Effective Transformations on Infinite Trees, with Applications to High Undecidability, Dominoes, and Fairness

DAVID HAREL

The Weizmann Institute of Science, Rehovot, Israel

Abstract. Elementary translations between various kinds of recursive trees are presented. It is shown that trees of either finite or countably infinite branching can be effectively put into one–one correspondence with infinitely branching trees in such a way that the infinite paths of the latter correspond to the "φ-abiding" infinite paths of the former. Here φ can be any member of a very wide class of properties of infinite paths. For many properties φ, the converse holds too. Two of the applications involve (a) the formulation of large classes of highly undecidable variants of classical computational problems, and in particular, easily describable domino problems that are Π_1^1-complete, and (b) the existence of a general method for proving termination of nondeterministic or concurrent programs under any reasonable notion of fairness.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Program Verification—*correctness proofs*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*logics of programs*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*computability theory; recursive function theory*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*decision problems*

General Terms: Langauges, Theory, Verification

Additional Key Words and Phrases: Domino problems, dynamic logic, fairness, high undecidability, recurrence lemma, recurring dominoes, recursive trees

1. *Introduction*

In this paper we establish elementary recursive one–one reductions between various kinds of infinite trees. Since the main appeal of these transformations is in their corollaries, the paper is structured in a way that presents much of the background and technical preliminaries for the applications before touching upon the results themselves.

In Section 2 we discuss Wang's dominoes [36] and describe the more recent bounded versions of Lewis [24] and van Emde Boas [35]. Both kinds of prob-

Preliminary versions of portions of this paper have appeared as "A simple highly undecidable domino problem," in *Proceedings of the Conference on Logic and Computation* (Clayton, Victoria, Australia, Jan.) 1984, and "A general result on infinite trees and its applications," in *Proceedings of the 16th ACM Symposium on the Theory of Computing* (Washington, D.C., Apr. 30–May 2). ACM, New York, 1984, pp. 418–427.

Author's address: Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel.

lems are among the clearest and most easily describable combinatorial problems suitable for exhibiting "bad" computational behavior, such as NP-hardness or undecidability.

In Section 3 we describe the two levels of undecidability relevant to the paper: the low Σ_1^0/Π_1^0 (recursively enumerable (r.e.)/co-recursively enumerable (co-r.e.)) level and the high Σ_1^1/Π_1^1 (coinductive/inductive) level, and indicate their classical recursive-well-founded-trees characterization.

In Section 4 the current research situation of the (seemingly unrelated) area of fair computations is described. The main line of research is the search for semantically complete methods for proving termination of nondeterministic or concurrent programs under increasingly more complex notions of fairness.

Sections 5 and 8 present the transformations on recursive trees. In the first of these, a "recurrence lemma" is proved, establishing the recursive isomorphism of well-founded ω-trees and recurrence-free finitely-branching trees. A recurrence is an infinite path containing infinitely many "marks." This thin/fat tree correspondence is then used in Section 6 to obtain simple highly undecidable domino problems, as well as similar variants concerning Turing machine computations and Post correspondences. In particular, deciding whether a set of dominoes T can tile the plane with a certain $d \in T$ occurring infinitely often is shown to be Σ_1^1-complete. An application of these to bounding validity problems from below appears in Section 7.

In Section 8 a considerable strengthening of one of the directions of the recurrence lemma is proved, showing that one–one effective transformations on ω-trees exist, mapping "φ-abiding" paths onto infinite paths. Here φ is any member of a broad class of properties of infinite paths, describable in a language containing infinitary quantifiers and infinitary Boolean connectives. Corresponding results concerning Turing machines, dominoes, and Post correspondences follow. Section 9 then applies the results of Sections 5 and 8 to the area of fair computations for nondeterministic or concurrent programs. The connections yield a generic proof method for the termination of programs under almost any conceivable notion of fairness. This solves, in a certain technical sense, some open problems in the literature on fairness.

Some of the corollaries to our tree transformations can be obtained by more or less standard techniques in recursion theory. Despite this, the uniformity and elementary nature of the present approach, coupled with the seemingly diverse applications, seem to indicate that marked recursive trees, the central mathematical objects of this paper, deserve more attention than they have received heretofore.

2. *Dominoes*

Domino or tiling problems, introduced by Wang [36] (with related independent work done by Buchi [7]), appear repeatedly in the literature, mainly in connection with negative results of undecidability or NP-completeness. The general setting of such problems is the integer grid $G = Z \times Z$ and certain notched tiles or colored dominos with which G or parts of it are to be tiled. We concentrate in this paper on colored dominos.

Geometrically, a domino is a 1×1 square, fixed in orientation, with edges parallel to the axes of G, and with colors associated with its edges. The *type* of a domino is the set of its four colors in the order, say, (*left*, *right*, *up*, *down*). Given a finite set T of domino types and a portion P of the grid G, we say that T *can tile* P if it is possible to cover P using only dominos of the types appearing in T, such that adjacent edges are monochromatic. A *domino problem* is a decision problem

characterized by the portion P to be tiled by the input set T, and often by some additional constraints on the placement of certain dominos or colors.

Domino problems considered in the literature can be divided into two classes, *unbounded* and *bounded*. The following are perhaps the "cleanest" versions in each class:

U1: Given T, can T tile G?
B1: Given T and n (in unary), can T tile an $n \times n$ subgrid of G?

Problem U1 is undecidable (cf. Berger [5], Robinson [31] and Lewis [25]). It is, in fact, co-r.e., and hence, using the hierarchy notation of Rogers [32], it is Π_1^0-complete. Problem B1 is NP-complete (cf. Lewis and Papadimitriou [26] and van Emde Boas [35]).

This behavior generalizes as one runs through some of the other problems that have been considered:

U2: Given T and $d \in T$, can T tile the positive quadrant G^+ of G such that d occurs at the origin?
U3: Given T and $T' \subseteq T$, can T tile the positive quadrant G^+ of G such that all dominos on the diagonal are from among T'?
B2: Given T, n (in unary) and two colors c_0, c_1, can T tile an $n \times n$ subgrid of G such that the leftmost colors of the top and bottom of the tiling are c_0 and c_1?
B3: Given T, n (in unary) and two colors c_0 and c_1, can T tile some $m \times n$ subgrid of G such that the leftmost top and bottom colors are c_0 and c_1?

Problems U2 and U3 are both Π_1^0-complete (cf. Wang [37] and Kahr et al. [20]), as are many variants with octants or half-grids replacing quadrants, color constraints on rows or columns instead of diagonals, restrictions on the appearance of combinations of colors and/or dominos rather than single dominos, etc. Problem B2 is NP-complete and B3 is PSPACE-complete (cf. van Emde Boas [35]); many similar-bounded variants are complete in these and in other complexity classes.

Domino problems are geometrically very appealing and are of sufficiently simple combinatorial character to make them easily describable and formalizable. They thus serve as excellent candidates for reductions in proofs of "bad" computational behavior, for example, undecidability or NP-completeness. Indeed, unbounded variants have been used in [15a], [20], [25], and [37] and elsewhere to prove undecidability of various subclasses of the predicate calculus, and in [35] to prove undecidability of the solvability of exponential Diophantine equations. Bounded variants have been used in [24], [26], and [35] to show NP-completeness of various problems, including the original NP-complete problem of satisfiability in the propositional calculus. In [33] and [35], van Emde Boas makes a strong case for adoption of both kinds of domino problems as "master-reduction" problems for use in such cases.

The main idea used in establishing lower bounds on the complexity of domino problems involves designing the domino sets T in such a way as to force tilings to correspond to legal computations of Turing machines (TMs), with successive rows of dominos representing successive configurations. The horizontal axis thus represents space and the vertical one time. It is not too hard, for example, to establish the undecidability of problem U2: the *origin constraint*, as it is sometimes called, can be used to force the first row to encode a start configuration on a blank tape, and then the positive quadrant G^+ can be tiled iff the given TM does not halt. (As it turns out, U1 is considerably more subtle, since there is no apparent way to force the appearance of any fixed part of any configuration, not even the very existence

of a state. See [5] and [31].) The bounded versions are proved complete for their complexity classes by similarly considering time- or space-bounded computations. So much for background on domino problems.

3. *Two Levels of Undecidability*

Although there exist many different levels of undecidability, there seem to be mainly two that stand out as being fundamental and naturally occurring: The Σ_1^0/Π_1^0 (i.e., the r.e./co-r.e.) level, and the Σ_1^1/Π_1^1 (sometimes called the coinductive/inductive) level. The former is the first level of the arithmetical hierarchy and is characterized by formulas over arithmetic with one number quantifier and a recursive matrix, and the latter is the first level of the analytical hierarchy, characterized by formulas over arithmetic with one function (or predicate) quantifier and an arithmetical matrix.

We do not attempt to convince the reader of the special role these two levels play by a review of undecidability results in general. However, we do point out that the theoretical computer science community has repeatedly seen examples of undecidable problems that, with few exceptions, turn out to be actually on one of these two levels. A striking example is in the field of logics of programs where numerous entirely different-looking logical systems have been shown in recent years (by almost as many methods) to have Π_1^1-complete validity problems. Some examples are first-order dynamic logic, context-free propositional dynamic logic, two-dimensional temporal logic, and global process logic (see [16]).

There have been essentially two ways of viewing the Σ_1^1/Π_1^1 level of undecidability, and these have served almost exclusively as the bases of proofs of Π_1^1- or Σ_1^1-hardness in the past. The first is simply to consider the syntactic form of formulas over $\mathbb{N}$ used to define sets on this level. For example, $\exists f\, \forall x R$ and $\forall f\, \exists x R$ are normal forms for Σ_1^1 and Π_1^1 sets, respectively, where f ranges over total functions from $\mathbb{N}$ to $\mathbb{N}$ and R is recursive. The second way involves infinite trees and is the approach we are interested in here.

Informally, Π_1^1 is generally associated with the set of *recursive well-founded ω-trees*, that is, with computable trees W with (at most) ω-branching, containing no infinite paths. Here we take "computable" to mean that there is an effective procedure for determining whether or not a given potential node is in W, and if it is, then whether or not it is a leaf. The set of (notations for) such trees is Π_1^1-complete. Accordingly, the set of non-well-founded recursive ω-trees, that is, those that contain at least one infinite branch, is Σ_1^1-complete. This characterization of the Σ_1^1/Π_1^1 level is central to Rogers' treatment of the subject [32, chap. 16] and has given rise to such well-known Π_1^1-complete sets as the set of (notations for) constructive ordinals [32] and the set of everywhere-halting programs that employ unbounded nondeterminism [2, 8].

The tree characterization is clearly in line with the analogous view of the Σ_1^0/Π_1^0 level: *recursive well-founded b-trees* (with b standing for bounded-branching) characterize Σ_1^0; these are simply the computation trees of NTMs, which do not diverge on some fixed input and are thus complete for r.e.

