The Limit View of Infinite Computations*

Nils Klarlund**

BRICS†, Department of Computer Science,
University of Aarhus
Ny Munkegade, DK-8000 Århus, Denmark
klarlund@daimi.aau.dk

Abstract. We show how to view computations involving very general liveness properties as limits of finite approximations. This computational model does not require introduction of infinite nondeterminism as with most traditional approaches. Our results allow us directly to relate finite computations in order to infer properties about infinite computations. Thus we are able to provide a mathematical understanding of what simulations and bisimulations are when liveness is involved.
In addition, we establish links between verification theory and classical results in descriptive set theory. Our result on simulations is the essential contents of the Kleene-Suslin Theorem, and our result on bisimulation expresses Martin's Theorem about the determinacy of Borel games.

1 Introduction

It is generally believed that to model general liveness properties of concurrent systems, such as those expressed by infinitary temporal logics, we must use machines with infinite (countable) nondeterminism. Such models arise for example in program verification involving fairness, where transformations of programs induce nondeterminism.

But it is disturbing that countable nondeterminism, for which no physical implementation seems to exist, is introduced in our model of computation. In contrast, any conventional Turing machine can be implemented and we can observe its finite runs, although not all of them, of course, due to physical constraints. But how would a physical device carry out a nondeterministic choice among uncountably many possibilities at each computation step? Instead, it seems more reasonable to let the machine compute finite information about progress that somehow gives rise to an acceptance condition on infinite computations.

---

* **This article is a revised and extended version of an earlier technical report ("Convergence Measures," TR90-1106, Cornell University), which was extracted from the author's Ph.D. thesis. Due to space limitations, all proofs have been omitted in this article.**

** Partially supported by an Alice & Richard Netter Scholarship of the Thanks to Scandinavia Foundation, Inc. and NSF grant CCR 88-06979.

† Basic Research in Computer Science, Centre of the Danish National Research Foundation.

Another foundational problem we would like to address is the lack of general notions of simulation and bisimulation for programs that incorporate liveness conditions. Since simulations are local equivalences, we also here need a better understanding of progress of finite computations towards defining infinite ones.

In this paper, we introduce a *limit concept* that allows deterministic machines to calculate *progress approximations* so that analytic or coanalytic sets[5]of computations are defined. Thus nondeterminism is not inherent to models of computations involving even very general liveness conditions, even those that are expressed by infinitary temporal logics.

Our concept is a natural generalization of Büchi or Rabin conditions, which define only sets very low in the Borel[6] hierarchy of properties. Our progress approximations generalize Büchi automata, where states are designated as accepting or non-accepting and the limit condition is that infinitely many accepting states are encountered.

Our main goal is to show that reasoning about the infinite behavior of our machines can be carried out directly in terms of progress approximations without transformations. Specifically, we turn our attention to two fundamental problems for programs with liveness conditions:

- finding a progress concept for showing that one program implements another program so that each step contributing to a live computation of the first program is mapped to a corresponding step of the second program; and
- finding a progress concept for showing that two programs can simulate each other so that each step of one corresponds to a step of the other equivalent with respect to making or not making progress towards a live computation.

Our results for these two problems are essentially the contents of the two perhaps most celebrated results of descriptive set theory: the Kleene-Suslin Theorem and Martin's Theorem about the determinacy of Borel games, respectively.

Previous Work

For certain kinds of specifications, such as those involving bounded nondeterminism or fairness, dozens of verification methods have been suggested;

[5] The notion of *analytic* set can be defined in many ways. For example, M is analytic if there is a nondeterministic automaton (with a countable state space) such that M is the set of infinite sequences allowing an infinite run, see [28]. The class of analytic sets is denoted Σ^1_1. The dual of an analytic set is said to be *coanalytic* or Π^1_1.

[6] The *Borel hierarchy* is the least class of sets containing the class Σ^0_1 of open sets and closed under countable intersection and union. For example, the Borel hierarchy contains the class Π^0_2 of sets that are countable intersections of open sets. Every property defined by a Büchi automaton, including usual fairness conditions, is a finite Boolean combination of Π^0_2 sets (and so is at the third level of the Borel hierarchy), see [27]. Every Borel set is analytic and coanalytic. Vice versa, the Kleene-Suslin Theorem states that any set that is both analytic and coanalytic is also Borel.

see [1, 2, 3, 4, 6, 7, 16, 17, 18, 24]. Yet the general problem has, to the author's knowledge, been addressed only in relatively few articles [5, 9, 28]. The earlier proposals solve the verification problem by transformations that introduce infinite nondeterminism.

