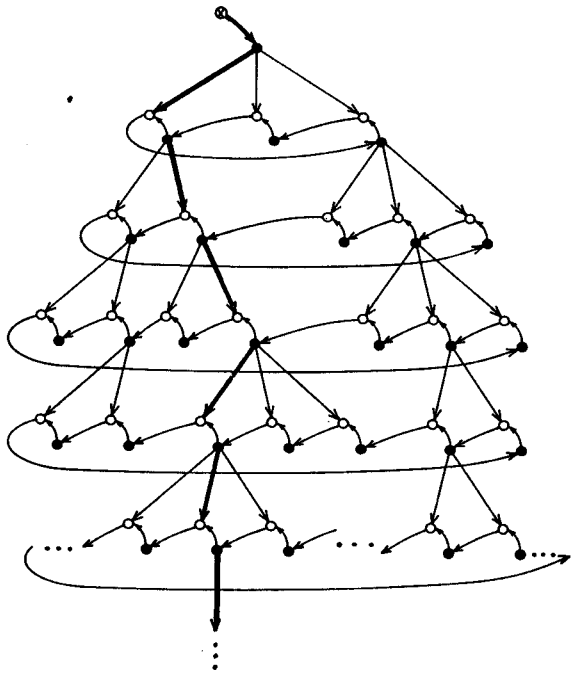

Fig. 1. Bean's reduction [3] from finitely branching trees (for directed graphs). In simulating an infinite path in the tree (thick lines), the Hamiltonian path cycles through all nodes on each level.

A fact that is crucial to this construction is the finiteness of T's outdegree,

2 Actually, Bean's paper contains two proofs of undecidability, of which this is the second.

so that the proof does not generalize to trees with infinite outdegree. In fact, the proofs in [3] only establish that Hamiltonicity is hard for Π_1^0, or co-r.e. In showing in this paper that the problem is actually Σ_1^1-complete, we exhibit a reduction from non-well-foundedness of recursive trees with possibly *infinite* outdegree, which is well-known to be a Σ_1^1-complete problem [12, Thm. 16.XX]. In this section, we establish the result for recursive graphs. The stronger version, for highly recursive graphs with bounded degree, is obtained in Section 4.

THEOREM 1: *Detecting (one-way or two-way) Hamiltonicity in a (directed or undirected) recursive graph is* Σ_1^1*-complete.*

Proof: 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.

To show Σ_1^1-hardness, assume a recursive tree T is given, whose set of nodes is $\mathcal{N} = 0, 1, 2, 3, \ldots$, whose root is 0, and whose **parent-of** function is recursive. Recall that T can be of infinite outdegree. We construct an undirected graph G_1, such that G_1 has a one-way Hamiltonian path iff T has an infinite path.

For each element $n \in \mathcal{N}$, the graph G_1 has five internal nodes, n^u, n^d, n^r, n^l and n^{ur}, standing, respectively, for **up, down, right, left** and **up-right**. Here are G_1's edges (see Fig. 2):

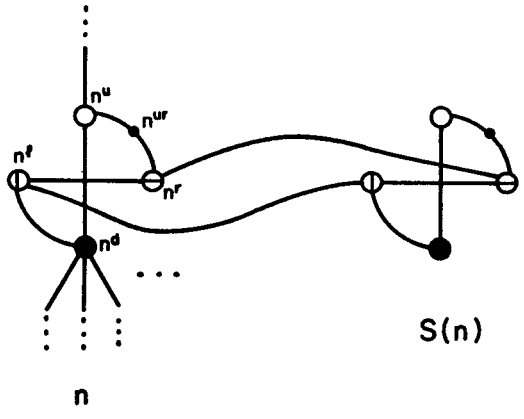

Fig. 2. The construction of G_1.

• For each such cluster, G_1 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_1.

• 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_1 we will think of $S(n)$ as being n's **shadow node**, and the two are connected as follows:[3]

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

• To complete the construction, there is one additional root node g in G_1, with an edge g —— 0^u.

Since T is a recursive tree and S, as a function, is recursive in T, it is obvious that G_1 is a recursive graph.

LEMMA 1: *T has an infinite path from 0 iff G_1 has a Hamiltonian path.*

Proof (Only-if): Suppose T has an infinite path p. A Hamiltonian path p' in G_1 starts at the root g, and moves down G_1's versions of the nodes in p, taking detours to the right to visit n's shadow node $S(n)$ whenever $S(n) \notin p$. To move down without visiting $S(n)$, p' covers n's five internal nodes in the order n^u, n^{ur}, n^r, n^l, n^d. When it *does* visit $S(n)$, the order is as follows (see Fig. 3):

$$n^u,\ n^{ur},\ n^r,\ S(n)^r,\ S(n)^{ur},\ S(n)^u,\ S(n)^d,\ S(n)^l,\ n^l,\ n^d.$$

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. In this way, p' will have clearly covered all five internal nodes of each element of $\mathcal{N}$, and will have done so exactly once. Hence, p' is Hamiltonian.