4. *Fairness*

Much effort has gone recently into the investigation of the behavior of nondeterministic or concurrent programs under the assumption of *fairness* (e.g., [1, 11, 12, 14, 15, 22, 23, 27–30]). In a nondeterministic program P with many possibilities (or *directions*) to choose from at certain points, an infinite computation is *fair* if

each direction is taken infinitely often; *P fairly terminates* if it admits no infinite fair computations, that is, if it always terminates assuming it acts fairly.

We refer in this paper to the following simple bidirectional nondeterministic program, cf. [9]:

$$\textbf{DO } A \rightarrow \alpha \,\square\, B \rightarrow \beta \textbf{ OD} \tag{1}$$

(= "repeatedly, if A is true, execute α; if B, execute β; if both, toss a coin to choose one; if neither, halt"). Here a direction is *enabled* in a state if its *guard* (A or B) is true, and it is *taken* if its *action* (α or β) is executed.

The main direction of research in this area (as is evident from Francez's encyclopedic survey [11]) is the search for semantically complete proof methods for fair termination under increasingly more complex notions of fairness, notably, those notions that take into account disabled directions and those that relativize fairness to given sets of states.

Here are three examples of the many notions of fairness that have been considered:

Weak fairness. An infinite computation is weakly fair if each direction that is enabled continuously from some point on is taken infinitely often.

Strong fairness. An infinite computation is strongly fair if each direction that is enabled infinitely often, is taken infinitely often.

Extreme fairness. An infinite computation is extremely fair if, for every first-order state formula p, if p is true infinitely often, each direction is taken infinitely often in states satisfying p.

For the first two notions, there are known complete proof methods for fair termination (cf. [1, 11, 15, 23, 27]), but for the third, introduced in [29] (as well as for a host of others (cf. [11, 30])), the problem has been left open.

One of the difficulties with devising semantically complete methods lies in the fact that the natural numbers do not suffice as the ordinals associated with fair termination. The two commonly used and closely related approaches for overcoming this are both connected with so-called *unbounded* nondeterminism, that is, with programs allowing assignments of the form $x \leftarrow ?$, setting x to any natural number. We describe one here.

Starting in [1] and later also in [3], [11], and [27], it is shown in this approach how to transform programs, such as program (1), that utilize bounded nondeterminism (bnd) into equivalent ones with unbounded nondeterminism (und), called *explicit schedulers*. The latter use the $x \leftarrow ?$ assignments to pick arbitrary finite priorities for scheduling the directions to be taken in the former, and terminate everywhere iff the original programs terminate fairly. Now, since there are complete methods for proving conventional termination of programs with und, albeit using all constructive ordinals (see [2], based on ideas of [6] and [8]), this yields a complete method for *fair* termination of programs with bnd.

As an example, the following are the explicit schedulers associated with program (1) by the methods of [1] and [11] for proving, respectively, weak and strong fair termination.

Weak fairness:

$$\begin{aligned} &a \leftarrow ?;\ b \leftarrow ?; \\ &\textbf{DO } (A \wedge (B \supset a \leq b)) \rightarrow (\alpha;\ a \leftarrow ?\ (\textbf{if } B \textbf{ then } b \leftarrow b - 1 \textbf{ else } b \leftarrow ?)) \\ &\quad \square\, (B \wedge (A \supset b < a)) \rightarrow (\beta;\ b \leftarrow ?\ (\textbf{if } A \textbf{ then } a \leftarrow a - 1 \textbf{ else } a \leftarrow ?))\ \textbf{OD}, \end{aligned} \tag{2}$$

Strong fairness:

$$a \leftarrow ?;\ b \leftarrow ?$$
$$\mathbf{DO}\ (A \wedge (B \supset a \leq b)) \rightarrow (\alpha;\ a \leftarrow ?\ (\mathbf{if}\ B\ \mathbf{then}\ b \leftarrow b - 1))$$
$$\square\ (B \wedge (A \supset b < a)) \leftarrow (\beta;\ b \leftarrow ?\ (\mathbf{if}\ A\ \mathbf{then}\ a \leftarrow a - 1))\ \mathbf{OD}. \tag{3}$$

The reader should be able to convince him/herself that program (2) (respectively, (3)) everywhere-terminates iff program (1) fairly terminates under weak (respectively, strong) fairness. As mentioned, the proof system of [2] can be used to prove ordinary termination of (2) or (3); that system is, in fact, complete, relative to an underlying μ-calculus-like language and might require any constructive ordinal in the proof.

Without going into the details of the other approach to proof methods for fair termination (represented, for example, by the results in [23]), we can say that it yields Floyd-like methods in which the prover is required to find some well-founded set and prove certain properties of the program with respect to that set. Showing completeness of such methods involves associating with the original bnd program a computation tree with *infinite* outdegree, and then using the ordinals corresponding to nodes in the tree as the well-founded set.

Upon reading the literature on fairness (see e.g., the extensive survey in Francez [11]), one gets the feeling that the connections, exposed by these methods and their completeness proofs, between the infinite paths of the computation trees of und programs and the fair infinite paths of those of bnd programs, are more fundamental, and that they should generalize.

5. *A Lemma on Infinite Trees*

The lemma we are about to prove illustrates a fundamental connection between the two kinds of infinite trees that were discussed above and that are defined below.

Definition 5.1. The *full ω-tree* is simply $\mathbb{N}^*$; it is just the set of finite sequences of natural numbers. The empty sequence λ is its *root*. For $u \in \mathbb{N}^*$ and $n, m \in \mathbb{N}$, the sequence $u \cdot n$ is an *offspring of* u, and $u \cdot n$ and $u \cdot m$ are *siblings*. For $u, v \in \mathbb{N}^*$ we say that u is a *descendant* of v, denoted $v \prec u$, if $u = v \cdot w$ for some w. An ω-tree is a subset W of $\mathbb{N}^*$ closed under $\prec$; that is, if $u \prec v$ and $v \in W$, then $u \in W$. A node with no offspring is a *leaf*. A *path* is a finite or infinite sequence $u_0, u_1, u_2, \ldots$ such that for each i, u_{i+1} is an offspring of u_i. An ω-tree W is *recursive* (*strongly recursive* in [32]) if its extended characteristic function χ_W is recursive, where, by definition,

$$\chi_W(u) = \begin{cases} 0, & u \notin W; \\ 1, & u \in W, \quad u \text{ is a leaf}; \\ 2, & u \in W, \quad u \text{ is not a leaf}. \end{cases}$$

An ω-tree is *well founded* if it contains no infinite paths. We subsequently associate a node u with its Gödel encoding, and a recursive ω-tree W with an encoding of some TM that computes χ_W.

Definition 5.2. For any $k > 0$, let $\bar{k} = \{0, 1, \ldots, k-1\}$. The *full k-tree* is simply $\bar{k}^*$, and a *k-tree* is a subset of $\bar{k}^*$ closed under $\prec$. Other tree notions are defined for k-trees just as for ω-trees. A *b-tree* U (b for *b*ounded) is a k-tree for some k (the smallest such k being the *width* of U); U is a *marked b-tree* if some of its internal (i.e., nonleaf) nodes are marked. A *recursive* marked b-tree U is one for which the

following function is recursive.

$$\Omega_U(u) = \begin{cases} 0, & u \notin U; \\ 1, & u \in U, \quad u \text{ is marked}; \\ 2, & u \in U, \quad u \text{ is not marked}. \end{cases}$$

(Note that leafhood here can be checked recursively too by testing the presence of any of a node's k offspring.) A *recurrence* in a marked b-tree is an infinite path containing infinitely many marked nodes; the tree is *recurrence free* if it contains no recurrences. Here we associate a recursive marked b-tree U with the pair consisting of its width and some TM that computes Ω_U.

INFINITE TREE RECURRENCE LEMMA. *The set A of recursive well-founded ω-trees and the set B of recursive marked recurrence-free b-trees are recursively isomorphic.*

PROOF. By a result of Myhill [32, theorem 7.VI], it suffices to show that A and B are one-one equivalent, denoted $A \equiv_1 B$; that is, that there are recursive 1–1 (but not necessarily onto) functions $g: A \to B$ and $h: B \to A$ with $\forall W\,(W \in A$ iff $g(W) \in B)$ and $\forall U\,(U \in B$ iff $h(U) \in A)$.

First, to reduce A to B, let $1^0 = \lambda$ and $1^{n+1} = 1^n \cdot 1$. Define $\gamma: \mathbb{N}^* \to \{0, 1\}^*$ by $\gamma(\lambda) = \lambda$, $\gamma(u \cdot n) = \gamma(u) \cdot 0 \cdot 1^n$. Now, given a tree $W \in A$, define the tree $U = g(W)$ by

$$\Omega_U(u) = \begin{cases} 1 & \text{if } u = \gamma(v \cdot 0), \quad \chi_W(v) = 2; \\ 2 & \text{if } u = \lambda, \quad \text{or} \\ & \text{if } u = \gamma(v \cdot n), \quad n > 0, \quad \chi_W(v) = 2, \quad \text{or} \\ & \text{if } u = \gamma(v \cdot 0), \quad \chi_W(v) = 1; \\ 0 & \text{if none of the above.} \end{cases}$$

It is easy to see that U is a 2-tree, and that Ω_U is recursive whenever χ_W is.

A pictorial illustration of the construction of $g(W)$ is given in Figures 1 and 2, and an "operational" way of viewing it is to traverse W, branching for each nonleaf of W downward to the (possibly absent) 0-offspring, and then indefinitely rightward from one (possibly absent) sibling to its immediate next. For a leaf of W only the downward step is made. Marked nodes are 0-offspring of nonleaves. This construction is a generalization of the "natural correspondence between [finite] forests and binary trees" of Knuth [21, p. 333].

The reader can easily verify that W contains an infinite path iff $g(W)$ contains a recurrence, so that we are left with having to show that g is 1–1. Indeed, let $W \neq W'$. Then, without loss of generality, either $W = \emptyset$ and $W' \neq \emptyset$ (in which case, by definition, $g(W) = \{\lambda\}$ but $0 \in g(W')$), or else there is some node $u \cdot n \in W'$, with $u \in W$ but $u \cdot n \notin W$. But then $\gamma(u \cdot n \cdot 0) \in g(W')$, whereas $\gamma(u \cdot n \cdot 0) \notin g(W)$. Thus g is 1–1, and hence $A \leq_1 B$.

