Liminf Progress Measures*

Nils Klarlund
IBM T.J. Watson Research Center
PO BOX 704
Yorktown Heights, NY 10598

Abstract Consider a program P that satisfies a specification S. It is natural to think that every single step of a computation of P somehow contributes to making the computation closer to satisfying S. For specifications that define liveness properties, methods that directly quantify such a notion of progress or convergence are often quite limited in scope. Instead many approaches, such as those that deal with termination under fairness, rely on program transformations, since there was no known way of directly expressing progress towards fair termination.

In this article we use the novel concept of *progress measures* to quantify progress for a large class of specifications. The main result is a precise characterization of what it means for a program to satisfy a *Liminf specification*. Such specifications define properties at a higher level of the Borel hierarchy than earlier methods based on Büchi automata and Propositional Temporal Logic. In particular, we give a solution to the problem of verification involving the very general notion of extreme fairness. More generally, our results establish a connection between assertional reasoning about programs and descriptive set theory.

1 Introduction

The art of programming is the art of understanding a complex world of events and situations in terms of small steps, each performing an atomic change of state. This is why many methods of verification, such as those based on the assertional techniques of Floyd and Hoare, build on explaining global behavior by local reasoning [4, 12, 18]. For example, the liveness property that a program terminates can be proved using well-founded sets, as Floyd pointed out in 1967 [12].

A *well-founded* set $(T, >)$ is a set T with a relation $>$ such that there is no infinite descending chain $t_0 > t_1 > \cdots$. To verify that a program terminates, one assigns an element $\mu(v) \in T$ to each program state v and checks that whenever there is a transition from v to v', then $\mu(v) > \mu(v')$ holds; Floyd termed such a local requirement a *verification condition*. The assignment $\mu(v)$ quantifies how

*This work was mainly carried out at Cornell University and supported by grants from the University of Aarhus, Denmark; Forskerakademiet, Aarhus; and the Thanks to Scandinavia Foundation Inc., NY.
Author's current address: Aarhus University, Department of Computer Science, Ny Munkegade, DK-8000 Aarhus C.

close the program is in state v to terminating. Therefore μ is a measure of progress, which we call a *termination measure*.

In this article we investigate how Floyd's ideas can be extended to verify properties of programs that define infinite computations. Such a program usually gives rise to uncountably many infinite computations. Therefore, to establish that the program satisfies a global property given by some specification, we would prefer to reason locally about finite computations, whose number is countable. We show that for a large class of properties such reasoning is possible by an approach very similar to Floyd's. Thus we can quantify that each step executed by the program makes the computation closer to satisfying the specification.

Our approach is based on a new kind of progress measure, called a *Liminf measure*. The liminf measure is formulated in terms of the *Liminf relation*, which is a variation on the well-founded Kleene-Brouwer ordering on nodes of finite-path trees [35]. Instead of being well-founded, our relation has the property that for any infinite descending sequence of nodes, the liminf of the node levels is finite.

Extensions of Floyd's ideas—for example involving fairness, liveness, temporal logic, or automata—have been the subject of numerous articles; see for example [1, 2, 3, 5, 10, 13, 16, 17, 26, 28, 29, 37, 39]. Many of the proposed methods are indirect, depending on repeated program transformations in order to reduce the original verification problem to problems that can be solved by standard techniques such as well-founded sets or refinement mappings.

Our analysis of progress yields a verification method that significantly extends the class of properties for which progress can be directly measured. In particular, we show how to measure progress towards termination of a program subjected to *extreme fairness constraints*. More generally, our results apply to descriptive set theory, where we show how to compare sets at the third level of the Borel hierarchy in terms of finite computations. Thus our results allow us to quantify progress for a large number of liveness properties, including all those expressible in *Propositional Temporal Logic*, since these are essentially at the second level of the Borel hierarchy (see [30]).

2 Motivation

Use of termination measures—and similar local descriptions, such as *variant* or *bound functions* for proving loop termination—is a cornerstone of program verification [15]. Termination measures are important because:

> A program terminates if and only if it has a termination measure.

Thus the method of termination measures is *sound*: if the program has a termination measure, then it terminates (a nonterminating computation $v_0, v_1, \ldots$ would give rise to an infinite descending sequence $\mu(v_0) > \mu(v_1) > \cdots$, which would contradict that $(D, >)$ is well-founded). The method is also *complete*: if

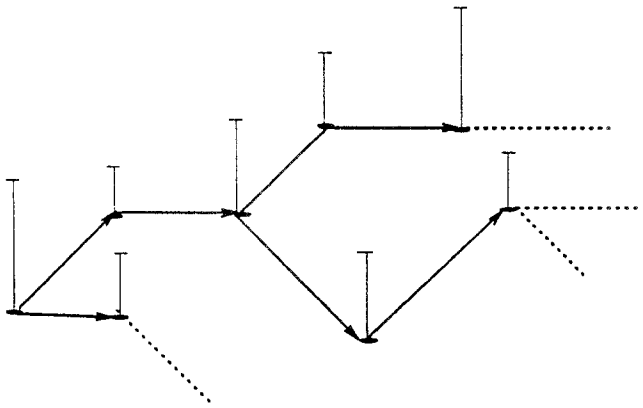

Figure 1: A graph with a height assignment.

a program terminates, then it has a termination measure (namely, $\mu(v) = v$, because the set of program states, with the relation $v > v'$ iff there is a transition from v to v', is a well-founded set).