(*If*) Suppose G_1 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_1 must contribute to p exactly two incident edges, one incoming and one outgoing.

---

3 Clearly, given T, the function $S : \mathcal{N} \to \mathcal{N}$ is not necessarily one-one. In fact, Fig. 2 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.

CLAIM: 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.[4]

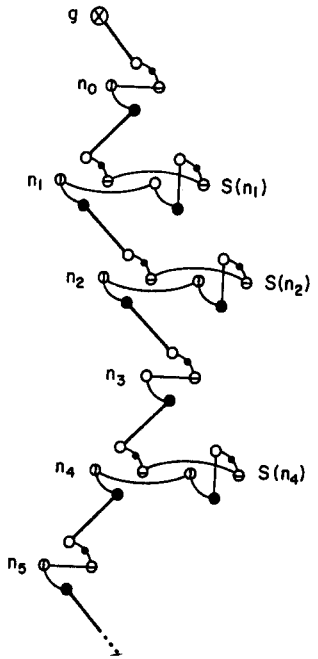

Fig. 3. A Hamiltonian path in G_1. Here, $S(n_0), S(n_3)$ and $S(n_5)$ are in p, and are therefore not visited as shadows.

Proof: (Refer to Fig. 2.) Assume p contains the T-edge incident to n^u. Consider n^{ur}. 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. Note now that 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. ∎

This completes the proof for the undirected case with one-way paths. Directed

4 In particular, this means that there is indeed some such T-edge, so that n cannot be a leaf of T.

graphs are treated easily by replacing each edge with two directed ones, and two-way paths are treated by simply adding new nodes $-1, -2, -3, \ldots$ backward-chained to the root g of the graph. ∎

We could have adopted a slightly different strategy in the proof of Theorem 1. It is possible to prove the hardness direction for directed graphs using only two internal nodes, n^u and n^d, and then to show how to go from directed graphs to undirected ones by a generic transformation. The present approach was preferred for expository reasons.

3. A Closer Connection

To prove Σ_1^1-hardness in the previous section, all we needed was to transform a recursive tree T into a graph G, such that T has an infinite path iff G has a Hamiltonian path. No kind of connection was required between the corresponding paths. In this section, after showing that the transformation from T to G_1 in Section 2 actually yields paths that are recursive in each other, we modify the construction, obtaining an even stronger connection.

Recall that we define a one-way infinite path p in a graph as a function from $\mathcal{N}$ to the set of vertices, such that, for each n, the pair $\langle p(n), p(n+1)\rangle$ is an edge. Accordingly, we say that a path p is **recursive in** a path p' if there is a Turing machine M, which, given n, computes $p(n)$ using an oracle for the function p'. Also, if A and B are sets of paths,[5] we say that a function $F : A \to B$ is **recursive-yielding,** if for all $p \in A$, $F(p)$ is recursive in p.

PROPOSITION 1: *The transformations between paths exhibited in the two parts of the proof of Lemma 1 are recursive-yielding.*

Proof: For the most part, this is easy, and we leave it to the reader. However, one point needs clarification.

Assume we are given an oracle for an infinite path p in T. Consider the corresponding Hamiltonian path p' in G_1 from the proof of Lemma 1. To show that p' is recursive in p, we have to convince ourselves, among other things, that

5 Here, A and B need not be recursive themselves, nor need their elements. In fact, as shown in [3], a recursive graph can be Hamiltonian but contain no recursive Hamiltonian path – a fact that is not true for Eulerian paths. Moreover, deciding whether a recursive graph has a *recursive* Hamiltonian path is easily seen to be arithmetical; it is actually Σ_3^0-complete [8].

whenever we reach a node n^r in p' we can indeed decide whether we take a trip to visit n's shadow $S(n)$ or not. Now, such a trip is taken iff $S(n)$, when considered as a node of T, is not in the path p; i.e., iff there is no i such that $p(i) = S(n)$. Despite the unbounded, existential nature of this formulation, we can make such a decision by advancing along p, using the oracle and the recursiveness of T to test, for increasing i, whether $p(i)$ is on the path from $S(n)$ to 0, the root of T. Since $S(n)$ must appear somewhere in T, and since p is infinite, eventually either $p(i)$ will be on that path or we will find some $p(i)$ to be equal to $S(n)$. ∎

A disturbing aspect of the point made in this proof is that to decide whether or not to visit n's shadow $S(n)$ in the Hamiltonian path p' we might have to pose an unknown number of queries to the oracle of p, the infinite path in T. Thus, we cannot *bound* the number of queries to p that are required between computing $p'(n)$ and $p'(n+1)$. Put loosely, the computation of the Hamiltonian path in G_1 does not proceed at a pace which is bounded by the pace of the infinite path in T.

We now present a more subtle version of the construction of Section 2, which results in a much tighter connection. We shall set things up so that progress in p' is **tightly synchronized** with progress in p. We take this to mean that between computing $p'(n)$ and $p'(n+1)$ we shall need at most a fixed number of queries to p's oracle. (Actually, in the case just discussed, that is, going from an infinite path in T to a Hamiltonian path in G_1, we shall really need only one query at most, which is, of course, optimal.)