To reduce B to A, let there be given a sequence $u \subseteq \overline{k}^*$. Define $u^{(k)}$ to be its index in the "breadth-first" ordering of $\overline{k}^*$, defined so that u is smaller than u' iff either $|u| < |u'|$ or $|u| = |u'|$ but u is lexicographically smaller than u'. This ordering corresponds to traversing $\overline{k}^*$ level by level starting each level from the "left"; see Figure 3. Clearly $u^{(k)}$ can be found effectively from u, and vice versa.

Given a marked b-tree U of width k, the ω-tree $W = h(U)$ is constructed as follows (see Figures 3 and 4)). The root has exactly k offspring $0, \ldots, k-1$, to indicate the breadth, and the main part of the tree is rooted at the node 0. We need the following notion: The *slice* of U at a node u is defined to be the subtree of U rooted at u (excluding u) with all descendants of marked nodes (within that

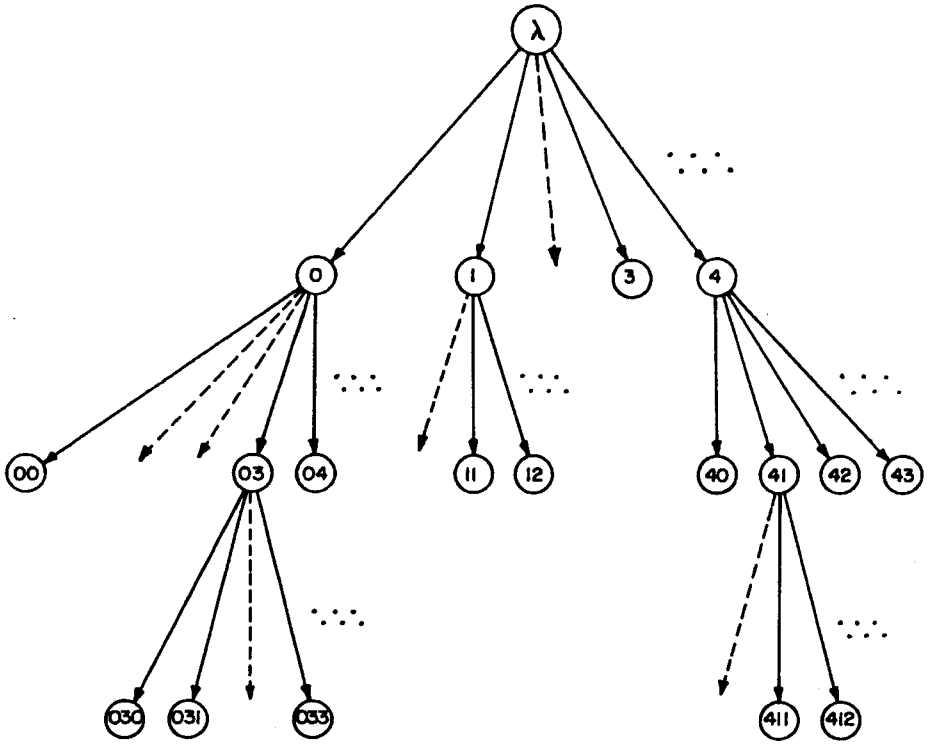


FIG. 1. Part of an ω-tree W. Absent nodes are denoted by broken arrows.

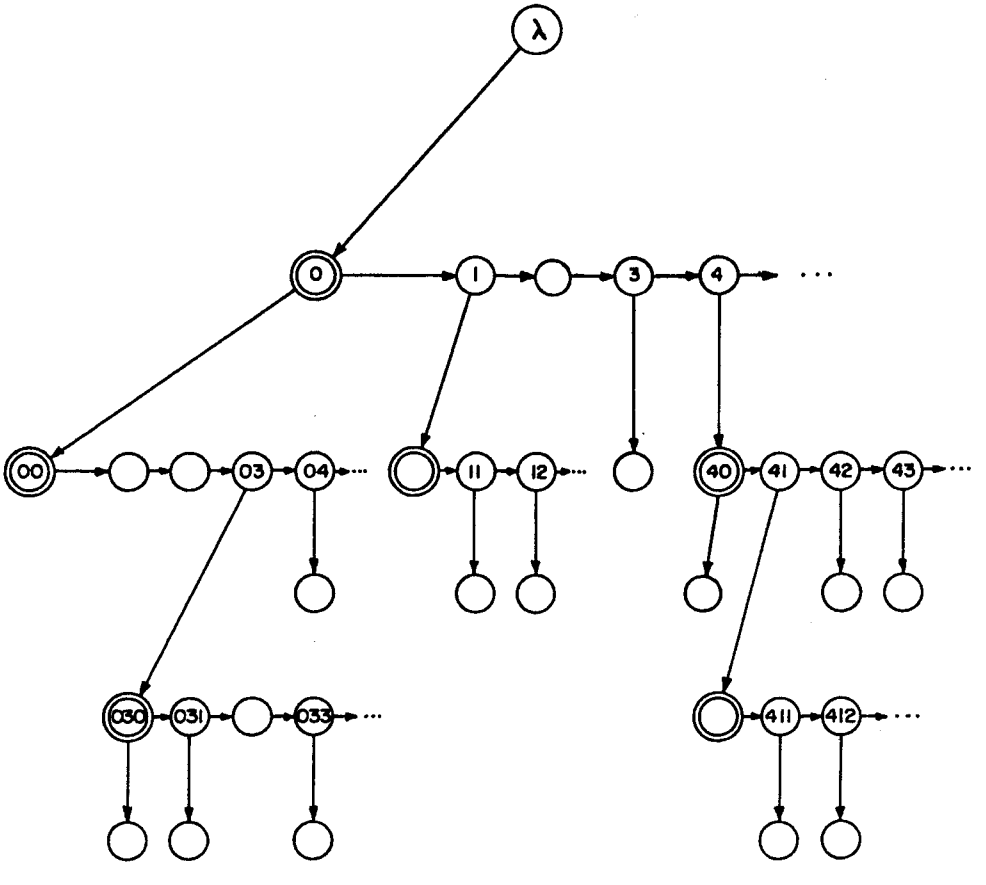


FIG. 2. Part of the 2-tree $g(W)$. Marked nodes are doubly circled.

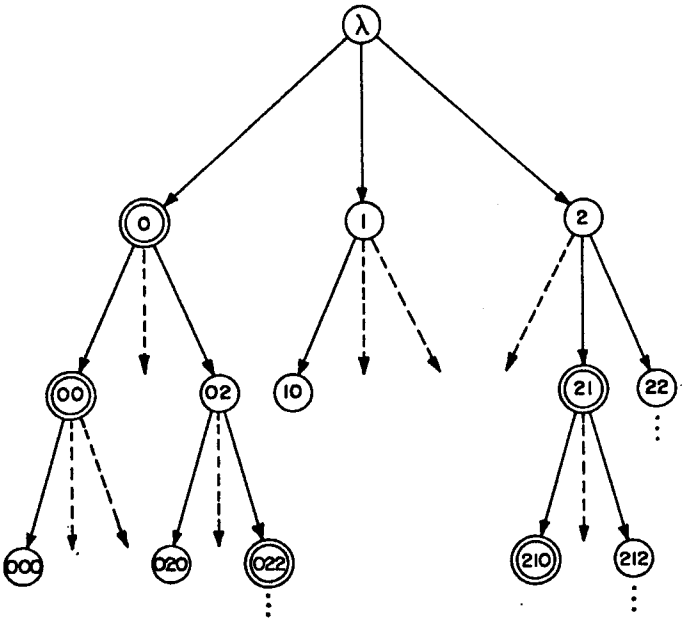


FIG. 3. Part of a 3-tree U. Marked nodes are doubly circled.

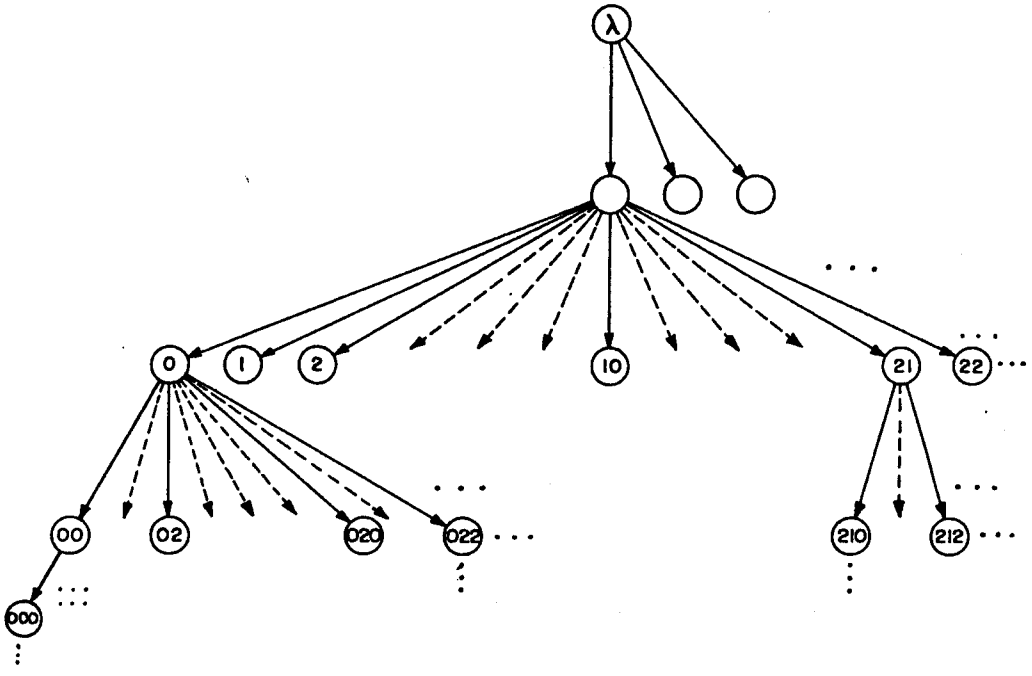


FIG. 4. Part of the ω-tree $h(U)$.

subtree) deleted. In other words, the slice contains all nodes reachable from u without moving below a marked node. As shown in Figure 3, the slice of U at λ includes 0, 1, 10, 2, 21, and 22 and excludes the subtrees rooted at 0 and 21. The slice at 0 includes 00, 02, 020, and 022.

Returning to the construction of W, the slice of U at λ is "attached" to the virtual root 0 of W, in linearized breadth-first order; that is, node u of that slice is

associated with node $0 \cdot u^{(k)}$ of W. If u is unmarked in U, then $0 \cdot u^{(k)}$ is a leaf of W, and if u is marked, then the slice of U at u is considered and similarly "attached" to $0 \cdot u^{(k)}$ in W. This process is repeated indefinitely for all marked nodes of U. Clearly W is an ω-tree.