In this article we generalize Floyd's method of using local descriptions to verify global program properties. To do this as abstractly as possible, we represent a program as a directed graph $G = (V, E)$, where the vertices V represent program states and the edges represent atomic state transitions. A program property or specification is defined by a *height assignment* $h(v)$, that to each vertex v assigns a natural number $h(v)$; see Figure 1. An infinite sequence $v_0, v_1, \ldots$ *satisfies* the *Liminf condition*, and we write $v_0, v_1, \ldots \models \liminf h$, if $\liminf_{i\to\infty} h(v_i)$ is finite, i.e. if $h(v_i)$ dips down to some fixed value infinitely often as i goes towards infinity. The graph G satisfies the Liminf condition, i.e. the property specified by h, and we write $G \models \liminf h$, if every infinite path in G satisfies the Liminf condition.

Many liveness properties can be specified as a Liminf condition. For a trivial example, assume that some of the vertices of G are marked with a $\surd$ (denoting a "good" state). The condition that along any path, infinitely many vertices are marked $\surd$, can be expressed by setting the height $h(v)$ of every vertex v marked with $\surd$ to zero and the height $h(v)$ of an unmarked vertex v to the height of the preceding vertex plus one (assuming that each vertex has one predecessor). If an infinite path $v_0, v_1, \ldots$ has infinitely many vertices marked $\surd$, then $\liminf h(v_i) = 0$; otherwise, if there are only finitely many marked vertices, then $\liminf h(v_i)$ is infinite. As we shall see, all finite *Propositional Temporal Logic* formulas and even some infinite ones, namely *countable Rabin conditions*, can be coded in this way when G is a tree (which it is if vertices represent finite computations.)

That the program G satisfies the specification h, i.e. that $G \models \liminf h$, is a *global* property of G, because it involves checking all infinite paths of G. What we want is a *local* way of verifying that $G \models \liminf h$, just as termination can

be proved by finding a termination measure and reasoning about single program transitions.

In this article we show that there is a progress measure, called the *Liminf measure*, that allows us to verify $G \models \liminf h$ by such local reasoning. A Liminf measure μ for (G, h) quantifies progress in terms of the nodes of a finite-path tree T, which we have equipped with a progress relation $\trianglerighteq_{\text{Li}}$, called the *Liminf relation*. Thus $\mu : V \to T$ is a mapping such that on any transition $(v, v') \in E$, from v to v', $\mu(v) \trianglerighteq_{\text{Li}} \mu(v')$ holds. The Liminf relation ensures in the limit that the Liminf condition is satisfied: if $\mu(v_0) \trianglerighteq_{\text{Li}} \mu(v_1) \trianglerighteq_{\text{Li}} \cdots$, then $v_0, v_1, \ldots \models \liminf h$. Therefore the resulting verification method is sound, and we also show that is complete. Thus $G \models \liminf h$ if and only if there is a Liminf progress measure for (G, h).

We show that this result is applicable to all liveness properties that are $\mathbf{\Sigma}^0_3$ in the Borel hierarchy. (The Borel hierarchy is described in Section 7.) Thus in principle, such properties can be verified by assertional reasoning that describes the value of μ at each point of the program as a function of the past computation. This verification method does not introduce nondeterminism into the program or the specification. In contrast, many of the earlier general approaches to verification are based on a nondeterministic representation of computations.

3 Previous Work

Liveness properties for programs were studied in terms of finite-state automata in [2, 3, 29, 36]. These automata, however, can only express properties at the $\mathbf{\Pi}^0_2$ level of the Borel hierarchy. This limitation also holds for Propositional Temporal Logic and the *Extended Temporal Logic* of [40], which both can be translated into finite-state Büchi automata.

Complete verification methods for termination under fairness were given in [13, 14, 16, 26, 27, 38]. These methods are based on *helpful directions* and the iterative use of proof rules applied to transformed programs. The methods of *explicit schedulers* developed in [5, 7, 10] involve transforming programs by adding auxiliary variables that are nondeterministically assigned values determining fair computations. For an extensive treatment of fairness based on helpful directions and explicit schedulers, see [14].

A general approach to verification with finite temporal formulas was proposed in [34]. It is based on establishing a direct correspondence between the program and the temporal formula by assigning a well-founded ordering to every program state. The verification conditions depend on inductively defined predicates on temporal formulas and are rather complicated.

Harel showed that by transformations on trees representing programs one can do program verification for all finite levels of the Borel hierarchy [17]. The approach in [10] is an adaptation of these ideas, based on modifying the program

with explicit schedulers, to the programming language of loops. Vardi's study of automata theoretic techniques [39] is based on combining automata, where nondeterministic specification automata specify incorrectness. This technique was also adopted by Manna and Pnueli [29].

