A General Result on Infinite Trees and Its Applications

(preliminary report)

David Harel

Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100, Israel

Abstract

A generic translation between various kinds of recursive trees is 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 an be any member of a very wide class of properties of infinite paths. Two of the applications involve the formulation of large classes of Π_1^1 variants of classical computational problems, and the existence of a general method for proving termination of nondeterministic or concurrent programs under any reasonable notion of fairness.

1. Introduction

In this paper a general theorem is proved, establishing elementary recursive one-one reductions between various kinds of infinite trees. The result itself might seem less appealing than some of its corollaries, which indeed served as the prime motivation for obtaining the result in the first place. Consequently, the paper is structured in a way that presents much of the background and technical preliminaries for the applications before touching upon the main result.

In Section 2 we describe the two levels of undecidability relevant to the paper: the low Σ_1^0/Π_1^0 (r.e./co-r.e.) level and the high Σ_1^1/Π_1^1 (co-inductive/inductive) level, and indicate their classical recursive-well-founded-trees characterization. We then present a recent recursive-recurrence-free-trees alternative to this characterization, taken from [H2] (which is a precursor of the present paper), and which leads, among other things, to the description of some simple highly undecidable problems about NTM's, dominoes, etc. In Section 3, 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.

The main result is presented in Section 4. In Sections 5 and 6 it is applied to the material of Sections 2 and 3, respectively, resulting in large classes of new Π_1^1 variants of classical computational problems and, perhaps more importantly, a generic proof method for the termination of programs under almost any conceivable notion of fairness.

The present paper reports on a direct continuation and culmination of the research described in [H2]. We expect to publish a final journal version combining the results of both.

2. Two Levels of Undecidability

While there exist many different levels of undecidability, there seem to be mainly two which stand out as being fundamental and naturally-occuring: the Σ_1^0/Π_1^0 (that is, the r.e./co-r.e.) level, and the Σ_1^1/Π_1^1 (sometimes called the co-inductive/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 shall not attempt to convince the reader of the special role these two levels play by a review of undecidability results in general, but we do point out that the theoretical computer science community has repeatedly seen examples of undecidable problems which turn out actually to be in one of these levels. A striking example is in the field of logics of programs, where numerous entirely different logical systems have been shown to have Π_1^1-complete validity problems, cf. [H1]. Fairness and unbounded nondeterminism is another. To see how unbounded nondeterminism is connected with Π_1^1 one is led to consider trees.

One of classical ways of viewing the low Σ_1^0/Π_1^0 level of undecidability is by finitely-branching recursive trees; say, the computation trees of NTM's. Specifically, this level captures the well-foundedness predicate (= "are all paths finite?") of these trees. Similarly, the classical treatment of the Σ_1^1/Π_1^1 level is via the well-foundedness of countably-branching recursive trees; the set of (notations for) well-founded such trees is Π_1^1-complete, and hence also is the set of (notations for) constructive ordinals. See Rogers [R].

In [H2] it is shown that the Σ_1^1/Π_1^1 level can be viewed alternatively by considering *finitely*-branching recursive trees. It is shown therein, using elementary transformations between trees, that the set of (notations for) recursive ***recurrence-free marked*** binary trees is Π_1^1-complete. Here a marked tree is one in which some recursive subset of the nodes are marked, and a recurrence is an infinite path containing infinitely many marked nodes. (A similar result was proved independently in [A].) This "thin-trees" characterization was shown in [H2] to lead to easily describable, highly undecidable computational problems, that are in fact variants of well-known ones on the Σ_1^0/Π_1^0 level. Three of the more appealing of these are the following:

Proposition 1 [EC, Fü,HPS,S,H2]: The problem of whether a NTM admits an infinite computation on blank tape, that reenters its start state infinitely often, is Σ_1^1-complete.

A ***domino*** is a 1×1 square fixed in orientation, with a color associated with each of its sides. A ***tiling*** requires adjacent edges to be monochromatic. See [W].