The idea is to define the shadow of a node differently, shifting all the computation to the phase of constructing the graph. The consequence is that we completely eliminate the need to *choose* whether we visit a shadow node or not; the Hamiltonian path p' in G_2, the graph we now construct, will simulate moving down its corresponding infinite path p in T as before, but will regularly visit the shadow of *every* n. Thus, progress in p' is linked to progress in p in the strongly synchronized sense discussed above.

Let $\mathcal{N}^+$ denote the set $\{0, 0^+, 1, 1^+, 2, 2^+, \ldots\}$, ordered by $i < i^+ < i+1$. Thus $\mathcal{N}^+$ contains "real" integers, interleaved with "dummy" $^+$-ed versions. The nodes of G_2 will represent elements of $\mathcal{N}^+$, not $\mathcal{N}$, and most of the construction is just as before. The difference is in the definition of shadows. Here, we define a new $S(n)$ as follows (and connect it to n as before, by the edges n^r —— $S(n)^r$ and $S(n)^l$ —— n^l). Define $S : \mathcal{N} \to \mathcal{N}^+$ inductively, moving down levels of T. For a

node n in T, $S(n)$ is the first element x of $\mathcal{N}^+$ (that is, first in the above ordering of $\mathcal{N}^+$) satisfying:

1. x does not appear on the path from 0, the root of T, to n, inclusive. (This clause applies only if $x \in \mathcal{N}$.)
2. x is not the shadow $S(m)$ of any node m on the path from 0 to n, exclusive of n, of course. (Only one application of S here; x might, in fact, be $S(S(m))$ for some m on that path.)
3. x is not in the subtree rooted at n. (Again, this applies only if $x \in \mathcal{N}$.)

LEMMA 2: *S is total recursive.*

Proof: That S is total follows from the presence of the $^+$-ed elements: For any n, the order in $\mathcal{N}^+$ implies that every second attempt to assign n its shadow will be a $^+$-ed element. However, the only reason not to set $S(n) = m^+$ is requirement 2 above, so that any m larger than n's level in T can be so assigned.

To see that S is recursive, when given n run through $\mathcal{N}^+$ in order, checking the three requirements; totality of S guarantees eventual success. To check requirements 1 and 2, we need the path from the root 0 to n, and for requirement 3 we need the path from 0 to the candidate x. Both are constructible by the recursiveness of T. As to the elements $S(m)$ that are needed for checking requirement 2, we compute them inductively, from 0 down the path to n. ∎

The following crucial lemma shows that the set of shadows of any infinite path in T completes the set of nodes of that path to be exactly all of $\mathcal{N}^+$, without repeats.

LEMMA 3: *Let p be any infinite path in T starting at the root 0, and let $S(p)$ be the set of shadows of the nodes of p; that is, $S(p) = \{S(n) \mid n \in p\}$. Then,*
(i) *if $n, m \in p$, with $n \neq m$, then $S(n) \neq S(m)$.*
(ii) $p \cap S(p) = \emptyset$
(iii) $p \cup S(p) = \mathcal{N}^+$

Proof: Clause (i) states that, while S is not in general 1-1, it becomes so when restricted to an infinite path p in T rooted at 0. This follows immediately from requirement 2 in the definition of S.

Clause (ii) follows from requirements 1 and 3 therein.

To establish (iii), we have to show that each element of $\mathcal{N}^+ - p$ will eventually show up as the shadow of some node in p. First consider the $^+$-ed elements. By

the observations made earlier, n^+ will be eventually assigned to some node along p at distance at most $2n$ from the root.

Turning to elements of $\mathcal{N} - p$, note that for any such node n, since it is in the tree T but not on the path p, there must be some point in p after which n does not appear in the subtree rooted at any subsequent node of p. Having thus satisfied requirement 3 of shadows, and noting that such an n always satisfies requirement 1 (since it is not on p at all), the fact that elements of $\mathcal{N}^+$ are tried out in order and never reassigned along p implies that n will eventually be assigned as the shadow of some node of p. ∎

A consequence of Lemma 3 is that, if we manage to simulate traveling down a path in T, visiting each node and paying a visit to its shadow too, then we will have encountered each element of $\mathcal{N}^+$, and we will have done so exactly once. It is now easy to see that an analog of Lemma 1 holds for G_2 too, and that the resulting pairs of paths are tightly synchronized, as described above.

This close connection between infinite paths in trees and Hamiltonian paths in graphs holds in a reverse transformation too. In Section 2 we showed Hamiltonicity to be *in* Σ_1^1 by formulating the property as a Σ_1^1 formula. It follows that we can transform a recursive graph G into a recursive tree T such that G has a Hamiltonian path iff T has an infinite path. However, this alone says nothing about the correspondence between the two kinds of paths. In the present context, we can construct T from G so that there will be a similarly recursive and tightly synchronized correspondence between the paths.