To see that it is also recursive, consider a potential node $u = (0, n_1, \ldots, n_j) \in \mathbb{N}^*$. (The case where the first component is nonzero is trivial.) To compute $\chi_W(u)$, one uses k to calculate the unique v_1 such that $v_1^{(k)} = n_1$. If $\Omega_U(v_1) = 0$, then $\chi_W(u) = 0$; otherwise, the ancestors of v_1 are checked to determine whether any one of them was marked in U. If so, v_1 is not in the slice of U at λ; hence $\chi_W(u) = 0$. If no ancestor was marked and if $j \neq 1$ and $\Omega_U(v_1) = 2$, then $\chi_W(u) = 0$. Otherwise, attention is shifted to n_2. Again v_2 is found in the slice at v_1, with $v_2^{(k)} = n_2$ and a similar procedure is carried out, etc. Finally, if n_j is reached with all such tests having positive results, $\chi_W(u)$ is set to 2 or 1, depending on whether the most recently considered node in U was marked or not.

Here too, it is straightforward to see that U contains a recurrence iff W contains an infinite path. To see that h is 1–1, one observes that the $0, \ldots, k-1$ offspring of λ in W determine the width, and given k, the rest of W uniquely determines the structure of U. □

This "fat/thin tree" correspondence lemma was stated informally at the end of [16]. A very similar result, though with different motivation and applications, was proved independently by Arnold [4, proposition 3.4].

COROLLARY 5.3. *The set of (notations for) recursive marked recurrence-free b-trees is* Π_1^1*-complete.*

PROOF. Follows immediately from the Π_1^1-completeness of the set A [32, theorem 16.XX] and the lemma. □

6. *Recurring Dominos*

We first derive a machine-oriented version of the Recurrence Lemma. Let C be the set of (notations for) NTMs, which on a blank tape admit no infinite computations that infinitely often enter the start state q_0.

LEMMA 6.1. $A \equiv C$. (*Here "$\equiv$" stands for recursive isomorphism.*)

PROOF. Consider the proof of $A \leq_1 B$ in the previous section. The construction of $g(W)$ therein can be thought of as traversing W with an NTM M, which nondeterministically decides whether to proceed to the leftmost offspring or the right-hand-side sibling of a node and signals by entering q_0 whenever the former option is chosen. The machine M can carry out all necessary tests (membership and "leafship" in W) by appealing to the TM defining W. This establishes $A \leq_1 C$.

On the other hand, the computation tree (on a blank tape) of a machine M in C is just a b-tree whose width is bounded by the number of states in M. This observation can be used to show $C \leq_1 B$, by marking nodes representing a configuration of M whose state is q_0, and by including an appropriate encoding of the NTM in some "harmless" portion of the resulting tree. Hence, $A \equiv B \equiv C$. □

COROLLARY 6.2. *C is* Π_1^1*-complete.*

Variants of this corollary have already been established in [10], [13], [17], and [34]. We subsequently use the corollary in the form that states that the following decision problem is Σ_1^1-complete, where M ranges over NTMs.

C1: Given M, does M, when started on a blank tape, admit an infinite computation that reenters its start state q_0 infinitely often?

Consider now the following simple *recurring domino problem*:

R1: Given T and $d \in T$, can T tile G with d occurring infinitely often in the tiling?

THEOREM 6.3. *R1 is Σ_1^1-complete.*

PROOF. To see that R1 is in Σ_1^1, let T and $d \in T$ be given. Construct a NTM M that starts on a blank tape by initially constructing the empty 0×0 tiling of G. At each step, M considers the next position in G in a spiral movement around the current tiling and nondeterministically tries to tile that position with some domino from T. M rejects if colors fail to match, and signals a successful use of the domino d by reentering q_0. Otherwise, it never enters q_0. Clearly, M has property C1 iff the pair (T, d) has property R1.

To show that R1 is Σ_1^1-hard we proceed in three steps. Consider first the following:

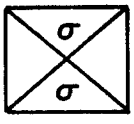
R2: Given T and $d \in T$, can T tile the positive quadrant G^+ of G with d occurring infinitely often and with the borderlines colored white?

CLAIM 1. *R2 is Σ_1^1-hard.*

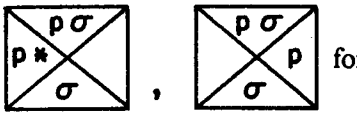
PROOF OF CLAIM. Consider C1′ in which the recurring activity of M is its entering from the right onto a blank cell and into state q_0. This is easily shown to be Σ_1^1-hard by appropriately changing the signaling activity of the machines in the first part of the proof of Lemma 6.1.

We now construct, for each M, a domino set T such that (T, d) has property R2 iff M has property C1′. This is very similar to standard practice in the domino literature. Let M be given, with B the blank symbol, p and q ranging over states, σ and τ ranging over tape symbols, and the transition table given in the form of quantuples (q, σ, R, τ, p) and (q, σ, L, τ, p). T is constructed to consist of the following groups of dominos, with colors taken to be combinations of symbols:

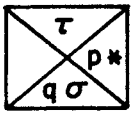
(0) 

(1) 

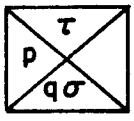
for each σ;

(2) 

for each p, σ;

(3) 

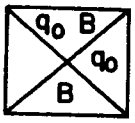
whenever (q, σ, R, τ, p) is present;

(4) 

whenever (q, σ, L, τ, p) is present.

Dominos from group (0) are the only ones possible along the bottom row of G^+ owing to the white boundary constraint; they force the initial $q_0BB\ldots$ configuration. Those from group (1) allow propagation of unaltered tape symbols from one

configuration to the next, and dominos from groups (2), (3), and (4) combine to transfer state movement and state/tape changes according to the rules of M. The * symbol in (2) and (3) prevents the two dominos of (2) from combining to create a new tape head. Consequently, each row has precisely one occurrence of a state. Thus G^+ can be tiled with T adhering to the boundary condition iff M admits an infinite computation starting at $q_0BB\ldots$. Accordingly, with d taken to be



(T, d) satisfies R2 iff M satisfies Cl′. This completes the proof of Claim 1. □

Now consider the following, in which there is no boundary condition:

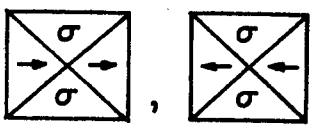
R3: Given T and $d \in T$, can T tile G^+ with d occurring infinitely often?

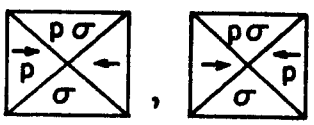
CLAIM 2. *R3 is* Σ_1^1*-hard.*

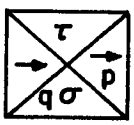
PROOF OF CLAIM. First, one notes that the boundary condition above was used to force the initial $q_0BB\ldots$ configuration, which in turn forced at least (and at most) one state per row. Here we consider the following machine problem:

C2: Given M, is there some tape upon which M can be started in some state at some location, so as to admit an infinite computation that reenters q_0 from the right onto a blank cell infinitely often?

C2 is easily seen to be Σ_1^1-hard since the machines in the proof of Lemma 6.1 can be made oblivious of everything that happens from their initial configuration up to the first time (if any) that they enter q_0. Now we slightly change the construction of T from M by adopting the following domino groups:

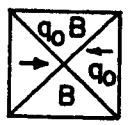
(1)  for each σ;

(2)  for each p, σ;

(3)  whenever (q, σ, R, τ, p) is present;

(4)  whenever (q, σ, L, τ, p) is present.

Here d is



.

The point is that an appearance of any state in any row in the tiling forces a $\cdots \rightarrow\rightarrow\leftarrow\leftarrow\leftarrow \cdots$ pattern of the arrow symbols on that row and thus prevents the appearance of more than one state along the row. The tiling mechanism forces

the neighboring rows to contain a state too. Thus d occurring even once forces exactly one state per row, and consequently (T, d) satisfies R3 iff M satisfies C2. This completes the proof of Claim 2. □

To complete the proof of the theorem, we have to extend these ideas from G^+ to G. The extension to the positive half-grid poses no problem; the NTMs are simply thought of as operating on two-way infinite tapes and the $\cdots \rightarrow\rightarrow\leftarrow\leftarrow\leftarrow \cdots$ tiling pattern is extended to the left. However, the presence of the bottom half-grid poses two (dual) problems: (i) the possibility that M admits no infinite "backward" computations, so that the tilings as described above will not be downward extendible, and (ii) the possibility that a tiling exists in which d occurs infinitely often *downward*, but appears nowhere above some given row. This would correspond to some infinite recurring backward computation of M, but to no forward computation.

Problem (i) is easily solved by adding to the machines M of Lemma 6.1 some fixed trivial backward loop not involving q_0. This will enable a trivial backward tiling of G. To solve problem (ii), the machines of Lemma 6.1 are modified as follows. There is a second track on the tape upon which there will be a positive integer at all times. Whenever the machine is about to enter its signaling situation (depicted by d), it first checks deterministically that this track indeed contains such an integer and then increases it by 1. A tiling of G with infinitely many appearances of q_0 in a *lower* half-grid would entail an infinite backward computation that infinitely often verifies the presence of increasingly smaller positive integers—an impossibility. Thus, for these modified machines (which constitute an effectively computable subset of all NTMs), M satisfies C2 iff (T, d) satisfies R1. This completes the proof of the theorem. □

Many variants of R1–R3 are also Σ_1^1-complete. We mention three useful ones (cf. [16]) in particular:

R4: Given T and a color c, can T tile G with c occurring infinitely often?

R5: Given T and $d \in T$, can T tile G^+ with d occurring infinitely often in the first column?

R6: Given T and $d \in T$, can T tile the strict upper positive octant $G^{++} = \{(i, j) \mid 0 \leq i < j\}$ with d occurring at least once in each ith row–column combination $G_i = \{(j, i) \mid 0 \leq j < i\} \cup \{(i, j) \mid j > i\}$?

THEOREM 6.4. *R4–R6 are all Σ_1^1-complete.*

PROOF. Similar to the previous proof. In R4, color c is taken to be the symbol q_0; in R5, M is simply required to move to the extreme left before signaling; in R6, M is programmed to signal only after having checked that the least i for which the partial tiling of G_i does not yet contain d has just been (perhaps implicitly) increased.

The Σ_1^1-hardness direction of R6 requires additionally modifying the NTMs of Lemma 6.1 (first part), so that their tapes are extended by one square at each step (hence the *octant* being tiled), and so that the signaling is carried out at strictly increasing squares on the tape, skipping none. Details of this case are straightforward and are omitted. □

This is an appropriate place to describe a Σ_2^0-complete domino problem that is used in [16]. The problem is not as "clean" as the others and the idea does not seem to extend naturally to higher levels.

U*. Given T and two colors c_0, c_1, can T tile G^+ such that the sequence of colors on the bottom of the first row is of the form $c_0^n c_1^n$ for some n?

