Limit Operators and Convergence Measures for ω-Languages with Applications to Extreme Fairness

Nils Klarlund*

TR 90-1100
February 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Partially supported by a grant from the University of Aarhus, Denmark; NSF grant CCR-88-06979; and the Thanks to Scandinavia Foundation, Inc. The author's electronic address is klarlund@cs.cornell.edu.

Limit Operators and Convergence Measures for ω-Languages with Applications to Extreme Fairness

Nils Klarlund[1]
Cornell University

Abstract Methods of program verification for liveness and fairness rely on measuring "progress" of finite computations towards satisfying the specification. Previous methods were based on explaining progress in terms of *well-founded sets*. These approaches, however, often led to complicated transformations or inductive applications of proof rules.

Our main contribution is a simpler measure of progress based on an ordering that is *not* well-founded. This ordering is a variation on the Kleene-Brouwer ordering on nodes of a finite-path tree. It has the unusual property that for any infinite ordered sequence of nodes, the *liminf* of the node depths (levels) is finite.

This novel view of progress gives a precise mathematical characterization of what it means for programs to satisfy very general program properties. In particular, we solve the problem of finding a progress measure for termination under *extreme fairness*.

1 Introduction

A multitude of liveness concepts, like general and extreme fairness, have been suggested for specification of the infinite computations of programs. At least as many different methods have been suggested for verifying that a program satisfies such a specification. Most have relied on various notions of "progress" towards satisfying the specification. These notions, which were often very complicated and formulated in opaque syntax-dependent terms, were based on expressing progress in terms of well-founded sets. For some

[1]Partially supported by a grant from the University of Aarhus, Denmark; NSF grant CCR 88-06979; and the Thanks to Scandinavia Foundation, Inc.

Author's electronic address is klarlund@cs.cornell.edu.

concepts, like termination under extreme fairness [Pnu83,Fra86], no measure of progress has been found.

Our main contribution is a simpler measure of progress formulated in terms of an ordering that we call the *LimInf ordering*. It is a variation on the well-founded Kleene-Brouwer ordering on nodes of finite-paths trees. Instead of being well-founded, our ordering has the property that for any infinite ordered sequence of nodes, the liminf of the node depths (levels) is finite.

Our analysis of progress yields a universal verification method, which significantly extends the class of properties for which progress can be measured. In particular, we show how to measure termination under extreme fairness.

2 Our Results

By their very nature, computations (infinite sequences of program states) are usually regarded as limits of finite computations. In particular, this is true for the *verification problem*, which is to verify that a program Π satisfies a specification Σ. Thus the verification problem is to determine whether $L(\Pi) \subseteq L(\Sigma)$, where Π and Σ define sets of computations denoted $L(\Pi)$ and $L(\Sigma)$. In practice, one always prefers to solve the verification problem by considering only *finite* computations instead of showing that every *infinite* computation of Π is a computation of Σ. Reasoning about finite computations can be done using *invariants* or other techniques that establish a correspondence between Π and Σ. These methods often allow direct and intuitive explanations of why Π satisfies Σ. Also, they are more manageable, because the set of finite computations is countable, whereas the set of infinite computations can be uncountable.

To be more precise, we define a program Π to be a pair $(f_\Pi, \lim_\Pi)$, where f_Π is a *finite approximation function* describing finite computations and $\lim_\Pi$ is a *limit operator* defining $L(\Pi) = \lim_\Pi f_\Pi$, which is the set of computations of Π obtainable as limits of finite computations. Similarly, a specification Σ

has the form $(f_\Sigma, \lim_\Sigma)$.

An important question is: For which programs Π and specifications Σ can the verification problem be solved by looking only at *finite* computations?