The most general such approach is that of Vardi [28]. Vardi uses a computational model corresponding to nondeterministic automata with infinite conjunctive branching for defining specifications. The apparent physical unrealizability of this concept motivated the limit view given in the present paper. Vardi's method can easily be reformulated as progress measures [29], i.e. as mappings from the states of the implementation. These progress measures, however, do not imply the ones presented here, since we use a different computational model. Together with the Boundedness Theorem for Π^1_1 sets, Vardi's results amount to the Kleene-Suslin Theorem, although this was not noted in [28].

A few methods have appeared that more directly quantify progress [3, 13, 11, 22], but these methods only apply to sets at low levels of the Borel hierarchy, namely finite Boolean combinations of Π^0_2 sets.

In [10], a simple progress measure based on a condition, called the *Liminf condition*, was proposed. This condition, however, can be used only to reason about specifications that are Σ^0_3, i.e. at the third level of the Borel hierarchy. Other approaches to viewing computations as limits are based on metrics of mathematical analysis [21]. These approaches also deal with sets at the third level.

Recursion-theoretic aspects of relating automata with different kinds of acceptance conditions have been studied in [26].

In this paper we report on the most general measure proposed in [15]. In addition, we introduce here the new concept of progress bisimulation.

2 Overview

Our results are based on an abstract, graph-theoretic formulation of the verification problem. We represent the transitions of a program as a directed, countable graph $G = (V, E)$, where vertices V and edges E correspond to program states and transitions. Then the infinite paths in G correspond to all possible infinite computations.

To define live computations, we introduce *progress approximations* on V that assign a finite amount of information to each vertex. In turns out that if we let this information be a labeled tree, then very general properties can be expressed. Thus a progress approximation τ associates a finite tree $\tau(v)$ to each vertex v. A computation $v_0 v_1 \cdots$ defines an infinite sequence $\tau(v_0), \tau(v_1), \ldots$ of progress approximations and a *limit tree* $\lim \tau(v_i)$ that consists of the nodes that from a point on occur in every progress approximation. The computation is *live* if the limit tree has only finite paths, i.e. if it is *well-founded* (it may still be infinite). We call this condition the *well-foundedness condition* of τ and abbreviate it WF τ.

The WF condition is extraordinarily powerful. We prove that the sets spec-

ified by WF conditions constitute the class Π^1_1 of coanalytic sets. This class includes all Borel sets as we show using progress approximations. In fact, we show that automata combined with temporal logics with infinite conjunctions and disjunctions express the class of Borel sets and can be coded as WF conditions.

The dual of the WF condition is the condition that requires the limit tree to contain some infinite path. This condition is called the *non-well-foundedness condition* and denoted $\neg$WF. The sets specified by $\neg$WF conditions constitute the class Σ^1_1 of analytic sets.

In order to relate two programs with liveness conditions, we first study a simpler problem. We may view WF τ as a *specification* that every computation of G is live. Thus we say that G *satisfies* WF τ if every infinite path $v_0 v_1 \cdots$ of G satisfies WF τ. Note that this property seems to call for considering uncountably many infinite computations. Our first result is to show that G satisfies WF τ can be established by local reasoning about vertices and edges.

2.1 Progress Measures

For proving the property of program termination, we usually resort to mapping program states to some values, and we then verify that the value decreases with every transition. These values quantify progress toward the property of termination. Similarly for a property specified by a WF condition, we seek a relation on some set of progress values that the states with their progress approximations can be mapped to. The relation must ensure that the limit tree is well-founded.

To do this, we use tree embeddings as the set of progress values. We fix a well-founded tree T and a mapping μ such that $\mu(v)$ specifies an embedding of $\tau(v)$ in T. We then define the *WF progress relation* $\trianglerighteq_{WF}$ on tree embeddings. Intuitively, it states that embedded nodes move forward in T according to a predefined ordering. If in addition μ satisfies the verification condition

(VC) for any transition from v to v', it is the case that $\mu(v) \trianglerighteq_{WF} \mu(v')$,

then μ is a *WF progress measure*. Our first result is:

Graph Result

All infinite paths in G satisfy WF τ
if and only if
there is a progress measure μ for G and τ.