Another approach to fairness is based on domain theory. Motivated by the relationship mentioned above between fairness and nondeterminism, Apt and Plotkin gave a semantic model for countable nondeterminism [6]. Kwiatkowska introduced a topological characterization of liveness properties within a non-interleaving trace theory augmented with a Scott topology [24].

The metric approaches based on Cauchy convergence in [9, 11, 32] also characterize infinite computations. To the author's knowledge, the Cauchy convergence concept has not formed the basis for a general theory of program verification—perhaps because in this context, Cauchy convergence does not appear to measure naturally the contribution of each program step.

Progress measures have also been formulated for nondeterministic specifications [22, 23], *Rabin conditions* [21, 23], and *WF conditions* [23]. Measures for WF conditions generalize the results in this article and are related to the Kleene-Suslin Theorem, but will not be further developed here. Measures for Rabin conditions give a practical method for proving termination under fairness [19, 23] and also have applications for the complementation of automata [20]. Rabin conditions, however, are rather cumbersome mathematical objects. The Liminf approach presented here gives a purer and more abstract theory of verification.

4 Definitions

A graph $G = (V, E)$ consists of a set V of *vertices* and a set $E \subseteq V \times V$ of directed *edges*. Let $W \subseteq V$. We say that $v' \in W$ is *reachable* from $v \in W$ by a path in W if there is a path $v_0, \ldots, v_n$ in G such that $v = v_0$, $v' = v_n$, and $v_i \in W$, $0 \leq i \leq n$. We write $v \rightarrow_W^* W$ for $v \in W$ if every $v' \in W$ is reachable by a path in W from v.

A *pointer tree* T is a prefix-closed countable subset of ω^*, where ω is the set of natural numbers. Each sequence $t = \langle t^1, \ldots, t^\ell \rangle$ in T represents a *node*, which has *children* $t \cdot \langle d \rangle \in T$. If t' is a prefix of $t \in T$, then t' is called an *ancestor* of t. If $t' = t \cdot \langle \theta \rangle \in T$ then t is a *parent* of t'. We visualize pointer trees as growing upwards; see Figure 2, where children are depicted from left to right in descending order. Any sequence of pointers $t^1, t^2, \ldots$ (finite or infinite) denotes a *path*— $\langle\rangle, \langle t^1 \rangle, \langle t^1, t^2 \rangle, \ldots$ (finite or infinite) in T, provided each $\langle t^1, \ldots, t^\ell \rangle \in T$. The *level* $|t|$ of a node $t = \langle t^1, \ldots, t^\ell \rangle$ is the number ℓ; the level of $\langle\rangle$ is 0. The *ancestor at level ℓ of* $t = \langle t^1, \ldots, t^n \rangle$ is $\langle t^1, \ldots, t^\ell \rangle$, where $\ell \leq n$, and it is denoted $t{\downarrow}\ell$. T is *finite-path* if there are no infinite paths in T. A *common ancestor* of nodes t and t' is a node that is an ancestor of both of t and of t'. The common ancestor at the highest level is the *highest common ancestor*.

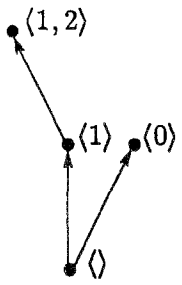

Figure 2: A pointer tree.

5 Liminf Relation and Measure

In this section we define the Liminf relation and show the main theorem. Let t, t' be nodes in a pointer tree. The Liminf relation $\trianglerighteq_{\mathrm{Li}}$ is defined as:

Definition 1 (Liminf Relation) $t \trianglerighteq_{\mathrm{Li}} t'$ if either t is an ancestor of t', t' is an ancestor of t, or there is a λ such that $|t|, |t'| > \lambda$, $t^{\lambda+1} > t'^{\lambda+1}$, and $t{\downarrow}\lambda = t'{\downarrow}\lambda$.

Intuitively, $t \trianglerighteq_{\mathrm{Li}} t'$ if one of t and t' is an ancestor of the other or if t' branches off to the right of t (assuming T is depicted as in Figure 2). Note that the Liminf relation is neither transitive nor antisymmetric; moreover, if the clause "t' is an ancestor of t" was omitted from the definition, we would have the Kleene-Brouwer ordering, which is well-founded when T is finite-path, see [35].

When T is finite-path, the Liminf relation guarantees that the liminf exists:

Lemma 1 (Liminf Relation Lemma) Let $t_0 \trianglerighteq_{\mathrm{Li}} t_1 \trianglerighteq_{\mathrm{Li}} \cdots$ be an infinite sequence of nodes in a finite-path pointer tree T. Then $\liminf_{i\to\infty} |t_i| < \omega$.

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

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 $\langle t_k^1, \ldots, t_k^\ell \rangle = \langle d^1, \ldots, d^\ell \rangle$.