In this paper, we investigate this question by introducing the notion of *convergence measure*. We show how to measure finite computations in terms of f_Π and f_Σ such that the lengthening of any finite computation allows the measure to change in a way that implies either progress (convergence) towards not being in $L(\Pi)$ or progress towards being in $L(\Sigma)$. When such a measure exists, we say that the pair (f_Π, f_Σ) is *convergence measurable.*

Our main contribution is to

- focus on the limit operators $\lim_{\mathbf{F}_{\sigma\delta}}$ and $\lim_{\mathbf{G}_{\delta\sigma}}$—which define the class $\mathbf{F}_{\sigma\delta}$ and the class $\mathbf{G}_{\delta\sigma}$ in the Borel Hierarchy—as a simple way of specifying sets of computations, and

- provide a characterization in terms of finite sequences of when an $\mathbf{F}_{\sigma\delta}$ program $\Pi = (f_\Pi, \lim_{\mathbf{F}_{\sigma\delta}})$ satisfies a $\mathbf{G}_{\delta\sigma}$ specification $\Sigma = (f_\Sigma, \lim_{\mathbf{G}_{\delta\sigma}})$, to wit:
  $L(\Pi) \subseteq L(\Sigma)$ *if and only if* (f_Π, f_Σ) *is convergence measurable.*

In particular, this characterization yields a verification method for program termination under *extreme fairness*. An extreme fairness constraint is an infinite set of possible ways of executing a program unfairly. A program terminates under an extreme fairness condition if each non-terminating computation is unfair. We show that termination under extreme fairness is $\mathbf{G}_{\delta\sigma}$ and that it can be directly measured.

3 Previous Work

Techniques similar to ours for program verification have been presented for various kinds of finite and infinite automata. For example, for a program Π and a specification Σ, which are both represented as *deterministic* infinite-state automata (or transition systems), the verification problem is easily solved in terms of finite computations. It suffices, in fact, to find an *invariant relation* that associates states of Σ with each state of Π and to verify that the correspondence between Π and Σ is preserved for each program state added to a finite run of Π, see e.g. [Jon87,LT87] or [AL88,KS89,Sis89a], which deal with finitely-branching nondeterministic automata.

Unfortunately, the simple limit operation of deterministic automata—an infinite sequence is an infinite computation iff all finite prefixes are finite computations—can define only *closed* sets in the usual product topology on infinite sequences. Many properties about programs, like liveness properties, are not closed sets. Some of these liveness properties, including some fairness properties, can be dealt with using deterministic automata with Büchi or Rabin acceptance conditions for which various methods have been devised [AS87,AS89,Kla90]. Such automata correspond to limit operators that define certain finite Boolean combinations of $\mathbf{G}_\delta$ and $\mathbf{F}_\sigma$ sets in the Borel Hierarchy. Hence they are not as powerful as our limit operators.

The use of the more powerful *nondeterministic* automata has also been investigated. Unfortunately, no reasonable verification method based on finite computations can be complete for nondeterministic programs and specifications [Sis89b]. Besides, the concept of nondeterminism does not directly define a computation as a limit: it involves a limit in the combined state and alphabet space, followed by a projection on the alphabet. A similar objection can be raised towards using automata with "and"-nondeterminism (where a word is recognized if *all* runs over the word are accepting). Verification methods for such automata can be found in [MP87,Kla90].

For extreme fairness, there has to our knowledge not been any method of directly measuring progress. Earlier universal verification methods [DH86, Har86,Var87]—which were also applicable to extreme fairness—and the method in [Mai89] relied on extensive modifications of the program to be verified in order to reduce progress to something that could be measured in terms of well-founded sets.

4 The LimInf Ordering

The main result in this section is formulated in terms of height graphs and countably branching trees. A *height graph* $G = (V, E, n)$ is a countable directed graph (V, E), where $n : V \to \omega$ is the *height function* assigning a natural number to each node. Note that the edge relation $E \subseteq V \times V$ can also be viewed as an ordering "$\rightarrow$" on V, where $v \to v'$ iff $(v, v') \in E$. An infinite path $v_0, v_1, \ldots$ in G has the *LimInf property* if $\liminf_{i \to \omega} n(v_i) < \omega$ and graph G has the LimInf property if all infinite paths in G has the LimInf property.

Let ω^* be the set of finite sequences $\langle d^0, \ldots, d^\ell \rangle$ of natural numbers, where $\ell \geq -1$. A tree T is a non-empty prefix closed subset of ω^*, i.e. the *root* $\langle\rangle$ is in T and for each *node* $t \cdot \langle d \rangle$ in T, the *parent* t is also in T. Here "$\cdot$" denotes catenation, $t \cdot \langle d \rangle$ is a *child* of t, and d is the *direction* of $t \cdot \langle d \rangle$ from t. Any sequence of directions $\langle d^0, d^1, \ldots \rangle$ (finite or infinite) denotes a path $\langle\rangle, \langle d^0 \rangle, \langle d^0, d^1 \rangle, \ldots$ (finite or infinite) in T. The *level* $|t|$ of a node $t = \langle t^0, \ldots, t^\ell \rangle$ is the number ℓ; note that the level of $\langle\rangle$ is -1. We view trees as growing downwards with the children ordered from left to right in increasing order.