The idea is to first construct a recursive **marked** tree T' (see [9]) so that the n'th offspring of its root corresponds to a certain unwinding of G from its node n. The unwindings are carried out in such a way that nodes do not repeat along paths, and with the i'th mark on a path signifying that we have already seen nodes 1 through i along that path. It is then easy to prove the analogue of Proposition 1, by exhibiting a recursive transformation from Hamiltonian paths in G to **recurrences** in T' (i.e., infinite paths, each containing infinitely many marks), and vice versa. T' is then transformed into an unmarked tree T as in [9], such that the recurrences of T' correspond recursively to the set of infinite paths in T. Both transformations can be carried out in such a way that the appropriate pairs of paths are also tightly synchronized in the sense of this section. We omit the details.

One more remark is in place. We have concentrated on showing that the appro-

priate pairs of paths are recursive in each other and are tightly synchronized. An even stronger connection between a tree T and a constructed graph G would be to show that the set of infinite paths of the former is **recursively isomorphic** (see [12]) to the set of Hamiltonian paths of the latter (and that tight synchronization holds too). For a directed graph version of G_2, with the shadow definition of the present section, this can be easily shown. However, we have not been able to establish recursive isomorphism for undirected graphs.

4. Highly Recursive Graphs

We now return to the main quest of the paper, namely, proving Σ_1^1-completeness of the Hamiltonicity problem, and establish the stronger version.

A **highly recursive graph** is a recursive graph $G = (V, E)$, for which there is a recursive function H from V to finite subsets of V, such that $H(v) = \{u \mid \langle v, u\rangle \in E\}$.

THEOREM 2: *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: Clearly, we need only prove Σ_1^1-hardness, and we may concentrate on undirected graphs.

We first modify the construction of G_1 of Section 2, yielding a graph G_3 with degree 5. Later, we shall address the required recursiveness of a node's neighborhood, and also reduce the 5 to 3.

Infinite branching arises in two contexts in G_1: (i) the branching at n^d inherited by n from the tree T, and (ii) the branching at both n^l and n^r that is caused by n's pre-shadows.[6]

To deal with the infinite degree of the down-nodes caused by the branching in T, we add to each cluster of nodes representing an $n \in \mathcal{N}$ (except for the root 0) a new node, n^{u_u}, connected to n^u. Rather than branching out of n^d to all of n's offspring in T, n^d leads out to $m_1^{u_u}$ only, where m_1 is the smallest offspring of n in the natural order on $\mathcal{N}$. Thereafter, $m_1^{u_u}$ leads to $m_2^{u_u}$, where m_2 is the next smallest offspring, and so on. See Fig. 4.

This is not quite as simple as it might sound, since we have to retain the Hamiltonian nature of the graph. To wit, say $m_j < m_i < m_k$ are three T-

6 In this section we return to the original definition of shadows from Section 2.

offspring of n, and m_i is n's successor in p. Then reaching m_i^{uu} from n^d as just described has the effect of visiting m_j^{uu} but *not* visiting m_k^{uu}. Hence, when the Hamiltonian path derived from p visits m_j's cluster,[7] it will have to skip over m_j^{uu}, whereas when it visits m_k's cluster it will have to visit m_k^{uu} too, without skipping. Consequently, we have to modify the internal cluster of a node n, so that, if and when it is visited as a shadow by the Hamiltonian path, we should be able to choose whether we skip over its uu node or not.

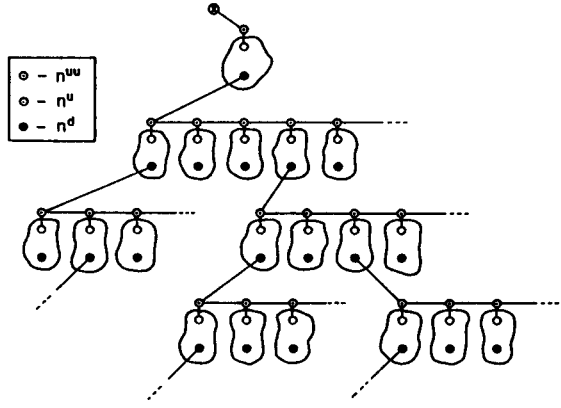

Fig. 4. Connecting to T-successors with finite branching. Smaller nodes are drawn to the left.

As to the branching resulting from the pre-shadow relationship, the idea is similar. Rather than connecting n with its shadow node $S(n)$ directly, via the two edges n^r —— $S(n)^r$ and $S(n)^l$ —— n^l, we consider, for each m, the set of pre-shadows $S^{-1}(m)$, ordered by the natural order on $\mathcal{N}$.[8] These are doubly-linked in a linear fashion, using two new nodes for each n, denoted n^{ts} and n^{fs} (standing for **to-shadow** and **from-shadow**). The smallest element of $S^{-1}(m)$ is connected to m directly. See Fig. 5. Here, too, we must construct the cluster so that a Hamiltonian path can either skip these two new nodes or visit them both.

Adding to the intricacy of the construction is the need to retain the delicate

7 Since $m_j \notin p$, this will happen at that node of p which is on level m_j in T, at which point m_j will be visited as a shadow.