THEOREM 6.5. *U^* is Σ_2^0-complete.*

PROOF. The special form of colors on the bottom row is used to encode a start configuration of M in which the input is the nonnegative integer n in unary. Thus, (T, c_0, c_1) is shown to satisfy U* iff there is some input n upon which M does not halt. We omit details. □

We close this section by noting that along these lines one can define Σ_1^1-complete versions of the well-known Post Correspondence Problem [cf., 19, sec. 8.5]. For example, given n and two vectors $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$, where each of the x_i and y_i is a nonempty finite word over some finite alphabet Σ. Consider the problem of deciding whether there is some infinite sequence $i_1, i_2, \ldots$ of indices from among $\{1, \ldots, n\}$, containing the index 1 infinitely often, and such that the infinite words $x_{i_1}x_{i_2}\ldots$ and $y_{i_1}y_{i_2}\ldots$ are equal. It can be shown by methods similar to those used herein that this problem is Σ_1^1-complete.

7. *An Application of Recurring Dominoes*

In the interest of further promoting domino problems as a tool for exhibiting "bad" computational behavior, we have presented a detailed series of transparent proofs of all known Π_1^1-hardness results for validity in various logics of programs in [16]. Here we give a short example of this by repeating the three-part proof of the lower bounds on the complexity of the validity problems for propositional, predicate, and infinitary or dynamic logics.

In the following reductions let the input set $T = \{d_0, \ldots, d_m\}$, involve colors $\{c_0, \ldots, c_{k-1}\}$ where k is a power of 2. We employ symbols LEFT^u, RIGHT^u, UP^u, and DOWN^u, for $1 \le u \le \log k$, to stand for propositional variables or (binary) predicates as needed; in the propositional case we subscript these as in, for example, $\text{LEFT}^u_{i,j}$, for $1 \le i, j \le n$. By convention, the unsuperscripted versions of these symbols stand for the ordered and appropriately signed sets of the log k superscripted ones. In this way, identifying color c_l with the binary representation of l, we write, say, $\text{LEFT}_{i,j} = c_l$ to stand for the appropriate conjunction asserting that the $\text{LEFT}^u_{i,j}$, $1 \le u \le \log k$, encode color c_l; similarly for LEFT(x, y), etc. For a domino d_l, we write $\text{LRUD}_{i,j} = d_l$ to stand for the conjunction of $\text{LEFT}_{i,j} = \mathit{left}(d_l)$, $\text{RIGHT}_{i,j} = \mathit{right}(d_l)$, etc.; similarly for LRUD(x, y). Here, for example, $\mathit{left}(d_l)$ is the left color of domino d_l.

THEOREM 7.1

(*i*) *Satisfiability in the propositional calculus is NP-hard.*
(*ii*) *Satisfiability in the predicate calculus is Π_1^0-hard.*
(*iii*) *Satisfiability in both infinitary and (quantified) dynamic logic is Σ_1^1-hard.*

PROOF

(i) (cf. [26]). Given T and n, construct $P_{T,n}$ as the conjunction

$$\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{n}\left(\bigvee_{l=0}^{m}\text{LRUD}_{i,j} = d_l\right) \tag{4}$$

and

$$\bigwedge_{i=1}^{n-1}\bigwedge_{j=1}^{n}(\text{RIGHT}_{i,j} = \text{LEFT}_{i+1,j} \wedge \text{UP}_{j,i} = \text{DOWN}_{j,i+1}). \tag{5}$$

Clearly the size of $P_{T,n}$ is a polynomial in $n + m$, and it is satisfiable iff the pair T, n satisfies B1. The latter is seen by observing that (4) associates a domino from

T with each point of $[1 \cdots n] \times [1 \cdots n]$, and (5) asserts correct matching of colors.

(ii) Given T, construct φ_T as the conjunction of

$$\forall x(f(x) \neq z \wedge \forall y(f(x) = f(y) \supset x = y)), \tag{6}$$

$$\forall x \forall y\left(\bigvee_{l=0}^{m} \mathrm{LRUD}(x, y) = d_l\right), \tag{7}$$

and

$$\forall x \forall y(\mathrm{RIGHT}(x, y) = \mathrm{LEFT}(f(x), y) \wedge \mathrm{UP}(x, y) = \mathrm{DOWN}(x, f(y))). \tag{8}$$

The claim is that φ_T is satisfiable iff T satisfies U1. The *if* direction is trivial since, if a tiling of G exists, then a tiling of G^+ exists and φ_T is satisfied in $\mathbb{N}$ with z interpreted as 0 and f as successor. Conversely, the domain of any structure satisfying φ_T must contain, by clause (6), an infinite set S constituting the values of $z, f(z), f(f(z)), \ldots$. The grid G^+ matches $S \times S$, with (i, j) corresponding to $(f^i(z), f^j(z))$. Clauses (7) and (8) behave as in part (i), yielding a tiling of G^+. A well-known application of König's Lemma (see [31]) yields a tiling of G.

(iii) Given T, construct ψ_T as the conjunction of φ_T from part (ii) above and either

$$\forall x \bigvee_{i \in w} (\mathrm{LRUD}(z, f^i(x)) = d_0), \tag{9}$$

or

$$\forall x < (x \leftarrow f(x))^* > (\mathrm{LRUD}(z, x) = d_0), \tag{10}$$

for infinitary or dynamic logic, respectively. The claim is that ψ_T is satisfiable iff (T, d_0) satisfies R5. The *if* direction is as before, but now clause (9) or (10) holds by virtue of the recurrence of d_0. Conversely, if (9) or (10) holds, d_0 occurs arbitrarily high up in the first column $\{(z, f^i(z))\}_{i \in w}$ of G^+. □

Recurring domino problems thus serve as examples of highly undecidable, easily decribable combinatorial problems, and complete the picture of naturally occurring levels of difficulty: bounded = decidable, unbounded = weakly undecidable, recurring = highly undecidable.

8. *A Generic Transformation on Trees*

In this section the B to A direction of the Recurrence Lemma of Section 5 will be considerably strengthened. First, we modify the terminology somewhat. Marked trees will be more general, and markedship will apply to ω-trees too.

Let Σ be some fixed (possibly infinite) alphabet. A *marked tree* is one in which nodes are labeled with (possibly infinitely many) letters from Σ; that is, W comes complete with a marking predicate $M_W \subseteq W \times \Sigma$. A marked tree will be said to be *recursive* if it is a recursive tree and if, in addition, M_W is recursive. Let T, T^+, T_f^+, T_b^+ stand, respectively, for the sets of recursive ω-trees (i.e., with ω-branching), recursive marked ω-trees, recursive marked f-trees (i.e., with finite but possibly unbounded branching), and recursive marked b-trees (i.e., with finite and bounded branching).

We now define a language L for stating properties of infinite paths in marked trees. An *atomic formula* is an expression of one of the forms $\exists a$, $\forall a$, $\exists^\infty a$, or $\forall^\infty a$, where $a \in \Sigma$ is a mark. Define L_0 to be the set of atomic formulas. For each $i \geq 0$,

let L_i' be the closure of L_i under finite conjunctions and disjunctions, and under denumerable recursive conjunctions (i.e., if $\{\varphi_i\}$ is a recursive sequence of formulas of L_i', then $\bigwedge_i \varphi_i$ is in L_i'). L_{i+1} is taken to be the closure of L_i' under denumerable recursive disjunctions. Let $L = \bigcup_i L_i$. Here we talk about L with the convention that each formula $\varphi \in L$ is given together with the least n for which $\varphi \in L_n$. This n is called φ's *type*.

(Note: L is (superficially) similar to the "$\exists$fullpath" fragment of Emerson and Clarke's [10] language CTF, for which they provide a translation into fixpoint-theoretic terms.)

Informally, each $\varphi \in L$ is interpreted over a given infinite path p by interpreting $\exists a$ as "there is a node on p marked with a," and $\exists^\infty a$ as "there are infinitely many nodes on p marked with a"; $\forall a$ and $\forall^\infty a$ denote the appropriate duals. This meaning is then extended up through the Boolean and infinitary connectives.

For example, consider playing chess on an infinite board (but with the standard set of 32 pieces) where moving rules are generalized in some reasonable way. An infinitely long game is a draw iff both players call "check" infinitely often; otherwise, it is a win for the player with the most calls. The game tree can be regarded as an element of T^+ (or $T_b^\dagger$ if pieces are not allowed to move too far) with, say ① or ② marking nodes where player 1 or 2 checks, respectively. The draw criterion is then given simply by the formula of L: $\exists^\infty ① \wedge \exists^\infty ②$.

For $\varphi \in L$, an infinite path is said to be *φ-abiding* if it satisfies φ, and a tree is *φ-avoiding* if it has no φ-abiding infinite paths. Note that the recursiveness of markings and trees allows referring in effect to ancestors of nodes, as in the following (liberally formulated) formula of L,

$$\varphi: \exists^\infty a \wedge \bigwedge_i (\forall^\infty (\text{number of nodes between two most recent } a\text{'s from root} > f(i))),$$

for some recursive f. Here the φ-abiding paths have infinitely many nodes marked a and the distances between these "grow" in the special manner described. Clearly, each of the countably many right-hand conjuncts can be associated with a recusive mark.

Note that by definition a tree is *well founded* iff it is ($\exists^\infty$*true*)-avoiding, for the trivial everywhere-occurring mark *true*.

Another important special formula in L is $\exists^\infty \circledast$, where $\circledast$ is any fixed mark in Σ. A $\exists^\infty \circledast$-abiding path is simply a *recurrence*, and the recurrence lemma of Section 5 can be written as follows:

INFINITE TREE RECURRENCE LEMMA. *The set of well-founded trees in T is recursively isomorphic to the set of $\exists^\infty \circledast$-avoiding trees in $T_b^\dagger$.*

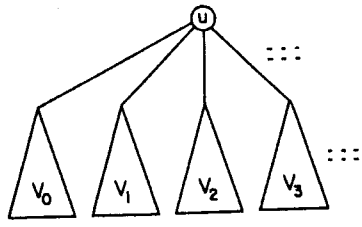
The main result of this section is

THEOREM 8.1. *Let φ be an arbitrary formula of L. The set of φ-avoiding trees in T^+ (and hence also those in $T_f^\dagger$ and $T_b^\dagger$) is one–one reducible to the set of well-founded trees in T.*

PROOF. We actually establish the following stronger claim:

(*) Let φ be an arbitrary formula of L. There is a recursive 1–1 function $\eta: T^+ \to T$ which, for each $W \in T^+$, induces a recursive transformation from the infinite paths of $\eta(W)$ *onto* the φ-abiding (infinite) paths of W.