We can now state the Liminf ordering, which we denote by $\succeq$:

Definition 1 *$t \succeq t'$ if t is a prefix of t' or t' is a prefix of t or there is a level ℓ such that $t^\ell > t'^\ell$ and for all $\lambda < \ell$, it holds that $t^\lambda = t'^\lambda$.*

Intuitively, $t \succeq t'$ if either t' extends t or is a prefix of t or branches off to the left of t. If we omitted "t' is a prefix of t" from the definition, we would just have the Kleene-Brouwer order, which is well-founded [Rog67].

Proposition 1 *Let $t_0 \succeq t_1 \succeq \cdots$ be an infinite sequence of nodes in a finite-path tree T. Then $\liminf_{i \to \omega} |t_i| < \omega$.*

Proof Suppose for a contradiction that $\liminf |t_i| = \omega$. Let $t_k = \langle t_k^0, \ldots, t_k^{|t_k|} \rangle$, for $k \geq 0$. Then for all ℓ, there is a K^ℓ such that for all $k \geq K^\ell$, $|t_k| \geq \ell$. Let $\ell = 0$. Then for any $k \geq K^0$, t_k^0 is defined and by definition of $\succeq$, we have $t_K^0 \geq t_{K+1}^0 \geq \cdots$. Hence, there is an H^0 and a d^0 such that for all $k \geq H^0$, $t_k^0 = d^0$.

Now, we inductively choose H^ℓ's and find d^ℓ's such that for all ℓ and $k \geq H^\ell$, it holds that $\langle t_k^0, \ldots, t_k^\ell \rangle = \langle d^0, \ldots, d^\ell \rangle$.

Thus $\langle d^0, \ldots, d^\ell \rangle \in T$. As this holds for all ℓ, $\langle d^0, d^1, \ldots \rangle$ is an infinite path in T. Contradiction. □

The connection between the LimInf property of height graphs and the LimInf ordering is defined in terms of height homomorphisms:

Definition 2 *A* height homomorphism $h : G \rightarrow T$ *maps each* $v \in V$ *to a node* $h(v) \in T$ *such that* $n(v) = |h(v)|$*; further* h *must respect order, i.e. if* $v \rightarrow v'$*, then* $h(v) \succeq h(v')$.

Proposition 2 $G = (V, E, n)$ *has the LimInf property if and only if there is a finite path tree* T *with the ordering* $\succeq$ *and a height homomorphism* $h : G \rightarrow T$.

Proof "Only if" If $v_0, v_1, \ldots$ is an infinite path in G, then $h(v_0) \succeq h(v_1) \succeq \cdots$. It follows from Proposition 1 that $\liminf |h(v_i)| < \omega$. Now because h preserves heights, we get $\liminf n(v_i) = \liminf |h(v_i)| < \omega$.

"If" Given that G has the LimInf property, we show how to construct a finite path tree T and a height homomorphism $h : G \rightarrow T$. To do this we need the following lemma.

> **Lemma 1** *Let* $G = (V, E)$ *be a non-empty countable graph. Then there exists a finite or infinite partitioning* $\mathcal{P} = W_0, W_1, \ldots$ *of* V*; a mapping* ι *that to each class* W *of* $\mathcal{P}$ *associates a node* $\iota(W) \in W$ *such that* $\iota(W) \rightarrow_W^* W$*; and a mapping* $\alpha(v)$*—defined by* $\alpha(v) = k$ *for the* k *satisfying* $v \in W_k$*—such that for all* $(v, v') \in E$, $\alpha(v) \geq \alpha(v')$*. Here,* $w \rightarrow_W w'$ *denotes that there is a path in* W *from* w *to* w'.
>
> **Proof** For $w \in W \subseteq V$, let $\mathcal{R}(w, W) = \{w' \mid w \rightarrow_W w'\}$. Now, assume that V is ordered $v_0, v_1, \ldots$ Let $W_0 = \{v \mid v \in \mathcal{R}(w_0, V)\}$, where w_0 is v_0, the least node in V. If $V \setminus \bigcup_{j<k} W_j$ is not empty, W_k is defined as $\{v \mid v \in \mathcal{R}(w_k, V \setminus \bigcup_{j<k} W_j\}$, where w_k is the least node in $V \setminus \bigcup_{j<k} W_j$. Then $W_0, W_1, \ldots$ are disjoint subsets of V. Either there is a k such that $\bigcup_{j<k} W_j = V$ or W_k is defined for all k. In the latter case, $\bigcup_j W_j = V$; for