8 Note that $S^{-1}(m)$ is really just the set of nodes residing at level m of T.

balance that will enable us to prove an analogue of the Claim appearing in Lemma 1. Here, then is the full definition of the graph G_3, whose degree is bounded by 5.

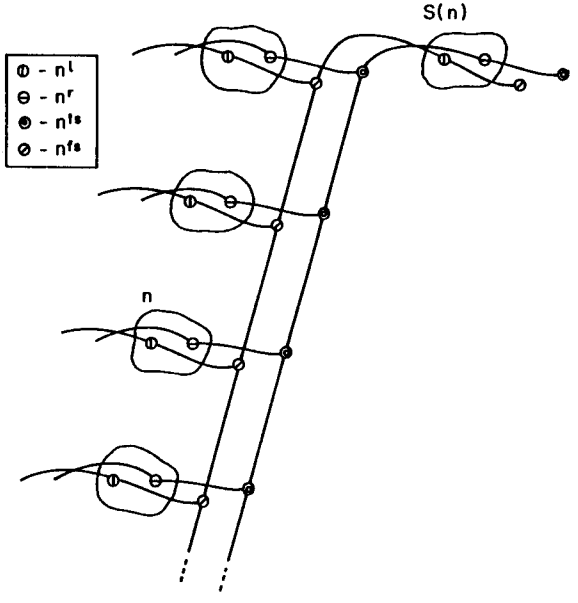

Fig. 5. Connecting to shadow nodes with finite branching. Smaller nodes are drawn higher up.

G_3 has ten internal nodes for each element $n \in \mathcal{N}$. In addition to the ones of G_1, namely, n^u, n^d, n^r, n^l and n^{ur}, we also have $n^{uu}, n^{us}, n^{rr}, n^{fs}$, and n^{ts}. Now for the edges (see Fig. 6 to help follow the definition):

- For each such cluster, G_3 has the following 13 internal edges:

$$n^l \text{ — } n^d \text{ — } n^u \text{ — } n^{us} \text{ — } n^{ur} \text{ — } n^{rr} \text{ — } n^r \text{ — } n^{fs} \text{ — } n^l,$$
$$n^u \text{ — } n^{uu} \text{ — } n^{us},$$
$$n^{rr} \text{ — } n^{ts} \text{ — } n^r,$$
$$n^{fs} \text{ — } n^{ts}.$$

- For each n, if $m_1 < m_2 < m_3 < \ldots$ is (the finite or infinite) set of offspring of the node n in T, then G_3 has the following T-edges:

$$n^d \text{ — } m_1^{uu} \text{ — } m_2^{uu} \text{ — } m_3^{uu} \text{ — } \cdots$$

• For each n, if $m_1 < m_2 < m_3 < \ldots$ is (the finite or infinite) set of all nodes on level n in T (i.e., $n = S(m_i)$ for each i), then G_3 has the following edges:

$$n^r \text{ —— } m_1^{ts} \text{ —— } m_2^{ts} \text{ —— } m_3^{ts} \text{ —— } \cdots$$

and

$$n^l \text{ —— } m_1^{fs} \text{ —— } m_2^{fs} \text{ —— } m_3^{fs} \text{ —— } \cdots$$

• The root node g is added, with an edge g —— 0^{uu}.

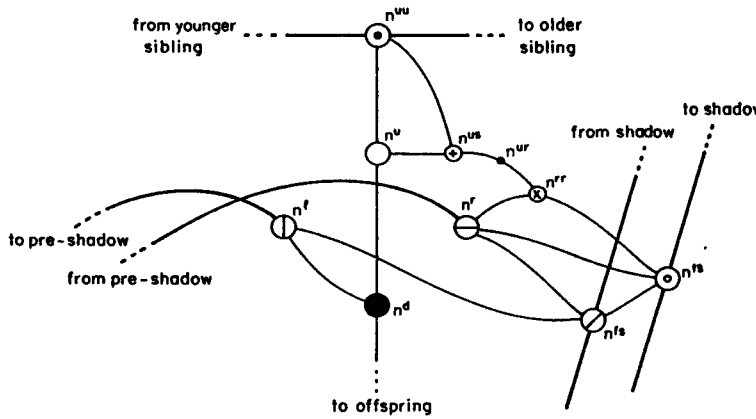

Fig. 6. Internal nodes and edges in G_3.

Before dealing with the issue of recursiveness, we prove the main lemma:

LEMMA 4: *T has an infinite path from 0 iff G_3 has a Hamiltonian path.*

Proof (Only-if): Given an infinite path p in T, a Hamiltonian path p' in G_3 will simulate moving down p in a manner that follows the discussion above. The path starts at g, and enters 0^{uu}. Thereafter, there are two kinds of n: those for which $n \in p$ (including, of course, 0), and those for which $n \notin p$.