Thus the question of verifying that *all* infinite computations satisfy the specification is equivalent to finding *some* mapping that is a WF progress measure. In other words, the existence of a progress measure means that each step of a program contributes in a precise mathematical sense to bringing the computation closer to the specification.

2.2 Progress Simulations

To formulate our results on progress simulations, we turn to a generally accepted model of infinite computations. There is an alphabet Σ of letters representing

actions, and a *program* $\mathcal{P}$ is a nondeterministic transition system or automaton over Σ. The *computations* or runs over an infinite word $a_0 a_1 \cdots$ are the sequences of states, beginning with an initial state, that may occur when the word is processed according to the transition relation. A word is *recognized* by $\mathcal{P}$ if it allows a run. The set of words recognized by $\mathcal{P}$ is called the *language* of $\mathcal{P}$ and denoted $L(\mathcal{P})$. (Note that in this model we have abstracted away the details of the machine structure and technical complications such as stuttering.)

Thanks to the countable nondeterminism present in such programs, they define the class of Σ_1^1 of analytic sets[7]. Now given two programs $\mathcal{P}$ and $\mathcal{Q}$, called the *implementation* and *specification*, we say that $\mathcal{P}$ *implements* $\mathcal{Q}$ if every word recognized by $\mathcal{P}$ is also recognized by $\mathcal{Q}$, i.e. if $L(\mathcal{P}) \subseteq L(\mathcal{Q})$. The *verification problem* is to establish $L(\mathcal{P}) \subseteq L(\mathcal{Q})$ by relating the states of the programs without reasoning directly about infinite computations.

It is well-known that if we can find a *simulation*, also known as a homomorphism or refinement map, from the reachable states of $\mathcal{P}$ to the states of $\mathcal{Q}$, then $L(\mathcal{P}) \subseteq L(\mathcal{Q})$. (This method is not complete, however, since $L(\mathcal{P}) \subseteq L(\mathcal{Q})$ might hold while no simulation exists [1, 14, 24].)

The preceding discussion has ignored liveness, including common concepts such as starvation and fairness. So assume that $\mathcal{P}$ also defines a set $Live_{\mathcal{P}}$ of state sequences said to be *live*. For example, the set $Live_{\mathcal{P}}$ may be specified by a formula in temporal logic or by a WF condition. The *live language* $L^{\circ}(\mathcal{P})$ of $\mathcal{P}$ is the set of words that allow a live computation. We say that $\mathcal{P}$ *satisfies* $\mathcal{Q}$ if the words allowing a live computation of $\mathcal{P}$ also allow a live computation of $\mathcal{Q}$, i.e. if $L^{\circ}(\mathcal{P}) \subseteq L^{\circ}(\mathcal{Q})$. The verification problem is now to show that $\mathcal{P}$ satisfies $\mathcal{Q}$ without considering infinite computations.

To simplify matters, we assume that a simulation already exists from $\mathcal{P}$ to $\mathcal{Q}$ and that the set $Live_{\mathcal{Q}}$ is expressed as a WF condition of a progress approximation $\tau_{\mathcal{Q}}$ on $\mathcal{Q}$'s state space. The set $Live_{\mathcal{P}}$ cannot be expressed as a WF condition if the verification problem is to be reduced to only a well-foundedness problem [25]. Thus we instead specify $Live_{\mathcal{P}}$ by a $\neg$WF condition of a progress approximation $\tau_{\mathcal{P}}$ on $\mathcal{P}$'s state space.

We show that there is an operation $merge^{\vee}$ that merge progress approximations so as to express the condition $Live_{\mathcal{P}} \Rightarrow Live_{\mathcal{Q}}$, i.e. $\neg\text{WF}\ \tau_{\mathcal{P}} \Rightarrow \text{WF}\ \tau_{\mathcal{Q}}$ or, equivalently, $\text{WF}\ \tau_{\mathcal{P}} \vee \text{WF}\ \tau_{\mathcal{Q}}$. Thus we formulate a *progress simulation* from $\mathcal{P}$ to $\mathcal{Q}$ as a simulation h together with a progress measure for the progress approximation $merge^{\vee}(\tau_P(p), \tau_S(h(p)))$, which is defined on $\mathcal{P}$'s reachable states.