`AssignLimInf`(W,t):

1. Let $\widehat{W} = \{v \in W \mid n(v) \geq |t|\}$.
   If $\widehat{W} = \emptyset$, then exit.
2. Use Lemma 1 to obtain a partitioning $\mathcal{P} = W_0, W_1, \ldots$ of $\widehat{W}$.
3. For each class W_k of $\mathcal{P}$:
   (a) $T := T \cup \{t \cdot \langle k \rangle\}$
   (b) $W(t \cdot \langle k \rangle) = W_k$,
   (c) `AssignLimInf`$(W_k, t \cdot \langle k \rangle)$

Figure 1: The algorithm `AssignGen`.

otherwise there is a least node v_h in $V \setminus \bigcup_j W_j$ and as there are infinitely many nodes w_k, there is a K such that $w_K = v_H$ with $H > h$. This would contradict that w_K is minimal.

Let $\iota(W_k) = w_k$. Then it is clearly the case that $\iota(W_k) \rightarrow_{W_k} W_k$. Now, suppose for a contradiction that $v \rightarrow v'$ with $v \in W_i$ and $v' \in W_{i'}$ for $i < i'$. Then $v' \in \mathcal{R}(w_i, V \setminus \bigcup_j W_{j<i})$. Thus, $v' \in W_i$. Contradiction. Hence, $\alpha(v) \geq \alpha(v')$. □

Now to construct the tree T, we use the algorithm in Figure 1. Each node t of T has a label $W(t)$, which is a subset of V. The algorithm `AssignLimInf` is initially invoked with $W = V, t = \langle\rangle$, and variable $T = \{\langle\rangle\}$.

The purpose of `AssignLimInf`(W,t) is to define the children of t together with labels, which designate subsets of V. First, nodes of height less than $|t|$ are filtered out. Second, the resulting set $\widehat{W}$ is partitioned using Lemma 1. Third, a child in direction k with a label W_k is created for each class W_k of the partitioning and the algorithm is reapplied on W_k.

It can be seen that the algorithm corresponds to a monotonic set operator w.r.t. the values of T. Thus, we take the final value of T to be the fixed point of the corresponding set operator. In addition, we define $W(\langle\rangle) = V$.

Note that to each label $W(t)$, where $t \neq \langle\rangle$, is associated a node $\iota(W(t))$

of G such that $\iota(W(t)) \to^*_{W(t)} W(t)$, because $W(t)$ is a class defined by an application of Lemma 1.

Claim 1

(1) For all $t \in T$, for all $v \in W(t)$, $n(v) \geq |t|$.

(2) For any $t \cdot \langle d \rangle \in T$, it holds that $W(t) \supseteq W(t \cdot \langle d \rangle)$. For each $v \in V$, there is a unique path $\langle d^0, \ldots, d^{n(v)} \rangle$ of T containing all nodes whose labels contain v.

Proof Follows by easy induction. □

Now using (2) of Claim 1, we define $h(v) = \langle d^0, \ldots, d^{n(v)} \rangle$. We prove that h is a height homomorphism.

First, it is obvious that $n(v) = |h(v)|$. Second, assume that $u, v \in V$ and $u \to v$. Let $h(u) = \langle d^0, \ldots, d^{n(u)} \rangle$, $h(v) = \langle e^0, \ldots, e^{n(v)} \rangle$ and $m = \min\{n(u), n(v)\}$. If $d^\ell = e^\ell$ for all $\ell \leq m$, then $h(u)$ is a prefix of $h(v)$ or $h(v)$ is a prefix of $h(u)$, i.e. $h(u) \succeq h(v)$. Otherwise, assume that $\lambda \leq m$ is such that $d^\lambda \neq e^\lambda$ and $d^\ell = e^\ell$ for all $\ell < \lambda$. Let $t = \langle d^0, \ldots, d^{\lambda-1} \rangle = \langle e^0, \ldots, e^{\lambda-1} \rangle$. Then $\texttt{AssignLimInf}(W(t), t)$ was invoked and $u, v \in W(t)$. This invocation defined $W(t \cdot d^\lambda)$ and $W(t \cdot e^\lambda)$, which are disjoint subsets of $W(t)$ by assumption that $d^\lambda \neq e^\lambda$. Hence, $d^\lambda > e^\lambda$ by Lemma 1. It follows that $h(u) \succeq h(v)$.