Proposition 2 [H2]: The following ***recurring dominoes*** problem is Σ_1^1-complete: Given a set $T = \{d_0, d_1, \ldots, d_m\}$ of dominoes, can T tile $Z \times Z$ such that d_0 appears infinitely often in the tiling?

Proposition 3 [H2]: The following variant of Post's correspondence problem is Σ_1^1-complete: Given $(x_1, \ldots, x_n), (y_1, \ldots, y_n), x_i, y_i \in \{0,1\}^*$, is there a sequence $\sigma = (i_1, i_2, \ldots)$, $1 \leq i_j \leq n$, with infinitely many occurrences of 1 in σ, such that $x_{i_1}x_{i_2}\cdots = y_{i_1}y_{i_2}\cdots$?

In all cases the proofs first establish the fact that with the objects at hand (NTM's, dominoes or Post correspondences) one can define any recursive finitely-branching tree, with the marking corresponding to the machine being in the start state, or the domino d_0 or the index 1 being most recently used. The thin/fat correspondence described above between the two kinds of trees characterizing the Σ_1^1/Π_1^1 level is then envoked.

Note that, just as with Turing machines, the "recurring" highly undecidable problems described here have finite and infinite analogues that are NP-complete and Π_1^0-complete, respectively. For example, whether or not, given a finite set of dominoes T and a number n, T can tile some $n \times n$ square is NP-complete [L], whereas whether T can tile $Z \times Z$ is Π_1^0-complete [W,Be]. In this sense Propositions 1–3 are natural extensions.

In [H1] all Π_1^1-hardness results in logics of programs (and some NP-, PSPACE-, and Π_2^0- ones too) are provided with short transparent proofs using variants of the recurring-domino problem of Prop. 2.

3. Fairness

Much effort has gone recently into the investigation of the behavior of nondeterministic or concurrent programs under the assumption of ***fairness***, e.g., [AO,F,FK,GF,GFMR,LPS,Pa,P,QS]. 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 shall be referring in this paper to the following simple bidirectional nondeterministic program, cf. [D]:

$$\text{DO } A \rightarrow \alpha \,\Box B \rightarrow \beta \text{ OD} \tag{1}$$

(= "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 is ***taken*** if its ***action*** (α or β) is executed.

The main direction of research in this area (as is evident, for example, from Francez' encyclopaedic survey [F]) 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 relativise fairness to given sets of states.

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

<u>weak fairness:</u>

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

<u>strong fairness:</u>

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

<u>extreme fairness:</u>

> An infinite computation is extremely fair if for every first-order state formula p, if p is true infinitely often then 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. [LPS,AO,F]) but for the third, introduced in [P] (as well as for a host of others (cf. [QS,F])), 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 to 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 [AO], and later also in [APS,F], it is shown in this approach how to transform programs, such as program (1) above, which utilize bounded nondeterminism (in the sequel, ***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 [AP] based upon ideas of [Bo,C]), 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 [AO,F] for proving, respectively, weak and strong fair-termination:

<u>weak-fairness</u>

$$
\begin{array}{l}
a \leftarrow ?;\ b \leftarrow ?; \\
\underline{\mathrm{DO}}\ (A \wedge a \leq b) \rightarrow \\
\qquad (\alpha; a \leftarrow ?\ (\underline{\mathrm{if}}\ B\ \underline{\mathrm{then}}\ b \leftarrow b-1\ \underline{\mathrm{else}}\ b \leftarrow ?)) \\
\square(B \wedge b < a) \rightarrow \\
\qquad (\beta; b \leftarrow ?\ (\underline{\mathrm{if}}\ A\ \underline{\mathrm{then}}\ a \leftarrow a-1\ \underline{\mathrm{else}}\ a \leftarrow ?))\ \underline{\mathrm{OD}},
\end{array} \tag{2}
$$