Thus $\langle d^1, \ldots, d^\ell \rangle \in T$. Since this holds for all ℓ, $\langle d^1, d^1, \ldots \rangle$ is an infinite path in T. This contradicts that T is finite-path. □

Definition 2 A *Liminf measure* (μ, T) for (G, h) consists of a finite-path pointer tree T and a mapping $\mu : G \to T$ such that

- for all v, $|\mu(v)| = h(v)$; and
- h respects the edge relation, i.e. if $(v, v') \in E$, then $\mu(v) \trianglerighteq_{\text{lex}} \mu(v')$.

The main result of this article is:

Theorem 1 (Liminf Theorem) $G \models \liminf h$ iff (G, h) has a Liminf measure.

For the proof of this theorem we need a lemma:

Lemma 2 Let $G = (V, E)$ be a nonempty countable graph. There exists a (possibly infinite) partition $\mathcal{S} = \{W_0, W_1, \ldots\}$ of V and a mapping ι that to each set W of $\mathcal{S}$ associates a vertex $\iota(W) \in W$ such that

- $\iota(W) \to_W^* W$; and
- for all $(v, v') \in E$, if $v \in W_k$ and $v' \in W_{k'}$, then $k \geq k'$.

Proof For $w \in W \subseteq V$, let $\mathcal{R}(w, W) = \{w' \mid w \to_W^* w'\}$. Assume that V is ordered $v_0, v_1, \ldots$ Let $W_0 = \{v | v \in \mathcal{R}(w_0, V)\}$, where w_0 is v_0, the least vertex in V. If $V \setminus \bigcup_{j<k} W_j$ is not empty, W_k is defined as $\mathcal{R}(w_k, V \setminus \bigcup_{j<k} W_j)$, where w_k is the least vertex 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, $\bigcup_j W_j = V$; otherwise, there is a least vertex v_h in $V \setminus \bigcup_j W_j$. But if k is the least number such that $v_0, v_1, \ldots, v_{h-1} \in \bigcup_{j<k} W_j$, then $v_h = w_k$ and $v_h \in W_k$. This is a contradiction.

Let $\iota(W_k) = w_k$. By construction, $\iota(W_k) \to_{W_k} W_k$. Suppose for a contradiction that $(v, v') \in E$ with $v \in W_k$ and $v' \in W_{k'}$ for $k < k'$. Then $v' \in \mathcal{R}(w_k, V \setminus \bigcup_j W_{j<k})$. Thus $v' \in W_k$, which is a contradiction. □

Proof (Theorem 1) "⇐" If $v_0, v_1, \ldots$ is an infinite path in G, we get $\liminf h(v_i) = \liminf |\mu(v_i)| < \omega$ by the Liminf Relation Lemma.

"⇒" Given that $G \models \liminf h$, we show how to construct a Liminf measure (μ, T) for (G, h). We use the algorithm `AssignLiminf` in Figure 3. In addition to T, it calculates a label $W(t) \subseteq V$ for each node $t \neq \langle\rangle$ in T; this label denotes the set of vertices that will be mapped by μ to a node having t as an ancestor. $W(\langle\rangle)$ is defined to be V, because $\langle\rangle$ is an ancestor of every node in t. The transfinite algorithm `AssignLiminf` is initially applied at the root, i.e. initially $t = \langle\rangle$, and variable $T = \{\langle\rangle\}$.

The purpose of `AssignLiminf`(t) is to define the children $t \cdot \langle k \rangle$ of t together with their labels $W(t \cdot \langle k \rangle)$. First, vertices of height less than $|t|+1$ are filtered out. Second, the resulting set $\widehat{W}$ is partitioned using Lemma 2. Third, a child $t \cdot \langle k \rangle$

`AssignLiminf`(t):

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

Figure 3: Algorithm `AssignLiminf`.

with label W_k is created for each class W_k of the partition, and the algorithm is reapplied on $t \cdot \langle k \rangle$.

The pointer tree T is the value of the variable T obtained by the transfinite applications of `AssignLiminf`(t) starting with $\langle\rangle$. Note that the construction in Step 2, based on Lemma 2, associates to each label $W(t)$, $t \neq \langle\rangle$, a vertex $\iota(W(t))$ such that $\iota(W(t)) \rightarrow^*_{W(t)} W(t)$.

Claim 1

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

(2) For any $t \cdot \langle d \rangle \in T$, $W(t) \supseteq W(t \cdot \langle d \rangle)$. For each $v \in V$, there is a unique path $\langle d^1, \ldots, d^{h(v)} \rangle$ of T containing all nodes whose labels contain v.

Proof This follows by a straightforward induction. □

Now using (2) of Claim 1, we define $\mu(v) = \langle d^1, \ldots, d^{h(v)} \rangle$. We prove that μ is a Liminf measure.