Claim 2 *There is no infinite path in T.*

Proof Suppose for a contradiction that $\langle d^0, d^1, \ldots \rangle$ is an infinite path. Then $W(\langle d^0 \rangle) \supseteq W(\langle d^0, d^1 \rangle) \supseteq \cdots$ and therefore for any $t = \langle d^0, \ldots, d^k \rangle \neq \langle \rangle$, it holds that $\iota(W(t)) \to^*_{W(t)} W(t \cdot \langle d^{k+1} \rangle)$, because $\iota(W(t)) \to^*_{W(t)} W(t)$, by Lemma 1. In particular, $\iota(W(t)) \to^*_{W(t)} \iota(W(t \cdot \langle d^{k+1} \rangle))$. We conclude that there exists an infinite path $v_0 \to v_1 \to \cdots$ in G and a monotonic increasing function $i(k)$ such that $v_{i(k)} = \iota(W(\langle d^0, \ldots, d^k \rangle))$ for $k \geq 0$, and for all $h \geq i(k)$, it holds by (1) of Claim 1 that $n(v_h) \geq k$. But then $v_0 \to v_1 \to \cdots$ does not have the LimInf property. Contradiction. □

5 LimInf Operations and Convergence Measure

A limit operator is a way of defining sets of infinite sequences of natural numbers, i.e. subsets of ω^ω, as limits of finite sequences. There are well-known limit operators, which we denote $\lim_{\mathbf{F}}$, $\lim_{\mathbf{G}}$, $\lim_{\mathbf{F}_\sigma}$, $\lim_{\mathbf{G}_\delta}$, defining the class $\mathbf{F}$ of closed sets, the class $\mathbf{G}$ of open sets, the class $\lim_{\mathbf{F}_\sigma}$ of countable unions of closed sets, and the class $\lim_{\mathbf{G}_\delta}$ of countable intersections of open sets. Let $f : \omega^* \to \{0,1\}$ be a finite approximation function. Then $w \in \omega^\omega$ is in the limit of f according to:

$$
\begin{array}{lll}
(1) & w \in \lim_{\mathbf{F}} f & \text{iff } \forall u \propto w : f(u) = 1 \\
(2) & w \in \lim_{\mathbf{G}} f & \text{iff } \exists u \propto w : f(u) = 1 \\
(3) & w \in \lim_{\mathbf{F}_\sigma} f & \text{iff } \forall^\infty u \propto w : f(u) = 1 \\
(4) & w \in \lim_{\mathbf{G}_\delta} f & \text{iff } \exists^\infty u \propto w : f(u) = 1
\end{array}
$$

where $u \propto w$ denotes that u is a finite prefix of w, $\forall^\infty$ means "for all but finitely many," and $\exists^\infty$ means "for infinitely many." It is not hard to prove that the limit operators define the respective classes [Arn83,Lan69].

Using finite approximation functions of type $f : \omega^* \to \omega$, we introduce limit operators $\lim_{\mathbf{G}_{\delta\sigma}}$ and $\lim_{\mathbf{F}_{\sigma\delta}}$:

$$
\begin{array}{llll}
(5) & w \in \lim_{\mathbf{F}_{\sigma\delta}} f & \text{iff } \forall \ell : \forall^\infty u \propto w : f(u) \neq \ell & \text{iff } \liminf_{u \to w} f(u) = \omega \\
(6) & w \in \lim_{\mathbf{G}_{\delta\sigma}} f & \text{iff } \exists \ell : \exists^\infty u \propto w : f(u) = \ell & \text{iff } \liminf_{u \to w} f(u) < \omega
\end{array}
$$

where $u \to w$ denotes that u takes the values $\langle\rangle, \langle w_0\rangle, \langle w_0, w_1\rangle, \ldots$ for $w = \langle w_0, w_1, \ldots\rangle$.