We use the Graph Result to derive

[7] If in addition the program $\mathcal{P}$ can be *effectively* or *recursively represented* (that is, the transition relation can calculated by a Turing machine, which on input (s, a, s') halts with the answer to whether (s, a, s') is in the transition relation), then the language recognized is said to be *analytical*. The class of such languages is denoted Σ_1^1. In general, the effective class corresponding to a class denoted by a boldface letter is denoted by the lightface letter.

General Progress Simulation Theorem

If there is a simulation from $\mathcal{P}$ to $\mathcal{Q}$, then
$\mathcal{P}$ satisfies $\mathcal{Q}$
if and only if
there is a progress simulation from $\mathcal{P}$ to $\mathcal{Q}$.

The General Progress Simulation Theorem in particular solves the verification problem for programs and specifications that are expressed using formulas in infinitary temporal logic (under the assumption that a simulation exists).

The theorem has an effective version, which we call the *Finite Argument Theorem*. It shows that there is a uniform way of obtaining a progress simulation. Thus there is an algorithm that calculates a Turing machine for calculating a progress simulation given as input Turing machines defining $\mathcal{P}$, $\mathcal{Q}$, and a simulation h with $L^\circ(P) \subseteq L^\circ(S)$. This is not a decidability result, but an explicit reduction of the Π^1_1-complete problem of establishing $L^\circ(P) \subseteq L^\circ(S)$ to the classic Π^1_1-complete problem of whether a recursive tree is well-founded.

There is a strong connection to descriptive set theory. In fact, we show that the Finite Argument Theorem expresses the Kleene-Suslin Theorem as a statement about the feasibility of program verification.

2.3 Progress Bisimulations

Consider a program $\mathcal{P}$ with state space P and transition relation $\rightarrow_{\mathcal{P}}$ and a program $\mathcal{Q}$ with state space Q and transition relation $\rightarrow_{\mathcal{Q}}$.

The notion of bisimulation stipulates that the programs are equivalent if there is a relation $R \subseteq P \times Q$ containing the pair of initial states such that:

- if $R(p,q)$ and $p \xrightarrow{a}_{\mathcal{P}} p'$, then there is q' such that $q \xrightarrow{a}_{\mathcal{Q}} q'$ and $R(p',q')$, and
- vice versa, if $R(p,q)$ and $q \xrightarrow{a}_{\mathcal{Q}} q'$, then there is p' such that $p \xrightarrow{a}_{\mathcal{Q}} p'$ and $R(p',q')$.

This definition is central to the algebraic treatment of concurrency. The essential result is that the existence of the bisimulation relation is equivalent to the impossibility of observing a difference in behavior of the two systems with respect to ability of carrying out actions.

Assuming now that $\mathcal{P}$ and $\mathcal{Q}$ are bisimilar in this traditional sense, can we then compare them also regarding liveness? That is, we would like to relate program states also with respect to how close they are to satisfying the liveness conditions so as to formalize the intuition: for any transition of one program there is a transition for the other program which is equivalent with respect to progress or non-progress towards the liveness condition.

To get an understanding of what observing liveness means, we formulate the process as an infinite game between an *observer* and a *responder*. The game is the same as the one that characterizes bisimilarity, although the winning conditions are different: bisimilar programs $\mathcal{P}$ and $\mathcal{Q}$ are *live equivalent* if no observer can devise bisimilar computations of $\mathcal{P}$ and $\mathcal{Q}$ so that one is live and the other is not.

More precisely, the observer is allowed to pick actions and transitions according to the following rules in order to produce corresponding computations.

First, the observer chooses an action and a transition on this action for one of the programs from its initial state. Then the responder lets the other program make a corresponding transition on the same action from its initial state. The new pair of states must belong to the bisimulation relation (the responder can always find a new state by definition of bisimulation relation).