<u>strong fairness</u>

$$
\begin{array}{l}
a \leftarrow ?;\ b \leftarrow ? \\
\underline{\mathrm{DO}}\ (A \wedge a \leq b) \rightarrow \\
\qquad (\alpha; a \leftarrow ?\ (\underline{\mathrm{if}}\ B\ \underline{\mathrm{then}}\ b \leftarrow b-1)) \\
\square(B \wedge b < a) \rightarrow \\
\qquad (\beta; b \leftarrow ?\ (\underline{\mathrm{if}}\ A\ \underline{\mathrm{then}}\ a \leftarrow a-1))\ \underline{\mathrm{OD}}.
\end{array} \tag{3}
$$

The reader should be able to convince his/herself that program (2) (resp. (3)) everywhere-terminates iff program (1) fairly terminates under weak (resp. strong) fairness. As mentioned, the proof system of [AP] can be used to prove 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 [LPS]) 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 w.r.t. 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 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. In a sense the "thin/fat trees" correspondence lemma of [H2] captures such a connection, but it is one of only a very simple nature.

4. The Main Result

The general setting is that of recursive trees. A *node* is a finite sequence of natural numbers (i.e., an element of N^*) denoting the path leading to it from the *root* λ, and a *tree* is simply a subset of N^* closed under the prefix operation. We take a *recursive tree* to be one for which both membership and leafship ("is x a leaf?") are recursive. This is the *strongly recursive* of Rogers [R]. A tree is *well-founded* if all its paths are finite; it is *finitely-branching* if each node has only finitely many offspring, and is *k-branching* if it is actually a subset of $\{0,\ldots,k-1\}^*$ (so that in particular each node has at most k offspring).

Let $\sum$ be some fixed (possibly infinite) alphabet. A *marked tree* is one in which nodes are labelled with (possibly infinitely many) letters from $\sum$; i.e., W comes complete with a marking predicate $M_W \subseteq W \times \sum$. 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_k^+ stand, respectively, for the sets of recursive trees, recursive marked trees, recursive finitely-branching marked trees, and recursive k-branching marked trees.

Throughout, we understand a recursive tree to be represented by some Turing machine defining it. For example, a tree W in T is represented by some machine computing

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

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 $a \in \sum$ 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 "∃fullpath" fragment of Emerson and Clarke's [EC] 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_k^+ if pieces are not allowed to move too far) with, say ① and 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 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 recursive mark.

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

Another important special formula in L is $\exists^\infty ⊛$, where ⊛ is any fixed mark in $\sum$. A $\exists^\infty ⊛$-abiding path is what was termed *recurrence* in Section 2, and we can now restate the recurrence lemma of [H2] (cf. also [A,EC]):

<u>Recurrence Lemma</u>: For each $k > 1$, the set of well-founded trees in T is recursively isomorphic to the set of $\exists^\infty ⊛$-avoiding trees in T_k^+.

The one-one recursive transformations used in [H2] to prove this lemma (relying on a theorem of Myhill [R. p. 85] to obtain an isomorphism from them) are particularly simple. The main technical result of this paper is the following, in which the $\sum_1$ direction of the recurrence lemma is significantly strengthened by generalizing the property of infinite paths which is used. In passing, we also remove the k subscript.

Theorem 4: Let φ be an arbitrary formula of L. The set of φ-avoiding trees in T^+ (and hence also those in T_f^+ and T_k^+ for each k) is one-one reducible to the set of well-founded trees in T.

Proof of Theorem 4: We actually establish the following stronger claim:

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

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 is in three steps and is illustrated by the following table.

	step 1		step 2		step 3	
tree	W	$\to$ W_1		$\to$ W_2		$\to$ $\eta(W)$
class	T^+	T^+		T_2^+		T
path property	φ	$\exists^\infty \circledast$		$\exists^\infty \circledast$		$\exists^\infty$ ***true***

In step 1, the main one of 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, appearing in [H2], 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 path correspondence.