FIGURE 5



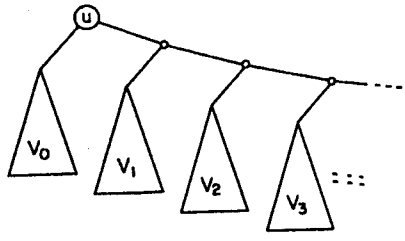


FIGURE 6

Note that the φ-abiding paths of W are thus required to be recursively isomorphic to the elements of a partition of the infinite paths of $\eta(W)$. As a special case, of course, W has no φ-abiding paths iff $\eta(W)$ has no infinite paths; hence the theorem.

The proof proceeds in three steps and is illustrated by the following table.

	Step 1		Step 2		Step 3		
Tree	W	$\rightarrow$	W_1	$\rightarrow$	W_2	$\rightarrow$	$\eta(W)$
Class	T^+		T^+		T_b^+		T
Path property	φ		$\exists^\infty\circledast$		$\exists^\infty\circledast$		$\exists^\infty true$

In Step 1, the main step in the proof, one shows, by induction on the structure of φ, how to construct for each $W \in T^+$ a tree $W_1 \in T^+$ containing only the single mark $\circledast$, with the φ-abiding paths of W corresponding to the recurrences of W_1. The ω-branching W_1 is then turned into a binary tree W_2, preserving recurrences. Finally, the proof of the $\geq_1$ direction of the Recurrence Lemma, is used to obtain the final unmarked tree $\eta(W) \in T$, with recurrences in W_2 corresponding to the infinite paths of $\eta(W)$. All transformations are one–one and recursive, and the path correspondences of the two last steps are actually recursive isomorphisms; it is the first step that yields the one–many aspect of the combined path correspondence.

The third step is subsumed by the second half of the proof of the Recurrence Lemma in Section 5. For the second step, simply replace each node in W_1 of the form of Figure 5 by one of the form of Figure 6, in an inductive fashion with the newly introduced nodes unmarked. The new infinite path, being unmarked from u onward, does not affect recurrences.

Let us now concentrate on the first step. For each $\varphi \in L$, we have to describe a recursive one–one procedure taking a tree $W \in T^+$ to a tree $W_1 \in T^+$ involving the mark $\circledast$ (assumed not to mark W), with the recurrences of W_1 being associated in a many–one fashion with the φ-abiding paths of W. For ease of exposition, and since W_1 depends on φ, we denote the desired W_1 by W_φ.

First, define the mark-free version of W_φ, denoted $W_{\bar{\varphi}}$, as follows. Given $W \in T^+$, denote by W^0 the tree obtained by duplicating each subtree of W in the manner illustrated in Figure 7. Formally, for a node $x = (x_1, \ldots, x_n) \in \mathbb{N}^*$, let

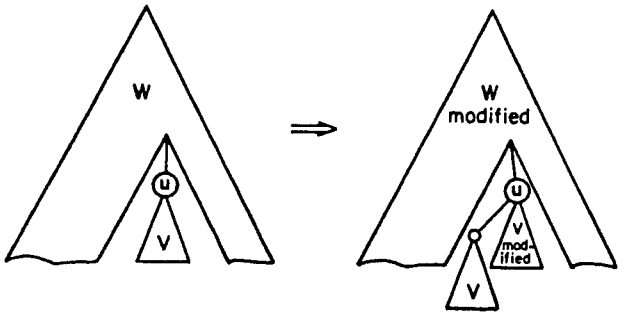
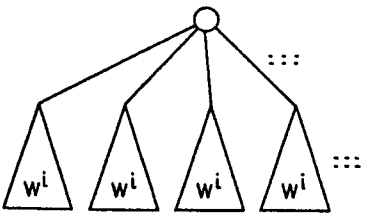


FIGURE 7

FIGURE 8



$x + 1 = (x_1 + 1, \ldots, x_n + 1)$, and let $W + 1 = \{x + 1 \mid x \in W\}$. W^0 is then defined as

$$W^0 = W + 1 \cup \bigcup_{u \in W} \{(u + 1)0y \mid uy \in W, y \in \mathbb{N}^*\}.$$

Referring to Figure 7, the node u is actually replaced by $u + 1$, and its leftmost offspring is $(u + 1)0$. We call these, respectively, the *old* and *new* u's, and use this terminology for their subtrees too. Thus, in W^0, each node has one old occurrence and an old and new subtree. The new subtree remains unchanged (and hence, paradoxically, it is just the "old" one), whereas all nodes of the old subtree get modified in the same way. Moreover, given a node $x \in W^0$, it is easy to determine whether or not x is old (x has no 0's), and if it is not, it is equally easy to find its old root, that is, its nearest old ancestor, since in this case we have $x = (u + 1)0y$. In either case, each $x \in W^0$ corresponds effectively to a unique $\hat{x} \in W$. This correspondence preserves ancestorship. Now, for every $i \geq 0$, W^{i+1} is defined as

$$W^{i+1} = \lambda \cup \bigcup_{\substack{j \in \omega \\ x \in W^i}} j \cdot x$$

and is illustrated in Figure 8. The jth subtree from the left is called the jth *copy* of W^i.

Given $W \in T^+$ and a formula $\varphi \in L$ of type n (i.e., n is the least interger such that $\varphi \in L_n$), take $W_{\bar{\varphi}}$ to be simply W^n.

We now describe the marking of $W_{\bar{\varphi}}$ with $\circledast$, yielding $W_\varphi (= W_1)$, by induction on the structure of φ. The base case of the induction is the four atomic formulas, all of whose types are 0, and the tree to be marked in each case is, therefore, W^0.

For the $\exists^\infty a$ case, simply mark $x \in W^0$ with $\circledast$ iff $\hat{x} \in W$ was marked a. For the $\exists a$ case (respectively, the $\forall a$ case) mark $x \in W^0$ iff some ancestor (respectively, all ancestors) y of $\hat{x}$ in W was (were) marked a. Clearly, recurrences of $\circledast$ in W^0 are associated, as required, with the appropriately abiding paths of W.

For the $\forall^\infty a$ case, no old nodes of W^0 are marked $\circledast$, and a new node is marked iff for every one of its *new* ancestors x, the corresponding $\hat{x}$ is marked with a in W.

Assume that p is a recurrence of ⊛ in W^0. By the construction, $p = q(u+1)r$, where $u+1$ is old and r is an infinite path in u's new subtree; moreover, for r to contain infinitely many ⊛'s it has to be universally marked ⊛. Consequently, in the corresponding path $\hat{p} = \hat{q}u\hat{r}$ in W, $\hat{r}$ is universally marked a, and hence $\hat{p}$ satisfies $\forall^\infty a$. The argument for the converse is similar.

Assume now that $\varphi = \psi_1 \vee \psi_2$, and that φ is of type n. By the definition of type, at least one of ψ_1 and ψ_2 is of type n, say, without loss of generality, ψ_1. Given $W \in T^+$ we can effectively find W_{ψ_1} and W_{ψ_2} and by our assumption $W^-_{\psi_1}$ (the unmarked version of W_{ψ_1}) is W^n, whereas, say, $W^-_{\psi_2}$ is W^m, with $m < n$. First, upgrade W^m to W^n, by carrying out the ω-duplication of Figure 8 $n - m$ times, with each copy of W^m retaining the ⊛ marking of W_{ψ_2}. The resulting trees, $W'_{\psi_1} = W_{\psi_1}$, and W'_{ψ_2}, are now identical in structure. The desired tree W_φ for $\psi_1 \vee \psi_2$ is simply W^n with a node marked ⊛ iff it is marked in either W'_{ψ_1} or W'_{ψ_2}.

For the case $\varphi = \psi_1 \wedge \psi_2$, first upgrade the simpler tree to yield W'_{ψ_1} and W'_{ψ_2} from the inductive hypothesis as before, both being now of type n. Now to obtain W_φ, nodes in W^n are marked inductively as follows: the root λ is marked, and a node $x = (x_1, \ldots, x_t)$ is marked iff there are $m \le i, j < t$, where $(x_1, \ldots, x_m)$ is the closest marked ancestor of x, with $(x_1, \ldots, x_i)$ marked in W'_{ψ_1} and $(x_1, \ldots, x_j)$ marked in W'_{ψ_2}. In short, one marks a node in W_φ by checking that there has been at least one mark in each of W'_{ψ_1} and W'_{ψ_2} since the most recent marking of a node in W_φ along the present path. This procedure is clearly recursive in the markings of W'_{ψ_1} and W'_{ψ_2}, and can easily be seen to yield the correspondence between recurrences required by the conjunction.

The case $\varphi = \bigwedge_i \psi_i$ is treated similarly, with each tree W_{ψ_i} from the inductive hypothesis being first upgraded to be of type W^n. Here, though, a node $x = (x_1, \ldots, x_t)$ in W'_φ is marked just when there are $m \le i_0, i_2, \ldots, i_k < t$, with m as before, and $(x_1, \ldots, x_{i_j})$ is marked in W'_{ψ_j} for each $0 \le j \le k$, and where k is the number of nodes along the path from λ to x already marked. In this way, a recurrence in W_φ can occur just when the path is marked in the W'_{ψ_j} by some sequence consistent with $\{0\}, \{0, 1\}, \{0, 1, 2\}, \ldots$. Hence, the marking in each of the W'_{ψ_j} is represented infinitely often in the path of W_φ, and vice versa.

For the case $\varphi = \bigvee_i \psi_i$, the type of φ is $n+1$, and, consequently, the trees W_{ψ_i} from the inductive hypothesis can be upgraded to be marked versions of W^n. Now W_φ is simply taken to be W^{n+1} marked by having the ith copy of W^n in it inherit the marking from W_{ψ_i}, for each $i \in \omega$. It is easy to see that the recurrences correspond as required. □

As an immediate corollary, we have

COROLLARY 8.2 *For every $\varphi \in L$, the sets of (notations for) φ-avoiding trees in each of T^+, T^+_f, or T^+_b, is in Π^1_1.*

Obviously, many $\varphi \in L$ are equivalent to trivial formulas (like $\exists$⊛) that give rise to classes of trees much simpler than Π^1_1, and in this sense Theorem 8.1 is but an upper bound. It is of interest that even $\forall^\infty$⊛, the dual of $\exists^\infty$⊛, resides much lower down, at least for finite branching.

THEOREM 8.3. *The $\forall^\infty$⊛-avoiding trees in T^+_b (respectively, in T^+_f) form a Π^0_2 set (respectively, Π^0_3).*

PROOF. Consider the statement S: "$\exists$ node x $\forall i \exists y$ (y on the ith level of x's subtree and $\forall z$ on path from x to y, inclusive, z is marked ⊛)." It is easy to see that S is Σ^0_2 or Σ^0_3, depending, respectively, on whether the tree is of bounded or merely finite outdegree (i.e., whether or not the $\exists y$ quantifier is bounded or not).