Proposition 3 *Limit operators* $\lim_{\mathbf{F}_{\sigma\delta}}$ *and* $\lim_{\mathbf{G}_{\delta\sigma}}$ *define the class* $\mathbf{F}_{\sigma\delta}$ *of countable intersections of* $\mathbf{F}_\sigma$ *sets and the class* $\mathbf{G}_{\delta\sigma}$ *of countable unions of* $\mathbf{G}_\delta$ *sets.*

Proof We only give a proof for $\mathbf{G}_{\delta\sigma}$, the proof for $\mathbf{F}_{\sigma\delta}$ being obtainable by dualization. A set $\lim_{\mathbf{G}_{\delta\sigma}} f$ is a $\mathbf{G}_{\delta\sigma}$ set, because it can be written as a union of $\mathbf{G}_\delta$ sets:

$$G = \bigcup_i \lim_{\mathbf{G}_\delta} f_i,$$

where

$$f_i(u) = \begin{cases} 1 & \text{if } f(u) = i \\ 0 & \text{if } f(u) \neq i \end{cases}$$

On the other hand, if G is a union $\bigcup_i \lim_{\mathbf{G}_\delta} f_i$ of $\mathbf{G}_\delta$ sets, then define

$$f(u) = \begin{cases} i & \text{if } i \text{ is minimal s.t. } f_i(u) = 1 \\ |u| & \text{otherwise} \end{cases}$$

It follows that $w \in \lim_{\mathbf{G}_{\delta\sigma}}$ iff $\exists i : \exists^\infty u : f(u) = i$ iff $\exists i : \exists^\infty u : f_i(u) = 1$ iff $\exists i : w \in \lim_{\mathbf{G}_\delta}$. □

To study the problem of whether $F \subseteq G$, where F is $\mathbf{F}_{\sigma\delta}$ and G is $\mathbf{G}_{\delta\sigma}$, in terms of finite approximation functions, we introduce a new concept of measure to describe the informal notion of "making progress" each time a finite sequence is extended.

Definition 3 *A* convergence measure (T, μ) *for* (f, g) *is a finite path tree* T *and a mapping* $\mu : \omega^* \to T$*, where* μ *is a height homomorphism with respect to the parent relation* $\to$ *on* ω^* *and height function* $u \mapsto \min\{f(u), g(u)\}$.

Here, $u \to u'$ if $u' = u \cdot \langle d \rangle$ for some d. Our main result is:

Theorem 1 *For any* $\mathbf{F}_{\sigma\delta}$ *set* $F = \lim_{\mathbf{F}_{\sigma\delta}} f$ *and any* $\mathbf{G}_{\delta\sigma}$ *set* $G = \lim_{\mathbf{G}_{\delta\sigma}} g$,

$F \subseteq G$

iff

(f, g) *is convergence measurable.*

Proof "Only if." Assume that (f, g) has measure (T, μ). Let $w = \langle w_0, w_1, \ldots \rangle \in F$, then $\liminf_{u \to w} f(u) = \omega$. As μ is a height homomorphism and $\langle\rangle \to \langle w_0 \rangle \to \langle w_0, w_1 \rangle \to \cdots$, we get an ordered sequence

$$\mu(\langle\rangle) \succeq \mu(\langle w_0 \rangle) \succeq \mu(\langle w_0, w_1 \rangle) \succeq \cdots,$$

so by Proposition 2, $\liminf_{u \to w} \min\{f(u), g(u)\} < \omega$. We conclude that $\liminf_{u \to w} g(u) < \omega$, i.e. $w \in G$.

"If." If $F \subseteq G$, then for all w, it holds that $\liminf_{u \to w} \min\{f(u), g(u)\} < \omega$. Use Proposition 2 to obtain T and μ. □

6 The Invariant Method for LimInf Specifications

Using the methods of invariants, we formulate a verification method for any program property that is in $\mathbf{G}_{\delta\sigma}$. This method only applies to programs that are closed sets, but it is included to illustrate the connection of our work to previous methods.

A program Π consists of a set of program states Q_Π, an initial state $p^0 \in Q_\Pi$ and a nondeterministic transition relation $p \to p'$, which denotes that an atomic action of Π may transform the program state from p to p'. A *computation* of Π is an infinite sequence $p_0 \to p_1 \to \cdots$ such that $p_0 = p^0$.