The third step appears in detail in [H2] and is hence omitted. For the second, simply replace each node in W_1 of the form of Fig. 1 by one of the form of Fig. 2, with the newly introduced nodes unmarked. The new infinite path, being unmarked from u onwards, 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

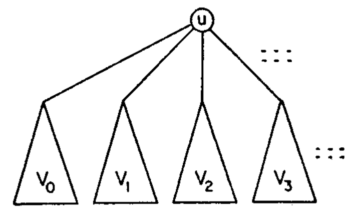

Figure 1

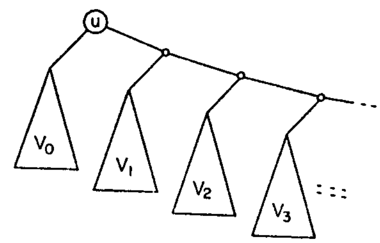

Figure 2

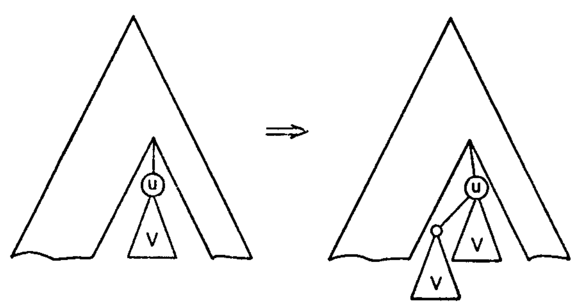

Figure 3

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_φ^-, as follows. Given $W \in T^+$, denote by W^0 the tree obtained by duplicating each subtree of W in the manner illustrated in Fig. 3. Formally, for a node $x = (x_1, \ldots, x_n) \in \mathbb{N}^*$, let $x+1 = (x_1+1, \ldots, x_n+1)$, 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 Fig. 3, 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. 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, i.e., 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 Fig. 4. The j'th subtree from the left is called the j'th *copy* of W^i.

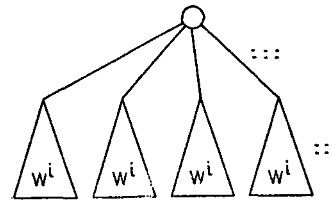

Figure 4

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

We now describe the marking of W_φ^- with ⊛, yielding $W_1 = W_\varphi$, by induction on the structure of φ. The base case of the induction are 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 ⊛ 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 ⊛ 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 ⊛, 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, w.l.o.g., 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 Fig. 4 $n-m$ 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 $m \leq i, j < t$, 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 $\varphi = \wedge_i \psi_i$ 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 $m \leq i_0, i_2, \ldots, i_k < t$, with m as before, and $(x_1, \ldots, x_{i_j})$ is marked in W'_{ψ_j} for each $0 \leq j \leq k$, 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 i'th 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 5: For every $\varphi \in L$, the sets of (notations for) φ-avoiding trees in each of T^+, T_f^+ or T_k^+, 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 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 6: The $\forall^\infty$⊛-avoiding trees in any of the T_k^+ (respectively, in T_f^+) form a Π_2^0 set (resp., Π_3^0).

Proof: Consider the statement S: "$\exists$ node $x \forall i \exists y$ (y on the i'th 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 Π_2^0 or Π_3^0, 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_f^+, to "$\exists$path $\forall^\infty$⊛". 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 ⊛. 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, and, in particular, S guarantees that it itself is marked⊛. ∎

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 5.

Theorem 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 it is possible to write, say, $\exists^\infty \bigwedge_i \varphi_i$. Both these extensions seem to require a considerably 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 6, 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 T_a, T_{a_f}, T_{a_k}, T_a^+, etc. Also let L_a be the language L in which the infinite conjunctions and disjunctions are also allowed to be arithmetical. Theorem 4 holds for this richer language with these richer trees:

Theorem 7: For every $\varphi \in L_a$, the set of φ-avoiding trees in T_a^+ (and hence also those in $T_{a_f}^+$ and $T_{a_k}^+$ for each k) is one-one arithmetically reducible to the set of well-founded trees in T_a.

Proof: Identical to the proof of Theorem 4, but with "arithmetical" replacing "recursive" throughout. ∎

Corollary 8: For every $\varphi \in L_a$, the set of (notations for) φ-avoiding trees in each of T_a^+, $T_{a_f}^+$, or $T_{a_k}^+$, is in Π_1^1.