First, it is obvious that $|\mu(v)| = h(v)$. Second, assume that $(v, v') \in E$. Let $\mu(v) = \langle d^1, \ldots, d^{h(v)} \rangle$, $\mu(v') = \langle e^1, \ldots, e^{h(v')} \rangle$ and $m = \min\{h(v), h(v')\}$. If $d^\ell = e^\ell$ for all $\ell \leq m$, then $\mu(v)$ is a prefix of $\mu(v')$ or $\mu(v')$ is a prefix of $\mu(v)$; thus $\mu(v) \trianglerighteq_{\mathrm{Li}} \mu(v')$. Otherwise, assume that $\lambda \leq m$ is such that $d^\lambda \neq e^\lambda$ and $d^\ell = e^\ell$ for all $\ell < \lambda$. Let $t = \langle d^1, \ldots, d^{\lambda-1} \rangle = \langle e^1, \ldots, e^{\lambda-1} \rangle$. Then `AssignLiminf`$(t)$ is applied and $v, v' \in W(t)$. This invocation defines $W(t \cdot d^\lambda)$ and $W(t \cdot e^\lambda)$, which are disjoint subsets of $W(t)$ by the assumption that $d^\lambda \neq e^\lambda$. Hence, $d^\lambda > e^\lambda$ by Lemma 2. It follows that $\mu(v) \trianglerighteq_{\mathrm{Li}} \mu(v')$.

Claim 2 There is no infinite path in T.

Proof Suppose for a contradiction that $\langle d^1, d^2, \ldots \rangle$ is an infinite path. Then $W(\langle d^1 \rangle) \supseteq W(\langle d^1, d^2 \rangle) \supseteq \cdots$, thus for any $t = \langle d^1, \ldots, d^k \rangle \neq \langle\rangle$, $\iota(W(t)) \rightarrow^*_{W(t)} W(t \cdot \langle d^{k+1} \rangle)$, because $\iota(W(t)) \rightarrow^*_{W(t)} W(t)$ by Lemma 2. In particular, $\iota(W(t))$

$\rightarrow^*_{W(t)} \iota(W(t \cdot \langle d^{k+1} \rangle)))$. We conclude that there exists an infinite path $v_0, v_1, \ldots$ in G and a monotonically increasing function $i(k)$ such that $v_{i(k)} = \iota(W(\langle d^1, \ldots, d^k \rangle))$ for $k \geq 0$, and for all $n \geq i(k)$, $h(v_n) \geq k$ by (1) of Claim 1. But then $v_0, v_1, \ldots$ does not satisfy the Liminf condition. This is a contradiction. □

6 Applications to Fairness

We now apply the Liminf Theorem to give a precise characterization of what it means for a program to terminate under very general fairness constraints. A fairness constraint is most easily described using Rabin's notion of acceptance for automata [33]. Let V be a countable set of states. A *Rabin pair* (R, I) on V consists of a set $R \subseteq V$ of *reconfirming* states and a set $I \subseteq V$ of *invalidating* states. We say that an infinite sequence $v_0, v_1, \ldots$ of states *satisfies* (R, I) and write $v_0, v_1, \ldots \models (R, I)$ if there are infinitely many k such that $v_k \in R$ and only finitely many k such that $v_k \in I$. Applied to fairness, R describes the set of states where an enabling condition is fulfilled, and I describes the set of states where an action is taken. Then a computation $v_0, v_1, \ldots$ is *unfair* with respect to (R, I) if $v_0, v_1, \ldots \models (R, I)$, i.e. if the enabling condition is fulfilled infinitely often and the action is only taken finitely often.

A *Rabin condition* (or *Rabin assignment*) $\mathcal{C}$ is a set $\{(R_\chi, I_\chi) \mid \chi \in X\}$ of Rabin pairs. Here X is a countable set of indices. An infinite sequence $v_0, v_1, \ldots$ *satisfies* $\mathcal{C}$, and we write $v_0, v_1, \ldots \models \mathcal{C}$, if for some χ, $v_0, v_1, \ldots \models (R_\chi, I_\chi)$. A Rabin condition expresses a fairness constraint by describing countably many ways a program can execute unfairly. We say that a graph $G = (V, E)$ satisfies a Rabin condition $\mathcal{C}$ on V and write $G \models \mathcal{C}$ if every infinite path $v_0, v_1, \ldots$ in G satisfies $\mathcal{C}$, i.e. if every infinite computation is unfair. In that case G represents a program that *fairly terminates* with respect to $\mathcal{C}$.

Now we assume that a program is represented as a directed tree $G = (V, E, v^0)$ with root v^0. (Any program can be represented as a tree if a variable that records the previous program states is introduced; this does not modify the behavior of the program, nor is nondeterminism introduced.) We show how to translate a Rabin condition $\mathcal{C}$ on V into a height assignment h.

Lemma 3 Let $\mathcal{C} = \{(R_i, I_i) \mid i \in \omega\}$ be a countable Rabin condition on a directed tree $G = (V, E, v^0)$. There is a function $f_\mathcal{C} : V \to \omega$ such that for any infinite path $v_0, v_1, \ldots$ in G:

$$v_0, v_1, \ldots \models \mathcal{C} \text{ iff } v_0, v_1, \ldots \models \liminf f_\mathcal{C}.$$

Proof Define $f_i(u) = |u| - @R_i(u) + @I_i(u)$, where $|u|$ is the level of vertex u in the tree G and $@M(u)$, $M \in \{R_i, I_i\}$, is the level of the closest ancestor u' of u for which $u' \in M$:

$$@M(u) = \begin{cases} |u'| & \text{if } u' \text{ is the closest ancestor of } u \text{ such that } u' \in M \\ 0 & \text{if there is no such ancestor} \end{cases}$$

Then it is easy to see that $\liminf_{k\to\infty} f_i(v_k) < \omega$ iff $v_0, v_1, \ldots \models (R_i, I_i)$. Now let $f_C(u) = \min_{i<\omega} \rho(i, f_i(u))$, where $\rho : \omega \times \omega \to \omega$ is a bijection.