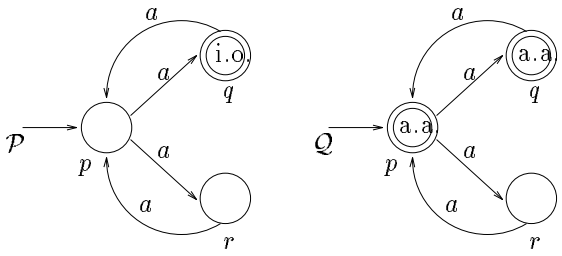
Next, the observer chooses a second action and a transition for one of the programs. This transition is again matched by the responder who lets the other program make a corresponding step.

This process continues *ad infinitum* and produces an infinite word and computations of $\mathcal{P}$ and $\mathcal{Q}$ over this word. If it is not the case that some observer can choose actions and transitions so that however the responder matches the observer's moves, a live and a non-live computation are produced, then $\mathcal{P}$ and $\mathcal{Q}$ are said to be *live equivalent*.

The way an observer chooses actions and transitions is called a *testing strategy*. Generally, a strategy is a function of all previous choices made by the other player. In case a choice is solely dependent on a current pair of states, the strategy is said to be *memoryless*. Similarly, the responder's answers are described by a *response strategy*, which is also a function of the previous choices. The response strategy is memoryless if it is dependent only of the current pair of states and the name and action of the process picked by the observer.

For usual bisimulation, it can be shown for both players that having a winning strategy is equivalent to having a winning memoryless strategy. Also, a bisimulation relation encodes a class of memoryless response strategies.

Unfortunately, the two programs below show informally that even for simple liveness conditions, it may happen that neither player has a winning memoryless strategy.



Here $\mathcal{P}$ and $\mathcal{Q}$ are the same program over a one letter alphabet except for the liveness condition: the program $\mathcal{P}$ accepts if the Büchi condition $\{q\}$ is satisfied, i.e. if the state q occurs infinitely often, and the program $\mathcal{Q}$ accepts if the states p and q occur almost always, i.e. if from some point on the state r is not encountered. It can be seen that neither the observer nor the responder has a winning memoryless strategy. In fact, the observer does have a winning strategy, namely "at p, pick the choice (q or r) that is the opposite of what the

responder last did," but this is not a memoryless strategy.[8] Thus $\mathcal{P}$ and $\mathcal{Q}$ are not live equivalent.

For general systems that are live equivalent, we shall show that a natural notion of progress bisimilarity can be formalized for their finite computations if the liveness conditions are Borel. If finite computations u and v of $\mathcal{P}$ and $\mathcal{Q}$ are progress bisimilar, we must express by a progress value ρ how close they are to being either both live or both non-live. Since we assumed that $Live_P$ is Borel, it is both analytic and coanalytic. Thus there is a pair $\tau_P = (\tau'_P, \tau''_P)$ of progress approximations such that $Live_P$ is the set of infinite state sequences that satisfy WF τ'_P and also the set of sequences that satisfy $\neg$WF τ''_P. For notational simplicity, we assume that these approximations are defined on finite computations. We then define an operation $merge^{\Leftrightarrow}$ on progress approximations such that $merge^{\Leftrightarrow}(\tau_P, \tau_Q)$ specifies the joint state sequences that are both live or both non-live.

A progress bisimulation $R^*(u, v, \rho)$ is now a relation that for some fixed well-founded T relates a finite computation u of $\mathcal{P}$, a finite computation v of $\mathcal{Q}$, and an embedding ρ of $merge^{\Leftrightarrow}(\tau_P(u), \tau_Q(v))$ in T such that:

- if $R^*(u, v, \rho)$ and $u \rightarrow_{\mathcal{P}} u'$, then there is v' and ρ' such that $v \rightarrow_{\mathcal{Q}} v'$, $R^*(u', v', \rho')$, and $\rho \trianglerighteq_{WF} \rho'$, and
- vice versa, if $R^*(u, v, \rho)$ and $v \rightarrow_{\mathcal{Q}} v'$, then there is u' and ρ' such that $u \rightarrow_{\mathcal{Q}} u'$, $R^*(u', v', \rho')$, and $\rho \trianglerighteq_{WF} \rho'$.

Our second main result is :

General Progress Bisimulation Theorem

If Borel programs $\mathcal{P}$ and Q are bisimilar, then
$\mathcal{P}$ and Q are live equivalent
if and only if
$\mathcal{P}$ and Q allow a progress bisimulation.