If $n \in p$, the path p' enters the n-cluster from "above", i.e., n^{uu} will be reached from n's younger sibling in T or directly from n's parent (which is g for the case $n = 0$). If $n \notin p$, then the n-cluster is visited as the shadow of the one node m along p which is on level n in T (i.e., $n = S(m)$). Thus, p' moves from m^{ts} along the ts nodes of all smaller pre-shadows of n until it reaches n^r, from which it will cycle through all the internal nodes of n. It will leave n via n^l, moving along the fs nodes of those same pre-shadows, and back to m^{fs}.

Let us now see what kind of tours p' has to carry out *within* clusters. We first claim that, for $n \in p$, when n^{uu} is entered by p' (and it will be entered from above), n^{fs} and n^{ts} will not have been visited yet. The reason is that they could only have been visited as part of the process of visiting some shadow node m with $m = S(n)$. But this would mean that we have already seen some other node on the path p at the same level in T as n, which is impossible, since n is the only node on p at that level. Hence, when an n-cluster is entered from above, we need not provide p' with the ability to skip n^{fs} and n^{ts}. It must only be able to visit all of n's ten internal nodes, with or without taking a detour to visit $S(n)$, and to leave from "below", via n^d. Fig. 7 shows how this is done.

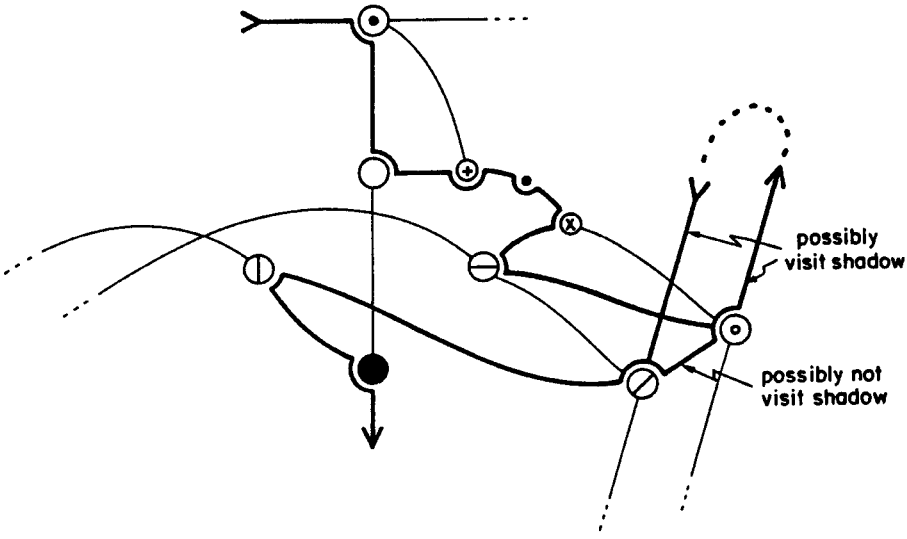

Fig. 7. Visiting a node that is on p in G_3. Here we might have to visit $S(n)$ too, leaving via n^{ts} and returning via n^{fs}.

Turning to $n \notin p$, here our path p' enters n from a pre-shadow, via n^r, but it must be able to visit the internal nodes in any of the four combinations resulting from having to visit or skip the node n^{uu} and the pair of nodes n^{ts} and n^{fs}. (We never have to visit just *one* of n^{ts} or n^{fs}.) Fig. 8 shows all four possibilities. Given these descriptions, it should be clear that p' is Hamiltonian.

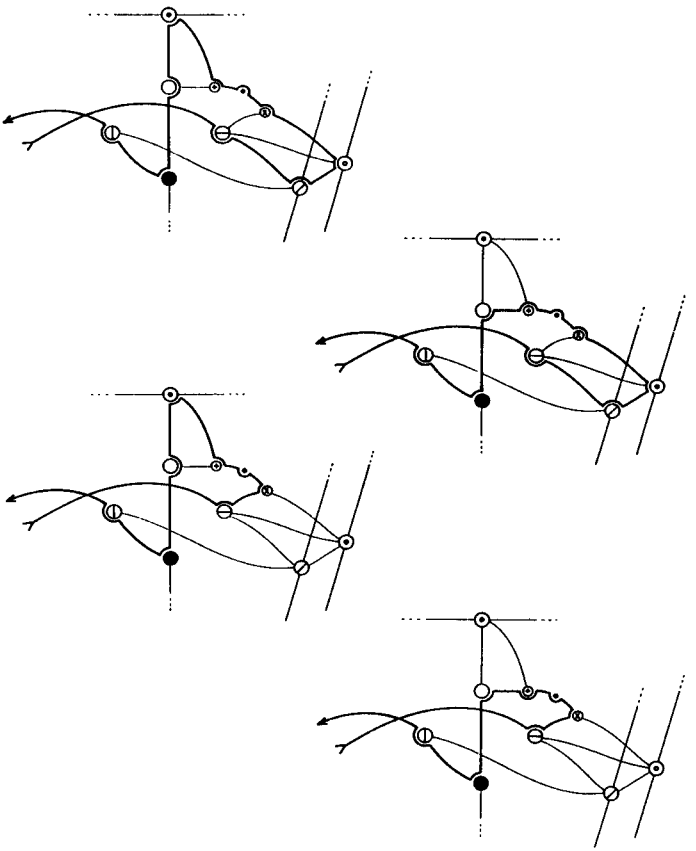

Fig. 8. Four ways of visiting a node as a shadow in G_3.