It remains to prove that $\liminf_{k\to\infty} f_C(v_k) < \omega$ iff there exists an i such that $\liminf_{k\to\infty} f_i(v_k) < \omega$.

If $\liminf_{k\to\infty} f_C(v_k) < \omega$, then there exists an h such that for infinitely many k, $f_C(v_k) = h$. Thus, there is an i such that $\rho(i, f_i(v_k)) = h$ for infinitely many k, whence $\liminf_{k\to\infty} f_i(v_k) < \omega$.

Conversely, if $\liminf_{k\to\infty} f_i(v_k) = j < \omega$ for some i and j, then for infinitely many k, $f_C(v_k) \leq \rho(i,j)$, hence $\liminf_{k\to\infty} f_C(v_k) \leq \rho(i,j) < \omega$. □

By Lemma 3 and Theorem 1, it follows that termination under extreme fairness can be measured under assumption that the computation graph is a tree.

7 Applications to Descriptive Set Theory

In this section we establish a connection to descriptive set theory, which is the study of sets in terms of their descriptions. Using the main result, we show that comparisons of sets at the third level of the Borel hierarchy can take place in terms of descriptions based on finite approximations.

7.1 The Baire space

The *Baire space* ω^ω is a topological space whose points consist of infinite sequences $\alpha = \alpha_0, \alpha_1, \ldots$ of natural numbers; see [31]. The *basic open sets* are $\{u \cdot \alpha | \alpha \in \omega^\omega\}$ for $u \in \omega^*$. The class of *open sets*, which are arbitrary unions of basic open sets, is denoted $\mathbf{\Sigma}^0_1$ (also called $\mathbf{G}$). The class of *closed sets*, which are the complements of open sets, is denoted $\mathbf{\Pi}^0_1$ (also called $\mathbf{F}$). (With this topology, a basic open set is also a closed set, and $\{u \cdot \alpha | \alpha \in \omega^\omega\}$, $u \in \omega^*$, constitute a *clopen basis*.)

The classes $\mathbf{\Sigma}^0_1$ and $\mathbf{\Pi}^0_1$ are at the first level of the *Borel hierarchy*. The class $\mathbf{\Sigma}^0_2$ (or $\mathbf{F}_\sigma$) consists of the countable unions of closed sets (whereas a countable intersection of $\mathbf{\Pi}^0_1$ sets is still a $\mathbf{\Pi}^0_1$ set). Similarly, the class $\mathbf{\Pi}^0_2$ (or $\mathbf{G}_\delta$) consists of the countable intersections of open sets. The class $\mathbf{\Sigma}^0_3$ (or $\mathbf{G}_{\delta\sigma}$) consists of the countable unions of $\mathbf{\Pi}^0_2$ sets, and the class $\mathbf{\Pi}^0_3$ (or $\mathbf{F}_{\sigma\delta}$) consists of the countable intersections of $\mathbf{\Sigma}^0_2$ sets.

Proceeding in this way, one obtains the *finite levels* $\mathbf{\Sigma}^0_i$ and $\mathbf{\Pi}^0_i$, $i \in \omega$, of the Borel hierarchy. The entire Borel hierarchy is the transfinite closure under countable unions and intersections of the basic open sets.

In this section we show how to determine $P \subseteq S$ from finite computations, where P is $\mathbf{\Pi}^0_3$ and S is $\mathbf{\Sigma}^0_3$.

7.2 Approximation Functions and Limit Operators

A limit operator is a way of defining certain sets in the Baire space as limits of finite sequences described by *approximation functions*. For example, let $f : \omega^* \to \{0,1\}$ be a binary approximation function and define the limit operator $\lim_{\Pi_1^0}$ by:

$$\lim_{\Pi_1^0} f = \{\alpha \in \omega^\omega \mid \forall u \propto \alpha : f(u) = 1\}$$

where $u \propto \alpha$ means that u is a finite prefix of α. Thus $\lim_{\Pi_1^0} f$ is the set of infinite sequences α such that the value of f is 1 on all finite prefixes of α. It can be shown that any closed set is equal to $\lim_{\Pi_1^0} f$ for some binary approximation function f; moreover for any f, $\lim_{\Pi_1^0} f$ is a closed set. This characterization is essentially the same as a well-known equivalence of closed sets and languages of deterministic automata [23].