This result follows from a very deep result in descriptive set theory by Martin [19] that all infinite games with Borel winning conditions are determined, i.e. it is always the case that one player has a winning strategy. Since determinacy of games with arbitrary winning conditions contradicts the Axiom of Choice [20], the General Progress Bisimulation Theory is hard to generalize. In fact, the study of the Determinacy Axiom is an important part of mathematical logic.

[8] Note however that only bounded memory about the past is necessary to specify the observer's moves. This is a general phenomenon: as shown in [8], games based on Boolean combinations of Büchi conditions have bounded-memory strategies, also known as forgetful strategies. For Rabin conditions, which are special disjunctive normal forms, memoryless strategies do exist [12].

3 Definitions

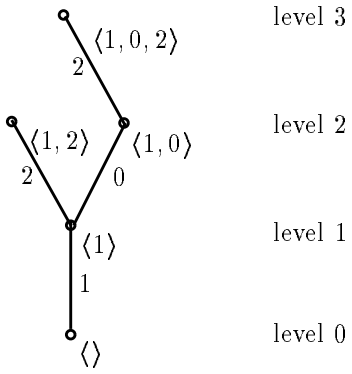
Programs and Simulations Assume a finite or countable alphabet Σ of *actions*. A program $\mathcal{P} = (P, \rightarrow, p^0, Live)$ over Σ consists of a *state space* P, a *transition relation* $\rightarrow \subseteq P \times \Sigma \times P$, an *initial state* p^0, and a liveness specification *Live*, which specifies a set of infinite state sequences. The program $\mathcal{P}$ is *deterministic* if for all p and a there is at most one p' such that $p \xrightarrow{a} p'$. A *computation* over an infinite word $a_0 a_1 \cdots$ is an infinite state sequence $p_0 p_1 \cdots$ such that $p_0 = p^0$ and $p_i \xrightarrow{a} p_{i+1}$, for all i. A computation is *live* if it satisfies *Live*. A *finite* computation $u \in P^*$ is a prefix of some infinite computation. The transition relation is extended to finite computations in the natural way: $u \xrightarrow{a} v$ if for some $\tilde{u}$, p, and p', $u = \tilde{u} \cdot p$, $v = u \cdot p'$, and $p \xrightarrow{a} p'$. The set of all words allowing some computation is denoted $L(\mathcal{P})$ and is called the *language* of $\mathcal{P}$. The subset of words in $L(\mathcal{P})$ that allow a live computation is denoted $L^{\omega}(\mathcal{P})$ and called the *live language* of $\mathcal{P}$.

A *simulation* $h : \mathcal{P} \hookrightarrow \mathcal{Q}$ is a partial function that maps the initial state of $\mathcal{P}$ to that of $\mathcal{Q}$ and respects the transition relation:

- $h(p^0) = s^0$, and
- $p \in \mathrm{dom}(h)$ and $p \xrightarrow{a}_{\mathcal{P}} p'$ implies $p' \in \mathrm{dom}(h)$ and $s \xrightarrow{a}_{\mathcal{Q}} s'$.

Note that if $\mathcal{P}$ and $\mathcal{Q}$ are deterministic with $L(\mathcal{P}) \subseteq L(\mathcal{Q})$, then a progress simulation exists (provided that $\mathcal{P}$ has no reachable state that has no successor). Also, h can be uniformly computed from effective representations of $\mathcal{P}$ and $\mathcal{Q}$.

Pointer Trees A *pointer tree* (or simply *tree*) T is a prefix-closed countable subset of ω^*, where ω is the set of natural numbers $0, 1, \ldots$ Each sequence $t = t^1 \cdots t^\ell$ in T represents a *node*, which has *children* $t \cdot d \in T$. Here $d \in \omega$ is the *pointer* to $t \cdot d$ from t. If t' is a prefix of $t \in T$, then t' is called an *ancestor* of t. We visualize pointer trees as growing upwards as in



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* — $\epsilon, t^1, t^1 \cdot t^2, \ldots$ (finite or infinite) in T, provided each $t^1 \cdots t^\ell \in T$. The *level* $|t|$ of a node $t = t^1 \cdots t^\ell$ is

the number ℓ; the level of ϵ is 0. T is *finite-path* or or *well-founded* if there are no infinite paths in T. This is also denoted WF T.