We show that S is equivalent, for trees in $T\dagger$, to "$\exists$path $\forall^{\infty}\circledast$", and the result follows directly. One direction is obvious. Conversely, consider a tree satisfying S, and let x be the node whose existence is guaranteed by S. We show that there is a path rooted at x and universally marked with $\circledast$. The argument proceeds in a König-like fashion by inductively proceeding down levels of x's subtree along nodes for which infinitely many i's satisfy the "$\exists y \ldots$" part of S. At each level there are finitely many offspring, and so one of them at least must account for infinitely many of the i's; in particular, S guarantees that the node itself is marked $\circledast$. □

Providing more general lower bound information on φ-avoiding trees for various $\varphi \in L$ seems like an interesting topic for future work, especially in view of Section 9.

Theorem 8.1 can apparently be generalized in several ways. The bounded-depth restriction can be removed, and the theorem proved for a language L', which is simply the closure of the atomic formulas under the Boolean and recursive–infinite conjunctions and disjunctions. Also, one can actually close the language under the $\exists$, $\forall$, $\exists^{\infty}$, and $\forall^{\infty}$ quantifiers, so that is it possible to write, say, $\exists^{\infty} \wedge_i \varphi_i$. Both these extensions seem to require a more delicate argument and, for our applications, do not seem to justify the additional work.

Another kind of generalization is important for the applications in Section 9, and so we present it here. Let an *arithmetical tree* be a tree whose membership, leafship, and markedship predicates are arithmetical (i.e., not necessarily recursive but expressable in first-order arithmetic). Denote the resulting classes of trees $a\text{-}T$, $a\text{-}T_{\mathrm{r}}$, $a\text{-}T_{\mathrm{b}}$, $a\text{-}T^{+}$, etc. Also let $a\text{-}L$ be the langauge L in which the infinite conjunctions and disjunctions are also allowed to be arithmetical. Theorem 8.1 holds for this richer language with these richer trees.

THEOREM 8.4. *For every $\varphi \in a\text{-}L$, the set of φ-avoiding trees in $a\text{-}T^{+}$ (and hence also those in $a\text{-}T\dagger$ and $a\text{-}T_{b}^{\dagger}$) is one–one arithmetically reducible to the set of well-founded trees in $a\text{-}T$.*

PROOF. Identical to the proof of Theorem 8.1, but with "arithmetical" replacing "recursive" throughout. □

COROLLARY 8.5. *For every $\varphi \in a\text{-}L$, the set of (notations for) φ-avoiding trees in each of $a\text{-}T^{+}$, $a\text{-}T\dagger$ or $a\text{-}T_{b}^{\dagger}$, is in Π_1^1.*

PROOF. The set of well-founded arithmetical trees is also Π_1^1-complete, (cf. [30]). □

The results of this section yield immediate corollaries concerning Turing machines. Post correspondences, dominoes, etc. Just as, for example, the $\exists^{\infty}\circledast$ formula of L expresses the recurrence property of the domino problems of Section 6, so do other formulas of L express complex properties of the required tiling that enforce recurring or effectively growing patterns, distances, etc. A generic corollary of Theorem 8.1 is the fact that determining whether any of these can occur is within the Σ_1^1/Π_1^1 level of undecidability. For some cases, such as $\exists^{\infty}\circledast$, it is no better, and for others, such as $\forall^{\infty}\circledast$, it is significantly better.

THEOREM 8.6. *For any $\varphi \in L$, the following problems are in Σ_1^1:*

(*i*) *does an NTM admit an infinite φ-abiding computation*?
(*ii*) *does a set T of dominoes admit an infinite φ-abiding tiling*?
(*iii*) *does a Post instance $(\bar{x})$, $(\bar{y}) \in (\{0, 1\}^*)^n$ admit an infinite φ-abiding correspondence*?

THEOREM 8.7. *For $\varphi = \forall^{\infty}\circledast$, the problems of Theorem 8.6 are in Σ^0_2.*

As an example, whether or not T can tile $Z \times Z$ with d occurring only finitely often is in Σ^0_2, that is, no worse than the totality problem for TMs.

We note that Theorem 8.6 (but not 8.7) holds also for NTMs with infinitely many states and/or an infinite alphabet, Post problems over an infinite alphabet and with infinite input sets, and infinite sets of domino types. In these cases the trees are in T^+, not T^+_b.

Another corollary of Theorem 8.1 concerns the topological characterizations of infinite behaviors of the transition systems (TSs) of Arnold [4]. In [4], a result similar to the recurrence lemma was (independently) proved in a different setting, and was used to establish the fact, stated now in the present terminology, that the class of sets of recurrences in T^+_b is a Souslin set (see [4] for definitions). Since the proof of Theorem 8.1 involves correspondences between the φ-abiding paths of W and the infinite paths of $\eta(W)$ one concludes:

PROPOSITION 8.8. *For each $\varphi \in L$ and for each $W \in T^+$, the set of φ-abiding infinite paths in W is a Souslin set.*

9. *Applications to Fairness*

Let us fix some arbitrary conventional programming language PROG with nondeterminism (even unbounded) and/or concurrency, such as those in [9] and [18]. Each program α in PROG can be associated with a *formal computation tree* C_α that consists essentially of all possible sequences of the atomic actions and tests, with common prefixes identified. In a given start state s (i.e., s provides initial values for all variables, etc.), one obtains the induced *computation tree at s*, $C_\alpha(s)$, in which each node u corresponds to an actual state reachable from s by performing the actions and passing the tests along the path from the root to u. False tests or other impassable parts of a program encountered during execution entail truncation of the subtree rooted at the appropriate node.

Given that conventional languages employ recursive (in the recursion-theoretic, not the programming sense) atomic actions and tests (although we allow even arithmetical ones), and given that the finitary nature of the programs results in a finite (albeit possibly unbounded) amount of state information relevant to each point in the computation, one sees that $C_\alpha(s)$, for each α and s, is a tree in a-T. For most languages, it will actually be in T_b, that is, recursive and boundedly branching, but we can afford to be liberal here. Here we are tacitly assuming that the structures over which programs run are standard arithmetic or some effective enrichment thereof (this is in line with all reasonable applications of programming languages). With this established, we can now assume we are given an effective enumeration $s_0, s_1, \ldots$ of all possible start states, and can consider the *universal computation tree* $\hat{C}_\alpha$, illustrated in Figure 9, consisting of $C_\alpha(s_0)$, $C_\alpha(s_1)$, $\ldots$ connected to a common root.

A state property p of interest can be modeled as a mark marking nodes of $\hat{C}_\alpha$, and if it is first-order definable, the marked version of $\hat{C}_\alpha$ will be in a-T^+. What we are saying, in more general terms, is that given any $\varphi \in L$ or $\varphi \in a$-L, where the marks involved model properties of states of the computation of a program $\alpha \in$ PROG, the appropriately marked tree $\hat{C}^+_\alpha$ is in a-T^+ and therefore is a candidate for application of (the proofs of) Theorems 8.1 and 8.4. Ladner [22] has also considered various notions of fairness as "path restrictions" on the appropriate computation trees, leading naturally to the present treatment.

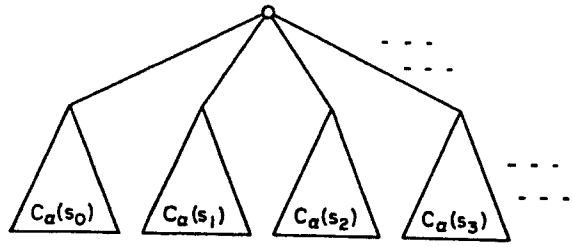


FIGURE 9

Applying these proofs to $\hat{C}_\alpha^+$ results in a tree $\eta(\hat{C}_\alpha^+)$ in a-T, whose infinite paths correspond to the φ-abiding paths of $\hat{C}_\alpha^+$.

Definition. Given $\alpha \in$ PROG and $\varphi \in L$, we say that α *φ-fairly terminates* if for all start states s, α admits no φ-abiding infinite computations starting in s.

The discussion above and the Corollaries 8.2 and 8.5 hence yield

THEOREM 9.1. *For each $\alpha \in PL$, $\varphi \in a$-L, the problem of whether α φ-fairly terminates is in Π_1^1.*

We now observe that L and a-L can express every previously proposed notion of fairness and many more. In fact, it is hard to imagine any notion of fairness, or unfairness, or any other property of infinite computations that might be of interest for programs in such languages, that is not expressible in L or a-L. For example, weak, strong, and extreme fairness for the program (1) of Section 3 can be written as follows (with liberal formulation of the arithmetical or recursive meanings of marks):

Weak fairness:

$$(\forall^\infty(A \textit{ true}) \supset \exists^\infty(\alpha \textit{ executed})) \wedge \forall^\infty(B \textit{ true}) \supset \exists^\infty(\beta \textit{ executed})),$$

and, more generally,

$$\bigwedge_{1 \le i \le n} (\forall^\infty i\textit{-enabled} \supset \exists^\infty i\textit{-taken}).$$

Strong fairness:

$$(\exists^\infty(A \textit{ true}) \supset \exists^\infty(\alpha \textit{ executed})) \wedge \exists^\infty(B \textit{ true}) \supset \exists^\infty(\beta \textit{ executed})),$$

and, more generally,

$$\bigwedge_{1 \le i \le n} (\exists^\infty i\textit{-enabled} \supset \exists^\infty i\textit{-taken}).$$

Extreme fairness (Here $\{q_j\}$ is an effective enumeration of all formulas in first-order arithmetic, for example):

$$\bigwedge_j (\exists^\infty(q_j \textit{ true}) \supset (\exists^\infty(\alpha \text{ executed with } q_j \textit{ true}) \wedge \exists^\infty(\beta \text{ executed with } q_j \textit{ true}))),$$

and, more generally,

$$\bigwedge_j (\exists^\infty(q_j \textit{ true}) \supset \bigwedge_{1 \le i \le n} (\exists^\infty(i\text{-taken with } q_j \textit{ true})).$$

How can Theorems 8.1 and 8.4 help in actual proofs of fair termination? We venture the following:

STATEMENT. *For each $\varphi \in a$-L, Theorems* 8.1 *and* 8.4 *and their proofs provide a semantically complete proof method for φ-fair termination.*

JUSTIFICATION OF STATEMENT. The claim in the statement can be justified in several ways. In a pure mathematical sense, given an arbitrary fixed $\varphi \in a\text{-}L$ and a program $\alpha \in PL$, the tree $\hat{C}_\alpha^+$ is an arithmetical (or recursive) marked tree, and hence its translate $\eta(\hat{C}_\alpha^+)$ with respect to φ can be represented by some finite machine (perhaps with arithmetical oracles). This machine can be thought of as a program with und (i.e., unbounded nondeterminism; actually the correct term should probably be *ω-nondeterminism*). The proof method of [2] for example, can then be used to prove the new programs's termination, i.e., the translate-tree's well-foundedness. Since the method of [2] is complete relative to an appropriate program-free language, the method outlined (translation via η; then proof of termination) is semantically complete, and in fact also complete relative to the same underlying language.