(*If*) Let p be a Hamiltonian path in G_3. Let n' be n's parent node in T, and let pre(n) be n_1^{uu}, where n_1 is the largest offspring of n' that is still smaller than n, if there is such an offspring, and $(n')^d$ if n itself is the smallest offspring of n'. Also, let post(n) be n_2^{uu}, where n_2 is the smallest offspring of n' which is still larger than n, and undefined if n itself is the largest offspring of n'.

CLAIM: For any n, if p contains the edge pre(n) — n^{uu}, then it must also contain either the edge n^{uu} — post(n) or the edge n^d — m^{uu}, where m is n's smallest offspring in T.[9]

Proof: (Refer to Fig. 6 to follow the arguments.) Assume that p contains

9 In words, if n is entered from above, then it is left from below.

pre(n) — n^{uu} but not n^{uu} — post(n). We have to show that p contains n^d — m^{uu}.

Since p must contain exactly two edges incident to any node (except the root g), it must contain one of n^{uu} — n^u or n^{uu} — n^{us}. We claim that the first of these must be the case. Here is why: Assume that n^{uu} — n^{us} is in p. Since n^{ur} has but two incident edges, both have to be in p, including n^{ur} — n^{us}. Hence, we have already accounted for the two incident edges of both n^{uu} and n^{us}; namely, pre(n) — n^{uu} — n^{us} — n^{ur}. Turning our attention to n^u, we now note that two of n^u's three incident edges have been eliminated as candidates for p. An impossibility. Thus, n^{uu} — n^{us} cannot be in p, so that n^{uu} — n^u must.

However, this means that n^u — n^{us} must also be in p, because it is one of the only two remaining edges incident to n^{us} (the third, n^{uu} — n^{us}, was just eliminated). We have thus established that the two edges contributed to p by n^u are n^{uu} — n^u and n^u — n^{us}.

Now consider n^d. It has three incident edges only: n^d — n^u, n^d — n^l, and n^d — m^{uu}, which is the edge required by the Claim.[10] The first of these cannot be in p, because we have just accounted for n^u's two edges in p; hence, the other two must. ∎

To complete the proof of the Lemma, note that p cannot contain an infinite subpath of the form n^d — m_1^{uu} — m_2^{uu} — m_3^{uu} — ⋯ The reason is that, since we are talking about a one-way Hamiltonian path, prior to getting "stuck" in this infinite sequence of uu nodes, p would have been able to visit at most finitely many clusters corresponding to nodes of the infinite tree T. This, however, would violate the Hamiltonicity of p. Hence, at some point, p must branch off the infinite uu path, and enter one of n's offspring, say, m_i. However, the Claim now implies that p will contain the edge leading down from m_i to the linked-list representing its offspring in T. Applying this argument inductively yields an infinite path in T, as required. ∎

We must now address the issue of recursiveness of neighborhoods. In way of illustrating the subtlety of this, consider, for example, the node n^{ts}, for some $n \neq 0$. We must be able to compute its set of neighbors from n, possibly using

10 Actually, if n is a leaf of T it will not have the third of these edges at all. Thus, the argument establishes that if n is entered from above this cannot be the case, and n must have offspring in T.

the Turing machine M_T that represents the **parent-of** relationship of the tree T. However, there is no way to determine whether n^{ts} has a *large* neighbor, i.e., the ts node appearing beneath it in Fig. 5, since there might be only finitely many nodes on the same level as n in the tree T. In fact, owing to this problem (and also to similar ones involving the fs and uu nodes), G_3, as it is defined right now, can be shown *not* to be highly recursive, although it is of bounded, not merely finite, degree.

The simplest way to overcome this problem is to carry out the entire reduction only from infinite trees T with the following properties: (i) T is of unbounded depth, (ii) each node of T that is not a leaf has infinitely many direct offspring, and (iii) both the **parent-of** relation and the **leafhood** predicate (i.e., whether a node is a leaf) are recursive. It can be shown, using the techniques of [12], that determining whether such a tree has has an infinite path is still Σ^1_1-hard. Carrying out the reduction from such trees only, we can show that the graph G_3 resulting from the above construction is indeed highly recursive. For example, the aforementioned problem with n^{ts} disappears, since the fact that there are infinitely many nodes on each level of T implies that any node will have a large neighbor, and that neighbor can now be found by running through the nodes larger than n in ascending order until one is found that resides on the same level.