A well-founded tree T is *ν-rankable* if there is an assignment of ordinals to the nodes of T such that the root has rank ν and if a node t has rank γ then every child of t has rank less than γ.

4 WF Limit Representations

In this section we show how to represent analytic and coanalytic sets by limits. For a sequence $\tau_0, \tau_1, \ldots$ of pointer trees, we define $\lim_i \tau_i$ as:

$$t \in \lim_i \tau_i \text{ if and only if for almost all } i, t \in \tau_i.$$

It is not hard to see that $\lim_i \tau_i$ is a tree, which we call the *limit tree*. To characterize finite computations, we use a *progress approximation* τ that assigns a finite pointer tree $\tau(u)$ to each finite word $u \in \Sigma^*$. Thus we assume here that the underlying program is the transition system that has Σ^* as its state space and where the transition relation is defined so that the current state is the sequence of actions encountered. With this representation, the *live languages* $\lim_{\neg WF} \tau$ and $\lim_{WF} \tau$ are defined by: for a word $\alpha = a_0 a_1 \cdots$,

$$\alpha \in \lim_{\neg WF} \tau \text{ if and only if } \neg \mathrm{WF} \lim_{u \to \alpha} \tau(u), \text{ and}$$

$$\alpha \in \lim_{WF} \tau \text{ if and only if } \mathrm{WF} \lim_{u \to \alpha} \tau(u),$$

where $u \to \alpha$ denotes that u takes the values $\epsilon, a_0, a_0 a_1, \ldots$

The class Σ^1_1 of analytic sets and the class Π^1_1 of coanalytic sets can be described by limits:

Theorem 1. (Representation Theorem) *The limit operators* $\lim_{\neg WF}$ *and* $\lim_{WF}$ *define the classes* Σ^1_1 *and* Π^1_1, *i.e.* $S \in \Sigma^1_1 \Leftrightarrow \exists \tau : S = \lim_{\neg WF} \tau$ *and* $S \in \Pi^1_1 \Leftrightarrow \exists \tau : S = \lim_{WF} \tau$.

Proof The proof uses the classic representation involving projections of trees [20, p.77]. □

As with the usual representations (see [20]), we have:

Theorem 2. (Boundedness Theorem for Π^1_1 sets) *Let* $C = \lim_{WF} \tau$ *be a coanalytic set. If there is a countable ordinal* ν *such that for all* $\alpha \in C$, $\lim_{u \to \alpha} \tau(u)$ *is* ν*-rankable, then* C *is Borel.*

We postpone the proof of Theorem 2 to Section 7.

In the following, we say that an *analytic program* $\mathcal{P}$ is of the form $(P, \to, p^0, \neg \mathrm{WF}\ \tau)$ and that a *coanalytic program* is of the form $(P, \to, p^0, \mathrm{WF}\ \tau)$, where τ is a progress approximation that assigns a pointer tree to each state in P.

5 WF Relation and Measure

We use tree embeddings to measure progress of computations towards defining a finite-path tree in the limit. Let T be a fixed tree. An *embedding* of a tree τ in T is an injective mapping $\rho : \tau \to T$ such that $\rho(\epsilon) = \epsilon$ and for all $t \cdot d \in \tau$ there is d' with $\rho(t \cdot d) = \rho(t) \cdot d'$. Note that $|\rho(t)| = |t|$; in fact, ρ is just a structure-preserving relabeling of τ. Also note that $\mathrm{dom}(\rho)$ is the tree τ and that $\mathrm{rng}(\rho)$ is the image in T of τ.

We can now define the *WF relation*, which we denote by $\trianglerighteq_{WF}$:

Definition 3. (WF Relation) $\rho \trianglerighteq_{WF} \rho'$ if for all $s \in \mathrm{dom}(\rho) \cap \mathrm{dom}(\rho')$, $\rho(s) \succeq \rho'(s)$, where "$\succeq$" is defined by: $d^0 \cdots d^n \succeq e^0 \cdots e^n$ if either $d^0 \cdots d^n = e^0 \cdots e^n$ or there is a level $\lambda \leq n$ such that $d^\lambda > e^\lambda$ and for all $\ell < \lambda$, $d^\ell = e^\ell$.