A *LimInf specification* Σ is a $\mathbf{G}_{\delta\sigma}$ set given as a function $f_\Sigma : Q_\Pi^* \to \omega$. A sequence $p_0, p_1, \ldots$ *satisfies* Σ (i.e. is in Σ) if $\liminf_{i \to \omega} f_\Sigma(\langle p_0, \ldots, p_i \rangle) < \omega$. Program Π satisfies Σ if all computations of Π satisfy Σ.

We can now state a sound and complete verification method:

Theorem 2 *Π satisfies Σ if and only if there is a finite path tree T and a relation $\mathcal{I} \subseteq Q_\Pi^* \times T$ such that*

(V0) $\forall u, t : \mathcal{I}(u, t) \Rightarrow |t| = f_\Sigma(u)$

(V1) $\exists t^0 : \mathcal{I}(\langle p^0 \rangle, t^0)$

(V2) $\forall p, p', u, t : p \to p' \wedge \mathcal{I}(u \cdot \langle p \rangle, t) \Rightarrow \exists t' : t \succeq t' \wedge \mathcal{I}(u \cdot \langle p, p' \rangle, t')$

Proof Straightforward □

Here, (V0), (V1), (V3) are the *verification conditions* that an *invariant* $\mathcal{I}$ must fulfill in order to assure that Π satisfies Σ. (V0) states that a node t associated with u by the invariant must be at level $f_\Sigma(u)$; (V1) assures that there is node t^0 corresponding to the initial program state p^0; and (V2) assures that the invariant is preserved for any program state p, new program state p' and history u, i.e. if t is any node associated with $u \cdot \langle p \rangle$, then there is a node t' associated with $u \cdot \langle p, p' \rangle$ such that $t \succeq t'$.

7 Extreme Fairness

An *extreme fairness constraint* F for a program Π is a countable set $\{(\phi_i,\psi_i)\}_{i>0}$ of *unfairness conditions*, where the ϕ_i's and ψ_i's are predicates on the program state.

Each unfairness condition specifies a way of executing Π unfairly: a computation is *unfair w.r.t.* (ϕ_i,ψ_i) if *enabling condition* ϕ_i is satisfied infinitely often and *action condition* ψ_i is satisfied only finitely often. A computation is *unfair* if it is unfair w.r.t. some (ϕ_i,ψ_i).

Moreover, we say that a computation is *terminating* if it enters a terminating state[2] and that a program *terminates under* F if every non-terminating computation is unfair. If we let (ϕ_0,ψ_0) express that Π has reached a terminating state (e.g. by letting $\phi_0 =$ "state is terminating," $\psi_0 = \mathit{false}$), then Π terminates under F iff Π is unfair w.r.t. $F' = \{(\phi_i,\psi_i)\}_{i\geq 0}$.

Let $\Sigma = \{w | w \text{ is unfair w.r.t. } F'\}$. Then it is not hard to see that Σ is an infinite union of finite Boolean combinations of $\mathbf{G}_\delta$ (or $\mathbf{F}_\sigma$) sets. It follows that $\Sigma = \lim_{\mathbf{G}_{\delta\sigma}} f_\Sigma$ for some function $f : \omega^* \to \omega$. There is, however, a way of directly expressing f_Σ in terms of $\{(\phi_i,\psi_i)\}_{i\geq 0}$:

Lemma 2 *There is a function f_Σ stated in terms of the ϕ_i's and ψ_i's such that for any computation w:*

$$w \text{ is terminating or } w \text{ is unfair w.r.t. } F$$

iff

$$\liminf_{u\to w} f_\Sigma(u) < \omega.$$

Proof Define $f_i(u) = |u| - @\phi_i(u) + @\psi_i(u)$, where $@\theta(u)$ is the last index of u for which the predicate θ was satisfied:

$$@\theta(u) = \begin{cases} h & \text{if } h = \text{maximal } h \leq |u| \text{ s.t. } u_h \models \theta \\ 0 & \text{if there is no such } h \end{cases}$$

Then it is easy to see that $\liminf_{u\to w} f_i(u) < \omega$ iff w is unfair w.r.t. (ϕ_i,ψ_i). Now let $f_\Sigma(u) = \min\{k \mid f_{\pi_1(k)}(u) = \pi_2(k)\}$, where $\rho : \omega\times\omega \to \omega$ is a bijection, $\pi_1(\rho(x,y)) = x$, and $\pi_2(\rho(x,y)) = y$.