To complete the proof of Theorem 2, we now reduce the degree of G_4 from 5 to 3. To that end, for each n, replace the nodes n^{uu}, n^r, n^{fs} and n^{ts} by the clusters given in Fig. 9. The reader should be able to verify easily that all the ways of moving through these nodes that are required for the *Only-if* direction of Lemma 4 are indeed possible (use Figs. 7 and 8 for this), and, moreover, that we can move through them visiting but once each of the new nodes introduced by the replacements. As to the *If* direction, the proof of the Claim therein requires a slightly longer, but similar, line of reasoning, since the single node n^{uu} has been replaced by four nodes. The details are omitted. ∎

5. Discussion

A recursive graph can be viewed simply as a computable binary relation on $\mathcal{N}$. In a similar vein, we may define a recursive model, or a recursive database, simply as a finite tuple of recursive relations (not necessarily binary) over $\mathcal{N}$. We thus obtain a natural generalization of the notion of a finite database. Recursive models have been the subject of much work in classical model theory (see, e.g.,

the survey [11]). However, no work seems to have been done from the perspective of databases. This appears to be a fertile area for future research, and raises a multitude of questions concerning the computability and complexity of queries and update operations, and the power of appropriate query languages. The present paper seems to provide evidence that such questions might turn out to yield interesting results.

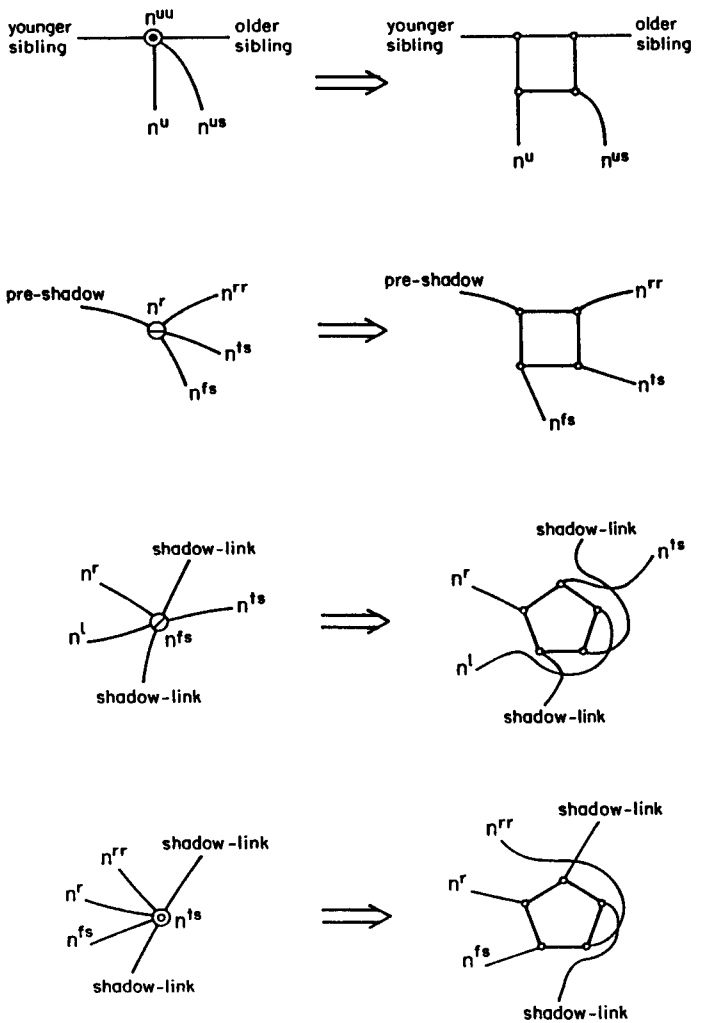

Fig. 9. Reducing the outdegree of G_3 to 3.

ACKNOWLEDGEMENT: Sincere thanks go to Richard Beigel, who introduced me to the problem. In addition, he and Bill Gasarch also claimed (correctly, as it turns out) that Hamiltonicity seemed as though it should be Σ_1^1-complete. Richard and Bill were also very helpful in providing background information and references.

References

1. R. Aharoni, M. Magidor and R. A. Shore, *On the strength of König's duality theorem for infinite bipartite graphs*, manuscript, 1990.
2. D. R. Bean, *Effective coloration*, J. Sym. Logic **41** (1976), 469–480.
3. D. R. Bean, *Recursive Euler and Hamiltonian paths*, Proc. Am. Math. Soc. **55** (1976), 385–394.
4. R. Beigel and W. I. Gasarch, unpublished results, 1986–1990.
5. 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.
6. S. A. Burr, *Some undecidable problems involving the edge-coloring and vertex coloring of graphs*, Disc. Math. **50** (1984), 171–177.
7. 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.
8. W. I. Gasarch, Personal communication, 1991.
9. D. Harel, *Effective transformations on infinite trees, with applications to high undecidability, dominoes and fairness*, J. Assoc. Comput. Mach. **33** (1986), 224–248.
10. A. Manaster and J. Rosenstein, *Effective matchmaking (recursion theoretic aspects of a theorem of Philip Hall)*, Proc. London Math. Soc. **3** (1972), 615–654.
11. 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.), Am. Math. Soc., Providence, R. I., 1985, pp. 323–375.
12. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.