Proof: The set of well-founded arithmetical trees is also Π_1^1-complete, [cf. [R]). ∎

5. Applications to High Undecidability

That the computation tree of a NTM is recursive and finitely branching is obvious. If one thinks of properties of points in the computation (such as the current state and tape symbol or similar information concerning the computation leading to the point) as recursive marks, one obtains a tree in T_k^+, for an appropriate k. Similarly, Post correspondence problems and many variants of the domino problem give rise to trees in T_k^+, as do many other computational and combinatorical formalisms. In each of these, the language L allows one to specify many complex properties of the infinite computation, or infinite tiling, etc. For example, the simple $\exists^\infty$⊛ recurrence property can specify, say that a particular domino in the input set T occurs infinitely often in the required tiling. Clearly, in L one can state complex properties of the required tiling, enforcing recurring or effectively growing patterns, distances, etc. A generic corollary of Theorem 4 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 $\exists^\infty$⊛, it is no better, and for others, such as $\forall^\infty$⊛, it is significantly better.

Theorem 7: For any $\varphi \in L$, the following problems are in Σ_1^1:

(i) does a 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: For $\varphi = \forall^\infty$⊛ the problems of Theorem 7 are in Π_2^0.

As an example, whether or not T can tile $Z \times Z$ with d_0 occurring only finitely often is in Π_2^0, i.e., equivalent to the totality problem for TM's.

We note that Theorem 7 holds also for NTM's with infinitely many states and/or an infinite alphabet, Post problems over an infinite alphabet and with infinite input sets, and infinite dominoes. In these cases the trees are in T^+.

Another corollary of Theorem 4 concerns the topological characterizations of infinite behaviors of the transition systems (TS's) of Arnold [A]. In [A] 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_k^+ is a Souslin set (see [A] for definitions). Since the proof of Theorem 4 involves correspondences between the φ-abiding paths of W and the infinite paths of $\eta(W)$ one concludes:

Proposition 9: For each $\varphi \in L$ and for each $W \in T^+$, the set of φ-abiding infinite paths in W is a Souslin set.

6. Applications to Fairness

Let us fix some arbitrary conventional programming language PL with nondeterminism (even unbounded) and/or concurrency, such as those in [D,H]. Each program α in PL can be associated with a ***formal computation tree*** C_α which 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 effective 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 T_a. For most languages it will actually be in T_k, i.e., recursive and finitely-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 Fig. 5,

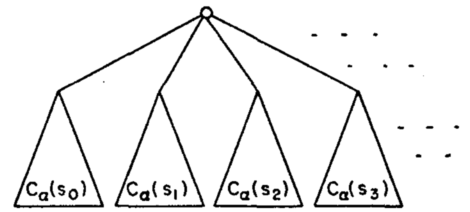

Figure 5

consisting of $C_\alpha(s_0), C_\alpha(s_1), \ldots$ connected to a common root.

A state property p of interest can be modelled 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 T_a^+. What we are saying in more general terms is that given any $\varphi \in L$ or $\varphi \in L_a$, where the marks involved model properties of states of the computation of a program $\alpha \in PL$, the appropriately marked tree $\hat{C}_\alpha^+$ is in T_a^+ and therefore is a candidate for application of (the proof of) Theorems 4 and 7.

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

Definition: Given $\alpha \in PL$ 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 Corollaries 5 and 8 hence yield

Theorem 10: For each $\alpha \in PL$, $\varphi \in L_a$, the problem of whether α φ-fairly terminates is in Π_1^1.