[2]We assume that once Π has entered a terminating state, it remains in that state.

It remains to prove that $\liminf_{u \to w} f(u) < \omega$ iff there exists an i s.t. $\liminf_{u \to w} f_i(u) < \omega$.

First, assume that $\liminf_{u \to w} f(u) < \omega$. Then there exists a k s.t. $\exists^\infty u \propto w : f(u) = k$. Thus, there are i and j such that $\rho(i,j) = k$ and $\exists^\infty u \propto w : f_i(u) = j$. Whence, $\liminf_{u \to w} f_i(u) \leq j < \omega$.

Second, assume that $\liminf_{u \to w} f_i(u) = j < \omega$ for some i and j. Then, $\exists^\infty u \propto w : f(u) \leq \rho(i,j)$. Hence, $\liminf_{u \to w} f(u) \leq \rho(i,j) < \omega$. □

By Lemma 2 and Theorem 2, it follows that termination under extreme fairness can be proved using the invariant method.

Acknowledgments

Thanks to Dexter Kozen for suggesting substantial simplifications and for pointing out the similarity of the LimInf ordering to the Kleene-Brouwer ordering. Also thanks to Steven Mitchell for comments on the paper.

References

[AL88] M. Abadi and L. Lamport. The existence of refinement mappings. In *Proc. 2. Symp. on Logic in Computer Science.* IEEE, 1988.

[Arn83] A. Arnold. Topological characterizations of infinite behaviors of transition systems. In *Proc. 10th Col. Automata, Languages and Programming*, pages 490–510. LNCS, Vol. 154, Springer-Verlag, 1983.

[AS87] B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2:117–126, 1987.

[AS89] B. Alpern and F.B. Schneider. Verifying temporal properties without temporal logic. *ACM Transactions on Programming Languages*, 11(1):147–167, January 1989.

[DH86] I. Dayan and D. Harel. Fair termination with cruel schedulers. *Fundamenta Informatica*, 9:1–12, 1986.

[Fra86] Nissam Francez. *Fairness.* Springer-Verlag, 1986.

[Har86] D. Harel. Effective transformations on infinite trees with applications to high undecidability, dominos, and fairness. *Journal of the ACM*, 33(1):224–248, 1986.

[Jon87] B. Jonsson. Modular verification of asynchronous networks. In *Proc. Sixth Symp. on the Principles of Distributed Computing*, pages 152–166. ACM, 1987.

[Kla90] N. Klarlund. Verification conditions for ω-automata and applications to fairness. Technical Report TR-1080, Cornell University, 1990.

[KS89] N. Klarlund and F.B. Schneider. Verifying safety properties using infinite-state automata. Technical Report TR-1036, Cornell University, 1989.

[Lan69] L.H. Landweber. Decision problems for ω-automata. *Math. System Theory*, 3:376–384, 1969.

[LT87] N. Lynch and M. Tuttle. Hierarchical correctness proof for distributed algorithms. In *Proc. Sixth Symp. on the Principles of Distributed Computing*, pages 137–151. ACM, 1987.

[Mai89] M.G. Main. Complete proof rules for strong fairness and strong extreme-fairness. Technical Report CU-CS-447-89, Department of Computer Science, University of Colorado, 1989.

[MP87] Z. Manna and A. Pnueli. Specification and verification of concurrent programs by $\forall$-automata. In *Proc. Fourteenth Symp. on the Principles of Programming Languages*, pages 1–12. ACM, 1987.

[Pnu83] A. Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proc. 15th ACM Symp. on Theory of Computing*, pages 278–290, 1983.

[Rog67] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill Book Company, 1967.

[Sis89a] A.P. Sistla. A complete proof system for proving correctness of nondeterministic safety specifications. Technical report, Computer and Intelligent Systems Laboratory, GTE Laboratories Inc., 1989.

[Sis89b] A.P. Sistla. On verifying that a concurrent program satisfies a nondeterministic specification. *Information Processing Letters*, 32(1):17–24, July 1989.

[Var87] M. Vardi. Verification of concurrent programs: The automata-theoretic framework. In *Proc. Symp. on Logic in Computer Science*. IEEE, 1987.