The classes $\mathbf{\Sigma}_1^0$, $\mathbf{\Pi}_2^0$, and $\mathbf{\Sigma}_2^0$ can also be described by limit operators. Define the limit operators $\lim_{\Sigma_1^0}$, $\lim_{\Pi_2^0}$, and $\lim_{\Sigma_2^0}$ according to:

$$\begin{aligned}
\alpha \in \lim_{\Sigma_1^0} f &\text{ iff } \exists u \propto \alpha : f(u) = 1 \\
\alpha \in \lim_{\Pi_2^0} f &\text{ iff } \exists^\infty u \propto \alpha : f(u) = 1 \\
\alpha \in \lim_{\Sigma_2^0} f &\text{ iff } \forall^\infty u \propto \alpha : f(u) = 1
\end{aligned}$$

Here $\exists^\infty$ and $\forall^\infty$ mean "for infinitely many" and "for all but finitely many," respectively. It is not hard to prove that these limit operators exactly define the classes $\mathbf{\Sigma}_1^0$, $\mathbf{\Pi}_2^0$, and $\mathbf{\Sigma}_2^0$, respectively; see [8, 25] for automata-theoretic proofs.

Using approximation functions of type $f : \omega^* \to \omega$, we introduce limit operators $\lim_{\Pi_3^0}$ and $\lim_{\Sigma_3^0}$:

$$\begin{aligned}
\alpha \in \lim_{\Pi_3^0} f &\text{ iff } \forall \ell : \forall^\infty u \propto \alpha : f(u) \neq \ell \text{ iff } \liminf_{u \to \alpha} f(u) = \omega \\
\alpha \in \lim_{\Sigma_3^0} f &\text{ iff } \exists \ell : \exists^\infty u \propto \alpha : f(u) = \ell \text{ iff } \liminf_{u \to \alpha} f(u) < \omega
\end{aligned}$$

where $u \to \alpha$ denotes that u takes the values $\langle\rangle, \langle\alpha_0\rangle, \langle\alpha_0, \alpha_1\rangle, \ldots$ for $\alpha = \alpha_0, \alpha_1, \ldots$

Proposition 1 Limit operators $\lim_{\Pi_3^0}$ and $\lim_{\Sigma_3^0}$ define the classes $\mathbf{\Pi}_3^0$ and $\mathbf{\Sigma}_3^0$; i.e. $S \in \mathbf{\Pi}_3^0$ if and only if there is an f such that $S = \lim_{\Pi_3^0} f$, and $S \in \mathbf{\Sigma}_3^0$ if and only if there is an f such that $S = \lim_{\Sigma_3^0} f$.

Proof We only give a proof for $\mathbf{\Sigma}_3^0$. The proof for $\mathbf{\Pi}_3^0$ is obtained by duality. A set $\lim_{\Sigma_3^0} f$ is a $\mathbf{\Sigma}_3^0$ set, because it can be written as a countable union of $\mathbf{\Pi}_2^0$ sets:

$$\bigcup_i \lim_{\Pi_2^0} f_i,$$

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 S is a union $\cup_i \lim_{\mathbf{\Pi}^0_2} f_i$ of $\mathbf{\Pi}^0_2$ sets, then define

$$f(u) = \begin{cases} i & \text{if } i \text{ is minimal such that } f_i(u) = 1 \\ |u| & \text{if no such } i \text{ exists} \end{cases}$$

It follows that $\alpha \in \lim_{\mathbf{\Sigma}^0_3} f$ iff $\exists i : \exists^\infty u : f(u) = i$ iff $\exists i : \exists^\infty u : f_i(u) = 1$ iff $\exists i : \alpha \in \lim_{\mathbf{\Pi}^0_2} f_i$. □

7.3 Liminf Convergence Measure

Consider the verification problem of determining whether $P \subseteq S$, where $P = \lim_{\mathbf{\Pi}^0_3} f_P$ and $S = \lim_{\mathbf{\Sigma}^0_3} f_S$, in terms of finite computations. We need a variation on Liminf measures that takes both f_P and f_S into account:

Definition 3 A *Liminf convergence measure* (μ, T) for (f_P, f_S) is a Liminf measure for $(G, u \mapsto \min\{f_P(u), f_S(u)\})$, where $G = (\omega^*, \{(u, u\cdot\langle n\rangle) \mid u \in \omega^*, n \in \omega\})$ is the infinite tree of finite computations.

A convergence measure guarantees that whenever a finite computation is lengthened, it converges towards being in S if it converges towards being in P. Formally we have the following characterization of the $P \subseteq S$ question:

Corollary 1 (of Theorem 1) For any $\mathbf{\Pi}^0_3$ set $P = \lim_{\mathbf{\Pi}^0_3} f_P$ and any $\mathbf{\Sigma}^0_3$ set $S = \lim_{\mathbf{\Sigma}^0_3} f_S$,

$$P \subseteq S \text{ iff } (f_P, f_S) \text{ has a Liminf convergence measure.}$$