In a more pragmatic sense, one can consider the formal computation tree C_α, expand it by duplicating nodes for each mark, one copy being marked and the other not, and then carry out the η translation *before* considering the various start states s_i. Again, this tree can be written as a program with und, but now the result is a *uniform* explicit scheduler S_α, which can then be applied to the various s_i. It seems reasonble to suppose that S_α can be written, in general, in terms of the basic actions and tests of α and the marks of φ, with some insignificant extra recursive machinery. Fully exploiting this possibility, however, would seem to require additional work beyond our general Theorems 8.1 and 8.4. □

We have worked through the proof of Theorem 8.1 in the cases of weak and strong fairness for program (1) of Section 4, and have indeed been able to exhibit *explicit* explicit schedulers S_α, written in terms of the original program, which are the (indirect) result of the η translation. As mentioned above, however, this procedure justifies further work and can, we believe, be generalized in the spirit of Theorems 8.1 and 8.4 to all $\varphi \in a\text{-}L$.[1]

The new explicit schedulers for program (1) are the following, and the reader should have no difficulty proving that they terminate iff program (1) fairly terminates with the appropriate notion of fairness:

Weak fairness:

(If $B \to \beta$ **IF**)*; **while** $A \vee B$ **do** (**IF** $A \to \alpha \,\square\, \neg A \to$ *skip* **FI**)$^+$
(**IF** $B \to \beta \,\square\, \neg B \to$ *skip* **FI**)$^+$ **od**,

Strong fairness:

(**IF** $A \to \alpha \,\square\, B \to \beta$ **FI**)*;
while $A \vee B$ **do** ((**if** A **then** α)$^+$ (**if** B **then** β)$^+$
or (**if** $\neg A$ **then** β)
or (**if** $\neg B$ **then** α)) **od**.

Here $\gamma^* = \bigcup_{i \geq 0} \gamma^i$ is short for $i \leftarrow ?;\ \gamma^i$, and $\gamma^+ = \bigcup_{i > 0} \gamma^i$ is short for $i \leftarrow ?;\ i \leftarrow i + 1;\ \gamma^i$.

10. *Conclusion*

We have presented elementary recursive translations between classes of recursive (or arithmetical) trees, yielding applications to high undecidability and fairness. We feel that characterizing Π_1^1 in terms of "thin" φ-avoiding trees for some appropriate φ is more beneficial for computer science than using well-founded ω-trees. This is because computer science deals with finite objects (programs,

[1] *Note added in proof*: Such a generalization has recently been obtained by I. Dayan and the author and appears in "Fair termination with cruel schedulers," *Fundamenta Informatica IX*, 1, 1986, to appear.

machines, graphs, combinatorical objects such as finite sets of dominoes, etc.) which usually give rise to finite branching. A detailed account of one aspect of this apparent advantage is given in [16].

As mentioned at the end of Section 9, there is still much work to be done, in the spirit of [1], [11], [12], [14], [15], [23], and [27] in finding clean and useful special-purpose proof methods for fair termination of various kinds, since, even if the uniform explicit schedulers S_α described in Section 9 are worked out generally, they might more often than not turn out to be quite unwieldy.

As another direction for further work, we suggest generalizing the results to languages for describing properties of certain infinite *subtrees*, not merely paths. This would parallel the investigation of branching-time versus linear-time formalisms for reasoning about programs.

ACKNOWLEDGMENTS. We thank A. Pnueli and G. Plotkin for their useful comments in the critical stages of the research, and P. van Emde Boas for his most insightful remarks and corrections on three versions of the paper. In a conversation in 1981, J. Makowsky and J. Stavi predicted that a general treatment of fairness could be given, using recursive trees. In a way, parts of this paper can be seen to follow their suggestion.

REFERENCES

1. APT, K.R., AND OLDEROG, E.R. Proof rules and transformations dealing with fairness. *Sci. Comput. Prog. 3* (1983), 65–100.
2. APT, K.R., AND PLOTKIN, G.D. Countable nondeterminism and random assignment. Unpublished manuscript, 1982.
3. APT, K.R., PNUELI, A., AND STAVI, J. Fair termination revisited with delay. In *Proceedings of the 2nd Conference on Foundations of Software Technology and Theoretical Computer Science (FST-TCS)* (Bangalore, India, Dec.) 1982, pp. 146–170.
4. ARNOLD, A. Topological characterizations of infinite behaviours of transition systems. In *Proceedings of the 10th International Colloquium on Automata, Languages, and Programming.* Lecture Notes in Computer Science, vol. 154. Springer-Verlag, New York, 1983, pp. 28–38.
5. BERGER, R. The undecidability of the domino problem. *Mem. Amer. Math. Soc. 66* (1966).
6. BOOM, H.J. A weaker precondition for loops. *ACM Trans. Prog. Lang. Syst. 4*, 4 (Oct. 1982), 668–677.
7. BUCHI, J.R. Turing machines and the entscheidungsproblem, *Math. Ann. 148* (1962), 201–213.
8. CHANDRA, A.K. Computable nondeterministic functions. In *Proceedings of the 19th IEEE Symposium on the Foundations of Computer Science.* IEEE, New York, 1978, pp. 127–131.
9. DIJKSTSRA, E.W. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N.J., 1976.
10. EMERSON, E.A., AND CLARKE, E.M. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th International Colloquium on Automata, Languages, and Programming.* Lectures Notes in Computer Science, vol. 85, Springer-Verlag, New York, 1980, pp. 169–181.
11. FRANCEZ, N. *Fairness*, Manuscript, 1983.
12. FRANCEZ, N., AND KOZEN, D. Generalized fair termination. In *Proceedings of the 11th ACM Symposium on the Principles of Programming Languages* (Salt Lake City, Utah, Jan. 15–18). ACM New York, 1984, pp. 46–53.
13. FÜRER, M. Alternation and the Ackermann case of the decision problem. *L'Enseignement mathematique T.XXVII*, fasc. 1–2 (1981), 137–162.
14. GRUMBERG, O., AND FRANCEZ, N. A complete proof rule for (weak) equifair termination. Res. Rep. RC-9634, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., 1982.
15. GRÜMBERG, O., FRANCEZ, N., MAKOWSKY, J.A., AND DE ROEVER, W.P. A proof rule for fair termination of guarded commands. In *Proceedings of the International Symposium on Algorithmic Languages.* North-Holland, Amsterdam, 1981, pp. 339–416.

15a. GUREVICH, Y. The decision problem for standard classes. *J. Symb. Logic 41* (1976), 460–464.

16. HAREL, D. Recurring dominos: Making the highly undecidable highly understandable. *Ann. Disc. Math. 24* (1985), 51–72.

17. HAREL, D., PNUELI, A., AND STAVI, J. Propositional dynamic logic of nonregular programs. *J. Comput. Syst. Sci. 26* (1983), 222–243.
18. HOARE, C.A.R. Communicating sequential processes. *Commun. ACM 21*, 8 (Aug. 1978), 666–677.
19. HOPCROFT, J.E., AND ULLMAN, J.D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 1979.
20. KAHR, A.S., MOORE, E.F., AND WANG, H. Entscheidungsproblem reduced to the $\forall\exists\forall$ case. In *Proceedings of the National Academy of Sciences, USA*, vol. 48. 1962, pp. 365–377.
21. KNUTH, D. *The Art of Computer Programming: Fundamental Algorithms.* Addison-Wesley, Reading, Mass., 1968.
22. LADNER, R.E. The complexity of problems in systems of communicating sequential processes. *J. Comput. Syst. Sci. 21* (1980), 179–194.
23. LEHMANN, D., PNUELI, A., AND STAVI, J. Impartiality, justice and fairness: The ethics of concurrent termination. In *Proceedings of the 8th International Colloquium on Automata, Languages, and Programming.* Lecture Notes in Computer Science, vol. 115. Springer-Verlag, New York, 1981, pp. 264–277.
24. LEWIS, H.R. Complexity of solvable cases of the decision problem for the predicate calculus. In *Proceedings of the 19th IEEE Symposium on the Foundations of Computer Science.* IEEE, New York, 1978, pp. 35–47.
25. LEWIS, H.R. *Unsolvable Classes of Quantificational Formulas.* Addison-Wesley, Reading, Mass., 1979.
26. LEWIS, H.R., AND PAPADIMITRIOU, C.H. *Elements of the Theory of Computation.* Prentice-Hall, Englewood Cliffs, N.J., 1981.
27. OLDEROG, E.R., AND APT, K.R. Transformations realizing fairness assumptions for parallel programs. In *Proceedings of STACS '84*, M. Fontet and K. Mehlhorn, Eds. Lecture Notes in Computer Science, vol. 166. Springer-Verlag, New York, 1984, pp. 20–42.
28. PARK, D. A predicate transformer for weak fair iteration. In *Proceedings of the 6th IBM Symposium on the Mathematical Foundations of Computer Science* (Hakone, Japan). IBM, Japan, 1981, pp. 259–275.
29. PNUELI, A. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing* (Boston, Mass., Apr. 25–27). ACM, New York, 1983, pp. 278–290.
30. QUEILLE, J.P., AND SIFAKIS, J. Fairness and related properties in transition systems—A temporal logic to deal with fairness. *Acta Informatica 19* (1983), 195–220.
31. ROBINSON, R.M. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Math. 12* (1971), 177–209.
32. ROGERS, H. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.
33. SAVELSBERG, M., AND VAN EMDE BOAS, P. BOUNDED TILING, an Alternative to SATISFIABILITY. In *Proceedings of the 2nd Frege Conference*, G. Wechsung, Ed. Akademie Verlag, Berlin DDR. *20* (1984), 354–363.
34. STREETT, R.S. Global process logic is $\Pi\frac{1}{1}$-complete. Manuscript, 1982.
35. VAN EMDE BOAS, P. Dominos are forever. In *Proceedings of the 1st GTI Workshop* (Paderborn). pp. 76–95, 1983.
36. WANG, H. Proving theorems by pattern recognition II. *Bell Systs. Tech. J. 40* (1961), 1–41.
37. WANG, H. Dominos and the $\forall\exists\forall$ case of the decision problem. In *Mathematical Theory of Automata.* Polytechnic Institute, Brooklyn, 1963, pp. 23–55.

RECEIVED OCTOBER 1983; REVISED JANUARY 1985; ACCEPTED JUNE 1985