We now observe that L can express every hitherto 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 but that is not expressible in 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:

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

and more generally

$$\bigwedge_{1 \le i \le n} (\forall^\infty i\text{-}\mathit{enabled} \supset \exists^\infty i - \mathit{taken});$$

strong fairness:

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

and more generally

$$\bigwedge_{1\leq i\leq n} (\exists^{\infty} i\text{-}enabled \supset \exists^{\infty} i - taken);$$

extreme fairness:

(here $\{\varphi_j\}$ is an effective enumeration of all first-order formulas)

$$\bigwedge_j (\exists^{\infty}(\varphi_j\ true) \supset (\exists^{\infty}(\alpha \text{ executed with } \varphi_j\ true) \wedge \exists^{\infty}(\beta \text{ executed with } \varphi_j\ true)));$$

and more generally

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

How can Theorems 4 and 7 help in actual proofs of fair termination? We venture the following:

Claim 11: For each $\varphi \in L_\alpha$, Theorems 4 and 7 and their proofs provide a semantically complete proof method for φ-fair termination.

Justification of Claim: The claim can be justified in several ways. In a pure mathematical sense, given an arbitrary fixed $\varphi \in L_\alpha$ and a program $\alpha \in PL$, the tree $\hat{C}_\alpha^+$ is an arithmetical (or recursive) marked tree, and hence its translate $\eta(\hat{C}_\alpha^+)$ w.r.t. φ can be represented by some finite machine (perhaps with arithmetical oracles). This machine can be thought of as a program with ***und***, and the proof method of [AP], for example, can be used to prove the programs's termination, i.e., the translate tree's well-foundedness. Since the method of [AP] 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 reasonable 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 4 and 7. ∎

We have worked through the proof of Thm. 4 in the cases of weak and strong fairness for program (1) of Section 3, and have indeed been able to exhibit ***explicit*** explicit schedulers S_α, written in terms of the original program, which are the result of the η translation. As mentioned above, however, this procedure justifies further work and can, we believe, be generalized in the spirit of Thm's 4 and 7 to all $\varphi \in L_\alpha$.

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

weak fairness:

$$\underline{\text{while}}\ A \vee B\ \underline{\text{do}}\ (\underline{\text{IF}}\ A \to \alpha \ \Box \neg A \to skip\ \underline{\text{FI}})^+ \ (\underline{\text{IF}} B \to \beta \ \Box \neg B \to skip\ \underline{\text{FI}})^+\ \underline{\text{od}},$$

strong fairness:

$$(\underline{\text{IF}}\ A \to \alpha \ \Box B \to \beta\ \underline{\text{FI}})^*;$$
$$\underline{\text{while}}\ A \vee B\ \underline{\text{do}}\ ((\underline{\text{if}}\ A\ \underline{\text{then}}\ \alpha)^+\ (\underline{\text{if}}\ B\ \underline{\text{then}}\ \beta)^+ \ \underline{\text{or}}\ (\text{if } \neg A\ \underline{\text{then}}\ \beta) \ \underline{\text{or}}\ (\text{if } \neg B\ \underline{\text{then}}\ \alpha))\ \underline{\text{od}}.$$

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

7. Conclusion

We have presented a general result providing 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 with well-founded ω-trees. This is because computer science deals with finite objects (programs, 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 [H1].

As mentioned at the end of Section 6, there is still much work to be done, in the spirit of [AO, F, FK, GF, LPS], 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 above are worked out generally, they might more often than not turn out to be quite unwieldly.

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 vs. linear-time formalisms for reasoning about programs.

Acknowledgements

We thank A. Pnueli and G. Plotkin for their most useful comments in the critical stages of the research.

REFERENCES

[AO] Apt, K.R. and E.R. Olderog, Proof Rules and Transformations Dealing with Fairness, TR 82–47, LITP, University of Paris, 7, October 1982. To appear.

[AP] Apt, K.R. and G.D. Plotkin, Countable Nondeterminism and Random Assignment, Manuscript, 1982. To appear in *JACM.*

[APS] Apt, K.R., A. Pnueli and J. Stavi, Fair termination revisited with delay, TR 82–51, LITP, University of Paris, 7, October 1982. Also in *Proc. 2nd Conference on Foundations of Software Technology and Theoretical Computer Science* (FST-TCS), Bangalore, India, December 1982.

