INFINITE TREES, MARKINGS AND WELL FOUNDEDNESS

By

Ran Rinat, Nissim Francez and Orna Grumberg

Department of Computer Science
Technion- Israel Institute of Technology
Haifa, Israel.

December 1985

Abstract
A necessary and sufficient condition for a given marked tree to have no infinite paths satisfying a given formula is presented. The formulas are taken from a language introduced by Harel, covering a wide scale of properties of infinite paths, including most of the known notions of fairness. This condition underlies a proof rule for proving that a nondeterministic program has no infinite computations satisfying a given formula, interpreted over state sequences. We also show two different forms of seemingly more natural necessary and sufficient conditions to be inadequate.

1. Introduction
The problem of finding a sound and complete proof rule for proving that a given nondeterministic program terminates under a certain fairness assumption has been solved for various notions of fairness (e.g [AO 83, APS 84, FK 84, GFK 83, GFMR 81, LPS 81, PN 83]). In order for a program to be fairly terminating under any given notion of fairness, it has to admit no infinite fair computations, where the definition of a fair computation varies from one version of fairness to another and from one model of computation to another.

A most convenient way of defining the semantics of a nondeterministic program is by using a tree, the vertices of which correspond to the intermediate states of computations. Thus, all these notions of fairness can be viewed as conditions on paths in the computation tree. Therefore, a language in which general conditions on paths in trees can be expressed can generalize all the already discussed types of fairness. One such a language, called $L$, is introduced in [HA 84]. It is also shown there how three notions of fairness can be expressed within $L$. Many of other versions of fairness, along with (what is claimed there as) any sensible condition one can think of, can also be expressed in $L$.

An important result of [HA 84] is a recursive transformation that, given a tree $T$ and a formula $\varphi \in L$, yields a tree $T''$, the infinite paths of which correspond to the infinite paths of $T$ satisfying $\varphi$. Thus, using this transformation, proving that $T$ has no infinite paths satisfying $\varphi$ reduces to proving that $T''$ has no infinite paths at all.

Returning to the issue of fair termination, two major approaches for proving it have been suggested in the literature (for a comprehensive discussion of these issues see [F 85]):

1. The method of helpful directions [GFMR 81, LPS 81]:

According to this approach one defines a ranking of states by means of elements of a well-founded set. This ranking has to decrease according to rules derived
directly from the fairness notion at hand.

2. The method of explicit scheduler \([\text{AO} 83, \text{APS} 84, \text{OA} 84]\):

This approach is based on program transformation. By augmenting the given program using \textit{random assignments}, an explicit fair scheduler is incorporated into the program. Thus, it remains to prove that the resulting program ordinarily terminates, for which a standard proof method exists.

Thus, a natural goal is to provide generalizations of both methods to the context of languages like \(L\): For any \(\varphi \in L\), prove the absence (in a given program) of infinite computations satisfying \(\varphi\).

The application of the \textit{explicit scheduler} method to \(L\) is pursued in [\text{DH} 85], though they use a different kind of explicit scheduler then used in [\text{AO} 83]. In this paper we pursue the alternative approach of \textit{helpful directions}, directly connecting the computation trees and their specifications in \(L\) with decreasing well founded rankings. We deal here with a subset of \(L\) called \(L^-\), containing all formulas in \(L\) having no infinite conjunctions or disjunctions. Weak and strong fairness can be expressed in \(L^-\) ([\text{HA} 84]), but extreme fairness requires an infinite formula, and thus cannot be expressed in \(L^-\). We remain in the level of trees, and the actual result concerns necessary and sufficient conditions, phrased in terms of decreasing well founded rankings, for a tree \(T\) to have no infinite paths satisfying a formula \(\varphi \in L^-\). This condition is intended to underly a syntactic proof rule for a programming language, having such trees as meaning for its programs.

Section 2 presents the language \(L^-\). In section 3 the necessary and sufficient condition is presented. In section 4 we prove its correctness, and in section 5 we show the impossibility of two, seemingly simpler and more natural, forms of necessary and sufficient conditions.

2. Basic definitions

We first define the trees, to which we refer. A \textit{node} is a finite sequence of natural numbers (i.e. an element of \(N^*\)) and a \textit{tree} is a set of nodes (i.e. a subset of \(N^*\)) closed under the prefix operation. The \textit{root} of the tree is \(\varepsilon\), and a \textit{path} is a maximal increasing sequence of successive nodes (by the prefix ordering) starting at \(\varepsilon\). Parts of a path are termed \textit{path-fragments}, or just \textit{fragments}. A node is a \textit{leaf} if it is the last element of a (finite) path. An example of a tree is shown in figure 1. A tree is \textit{well founded} if all its paths are finite.

Let \(\Sigma\) be some fixed (possibly infinite) alphabet. A \(\Sigma\)-\textit{marked tree} is one in which nodes are marked with (possibly infinitely many) letters from \(\Sigma\), i.e a tree \(T\) comes complete with a marking predicate \(M_T \subseteq T \times \Sigma\).

Throughout this paper, we refer to \textit{recursive} marked trees, i.e marked trees for which two algorithms exist: one that given an element of \(N^*\) decides whether it is a node in the tree and of which kind (leaf or internal). The other decides, given a node \(v\) in the tree and a mark \(a\), whether \(v\) is marked with \(a\). Keeping in mind that the trees are to be regarded as denotations of programs, we discuss only recursive trees, and thus we do not specify this explicitly unless needed.

We now define the language \(L^-\) for stating properties of infinite paths in a marked tree (we repeat the definition of \(L\) from [\text{HA} 84] omitting infinite disjunctions and conjunctions). An \textit{atomic formula} is an expression of one of the forms \(\exists a\), \(\forall a\), \(\exists^\omega a\) or \(\forall^\omega a\), where \(a \in \Sigma\) is a mark. \(L^-\) is the closure of the atomic formulas under finite conjunctions and disjunctions. Note the absence of negation from \(L^-\) (and \(L\)).

We interpret the formulas of \(L^-\) over infinite paths in a marked tree as follows: Let \(\varphi\) be an atomic formula and let \(\pi\) be an infinite path. \(\pi\) \textit{satisfies} \(\varphi\) (\(\pi \models \varphi\)) if either

- (a) \(\varphi = \exists a\) and there is a node on \(\pi\) marked with \(a\).
- (b) \(\varphi = \forall a\) and all the nodes on \(\pi\) are marked with \(a\).
- (c) \(\varphi = \exists^\omega a\) and there are infinitely many nodes on \(\pi\) marked with \(a\).
- (d) \(\varphi = \forall^\omega a\) and there is a vertex in \(\pi\) from which all the nodes are marked with \(a\).