Proof "$\Leftarrow$" Assume that (f_P, f_S) has measure (T, μ). Let $h(u) = \min\{f_P(u), f_S(u)\}$. Let $\alpha = \alpha_0, \alpha_1, \ldots \in P$. Then $\liminf_{u\to\alpha} f_P(u) = \omega$. Since $|\mu(v)| = h(v)$ and $\langle\rangle \to \langle\alpha_0\rangle \to \langle\alpha_0, \alpha_1\rangle \to \cdots$, we get a sequence $\mu(\langle\rangle) \trianglerighteq_{\mathrm{LI}} \mu(\langle\alpha_0\rangle) \trianglerighteq_{\mathrm{LI}} \mu(\langle\alpha_0, \alpha_1\rangle) \trianglerighteq_{\mathrm{LI}} \cdots$, so by Lemma 1, $\liminf_{u\to\alpha} \min\{f_P(u), f_S(u)\} < \omega$. We conclude $\liminf_{u\to\alpha} f_S(u) < \omega$, i.e. $\alpha \in S$.

"$\Rightarrow$" If $P \subseteq S$, then for all $\alpha \in \omega^\omega$, $\liminf_{u\to\alpha} \min\{f_P(u), f_S(u)\} < \omega$. Use Theorem 1 to obtain T and μ. □

Acknowledgements

Thanks to Dexter Kozen for suggesting substantial simplifications and for pointing out the similarity of the Liminf relation to the Kleene-Brouwer ordering. Also thanks to an anonymous referee for helpful comments.

References

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

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

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

[4] K.R. Apt. Ten years of Hoare's logic: A survey—part I. *ACM Transactions on Programming Languages*, 3(4):431–483, 1981.

[5] K.R. Apt and E.-R. Olderog. Proof rules and transformations dealing with fairness. *Science of Computer Programming*, 3:65–100, 1983.

[6] K.R. Apt and G.D. Plotkin. Countable nondeterminism and random assignment. *JACM*, 33(4):724–767, 1986.

[7] K.R. Apt, A. Pnueli, and J. Stavi. Fair termination revisited with delay. *Theoretical Computer Science*, 33:65–84, 1984.

[8] 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.

[9] A. Arnold and Nivat M. Metric interpretations of infinite trees and semantics of non deterministic recursive programs. *Theoretical Computer Science*, 11:181–205, 1980.

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

[11] P. Degano and U. Montanari. Liveness properties as convergence measures in metric spaces. In *Proc. 16th ACM Symposium on Theory of Computing*, pages 31–38, 1984.

[12] R. Floyd. Assigning meaning to programs. In *Mathematical Aspects of Computer Science XIX*, pages 19–32. American Mathematical Society, 1967.

[13] N. Francez and D. Kozen. Generalized fair termination. In *Proc. 11th POPL, Salt Lake City*. ACM, January 1984.

[14] Nissim Francez. *Fairness*. Springer-Verlag, 1986.

[15] David Gries. *The Science Of Programming*. Springer-Verlag, 1981.

[16] O. Grumberg, N. Francez, J.A. Makowsky, and W.P. de Roever. A proof rule for fair termination of guarded commands. *Information and Control*, 66(1/2):83–102, 1985.

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

[18] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, October 1969.

[19] N. Klarlund. Progress measures and stack assertions for fair termination. Submitted, 1991.

[20] N. Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *Proc. Foundations of Computer Science.* IEEE, 1991.

[21] N. Klarlund and D. Kozen. Rabin measures and their applications to fairness and automata theory. In *Proc. Sixth Symp. on Logic in Computer Science.* IEEE, 1991.

[22] N. Klarlund and F.B. Schneider. Proving nondeterministically specified safety properties using progress measures. To appear in *Information and Computation*, 1992.

[23] Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations.* PhD thesis, TR-1153, Cornell University, August 1990.

[24] M. Kwiatkowska. On topological characterization of behavioural properties. Technical report, Department of Computing Studies, University of Leicester, 1990.

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

[26] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: the ethics of concurrent termination. In *Proc. 8th ICALP.* LNCS 115, Springer-Verlag, 1981.

[27] 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.

[28] Z. Manna and A. Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Science of Computer Programming*, 4(3):257–290, 1984.

[29] 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.

[30] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. Ninth Symp. on the Principles of Distributed Computing*, pages 377–408. ACM, 1990.

[31] Yiannis N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Log. and the Found. of Math.* North-Holland, 1980.

[32] L. Priese and D. Nolte. Strong fairness, metric spaces, and logical complexity. Technical Report 65, Universität-Gesamthochschule-Paderborn, Fachbereich Mathematik-Informatik, 1990.

[33] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *American Mathematical Society*, 141:1–35, 1969.

[34] R. Rinat, N. Francez, and O. Grumberg. Infinite trees, markings and well-foundedness. *Information and Computation*, 79:131–154, 1988.

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

[36] A.P. Sistla. On using automata in the verification of concurrent programs. Technical report, Computer and Intelligent Systems Laboratory, GTE Laboratories Inc, 1987.

[37] 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.

[38] F.A. Stomp, W.P. de Roever, and R.T. Gerth. The μ-calculus as an assertion-language for fairness arguments. *Information and Computation*, 82:278–322, 1989.

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

[40] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.