[A] Arnold, A., Topological Characterizations of Infinite Behaviours of Transition Systems, *Proc. 10th Int'l. Colloq. Automata, Lang., Prog.,* Springer-Verlag LNCS **154**, 1983, pp. 28–38.

[Be] Berger, R., The Undecidability of the Domino Problem, *Mem. Amer. Math. Soc.* **66** (1966).

[Bo] Boom, H.J., A Weaker Precondition for Loops, *ACM Trans. Prog. Lang. Syst.* **4**, (1982), 668–677.

[C] Chandra, A.K., Computable Nondeterministic Functions, *19th IEEE Symp. Found. of Comp. Sci.*, 127–131, 1978.

[D] Dijkstsra, E.W., *A Discipline of Programming,* Prentice Hall, Englewood Cliffs, NJ, 1976.

[EC] Emerson, E.A. and E.M. Clarke, Characterizing Correctness Properties of Parallel Programs Using Fixpoints. *Proc. 7th Int'l. Colloq. Automata, Lang. Prog.,* Springer-Verlag LNCS **85**, 1980, pp. 169–181.

[F] Francez, N., *Fairness,* Manuscript, 1983.

[FK] Francez, N. and D. Kozen, Generalized Fair Termination, *Proc. 11th ACM Symp. on Princ. Prog. Lang.,* 1984.

[Fü] Fürer, M., Alternation and the Ackermann Case of the Decision Problem, *L'Enseignement mathematique,* T.XXVII, fasc 1–2 (1981), 137–162.

[GF] Grumberg, O. and N. Francez, A Complete Proo Rule for Weak Equifairness, IBM RC-9634, 1982.

[GFMR] Grümberg, O., N. Francez, J.A. Makowsky, W.P. de Roever, A Proof Rule for Fair Termination of Guarded Commands, *Proc. of the Int. Symp. o Algorithmic Languages,* Amsterdam, October, 1981.

[H1] Harel, D., Recurring Dominos: Making the Highly Undecidable Highly Understandable, (*Proc. Int'l. Conf. Fund. Comput. Theory,*, Burgholm, Sweden, 1983.) *Ann. Discrete Math.*, In press.

[H2] Harel, D., A Simple Highly Undecidable Domino Problem (or, A Lemma on Infinite Trees, with Applications), *Proc. Logic and Computation Conference.* Clayton, Victoria, Australia, Jan. 1984.

[HPS] Harel, D., A. Pnueli and J. Stavi, Propositional Dynamic Logic of Nonregular Programs, *J. Comput. Syst. Sci.* **26** (1983), 222–243.

[H] Hoare, C.A.R., Communicating Sequential Processes, *Comm. ACM* **21** (1978), 666–677.

[LPS] Lehmann, D., A. Pnueli and J. Stavi, Impartiality, Justice and Fairness: The Ethics of Concurrent Termination, *Proc. 8th Int'l. Colloq. on Automata, Lang. and Programming,* Springer-Verlag LNCS **115**, 1981.

[L] Lewis, H.R., Complexity of Solvable Cases of the Decision Problem for the Predicate Calculus, *19th IEEE Symp. Found. Comput. Sci.,* 35–47, 1978.

[PA] Park, D., A Predicate Transformer for Weak Fair Iteration, *Proc. 6th IBM Symp. o Math. Found. Comput. Sci.*, Hakone, Japan, 1981.

[P] Pnueli, A., On the Extremely Fair Treatment of Probabilistic Algorithms, *Proc. 15th ACM Symp. on Theory of Computing,* 1983, pp. 278–290.

[QS] Queille, J.P. and J. Sifakis, Fairness and Related Properties in Transition Systems—A Temporal Logic to Deal with Fairness, *Act. Informatica* **19** (1983), 195–220.

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

[S] Streett, R.S. Global Process Logic is Π_1^1-Complete, Manuscript, 1982.

[W1] Wang, H., Proving Theorems by Pattern Recognition II, *Bell Syst. Tech. J.* **40** (1961), 1–41.