Intuitively, $\rho \trianglerighteq_{WF} \rho'$ holds if for any node s in both $\mathrm{dom}(\rho)$ and $\mathrm{dom}(\rho')$, the image in T of s under ρ' is the same as or to the right of the image under ρ (assuming that pointer trees are depicted as explained earlier). Although $\trianglerighteq_{WF}$ is not a well-founded relation, it ensures well-foundedness in the limit provided T is well-founded.

Lemma 4. (WF Relation Lemma) *If WF T and $\rho_0 \trianglerighteq_{WF} \rho_1 \trianglerighteq_{WF} \cdots$, then WF $\lim_i dom(\rho_i)$.*

This lemma is an immediate consequence of:

Proposition 5. *Let T be a fixed tree and let $\rho_0 \trianglerighteq_{WF} \rho_1 \trianglerighteq_{WF} \cdots$ be an infinite $\trianglerighteq_{WF}$-related sequence of embeddings in T. Then there is an embedding ρ of $\lim_i dom(\rho_i)$ in $\lim_i rng(\rho_i)$.*

Hence if T is well-founded, $\trianglerighteq_{WF}$ measures progress of pointer trees towards defining a well-founded tree. To state this more forcefully, we need some definitions.

Definition 6. Let $G = (V, E)$ be a countable, directed graph and let τ be a progress approximation on V. We say that an infinite path $v_0 v_1 \cdots$ satisfies the *WF condition* of τ, and write $v_0 v_1 \cdots \models \mathrm{WF}\tau$, if $\mathrm{WF}\ \lim_i \tau(v_i)$. A graph G satisfies the WF condition of τ, and we write $G \models \mathrm{WF}\ \tau$, if every infinite path in G satisfies the WF condition.

Definition 7. A *WF progress measure* (μ, T) for (G, τ) is a finite-path tree T and a mapping $\mu : v \in V \to (\tau(v) \to T)$ such that

- $\mu(v)$ is an embedding of $\tau(v)$ in T; and
- μ respects the edge relation of G, i.e. $(u, v) \in E$ implies $\mu(u) \trianglerighteq_{WF} \mu(v)$.

Theorem 8. (Graph Result) *$G \models$ WF τ if and only if (G, τ) has a WF progress measure.*

Proof "$\Leftarrow$" This follows from the WF Relation Lemma.
"$\Rightarrow$" The proof consists of a transfinite construction of μ and T. □

6 Progress Simulations

In this section, we present the General Progress Simulation Theorem. Let $\mathcal{P} = (P, \rightarrow_{\mathcal{P}}, p^0, \neg \text{WF } \tau_{\mathcal{P}})$ be an analytic implementation and $\mathcal{Q} = (Q, \rightarrow_{\mathcal{Q}}, q^0, \text{WF } \tau_{\mathcal{Q}})$ a coanalytic specification. To prove that $L^\circ(\mathcal{P}) \subseteq L^\circ(\mathcal{Q})$, we need to combine the progress approximations.

Definition 9. Given finite trees τ and τ', the set $merge^\vee(\tau, \tau')$ consisting of nodes $d^0e^0 \cdots d^ne^n$ and $d^0e^0 \cdots e^{n-1}d^n$, where $n \geq -1$, $d^0 \cdots d^n \in \tau$, and $e^0 \cdots e^n \in \tau'$, is called the *or-merge* of τ and τ'.

It is not hard to see that $merge^\vee(\tau, \tau')$ is a tree. The or-merge has the following properties:

Proposition 10. *Let τ_i and τ'_i be infinite sequences of trees.*

(a) WF$\lim_i merge^\vee(\tau_i, \tau'_i)$ *if and only if WF*$\lim_i \tau_i$ *or WF*$\lim_i \tau'_i$.
(b) If $\lim merge^\vee(\tau_i, \tau'_i)$ *is ν-rankable and $\neg$ WF*$\lim_i \tau_i$, *then* $\lim_i \tau'_i$ *is ν-rankable.*

Given a simulation $h : \mathcal{P} \rightarrow \mathcal{Q}$, we measure progress towards $Live_{\mathcal{P}} \Rightarrow Live_{\mathcal{Q}}$ as follows.

Definition 11. A *progress simulation* (h, μ, T) from $\mathcal{P}$ to $\mathcal{Q}$ relative to h is a WF progress measure for $((V, E), p \mapsto merge^\vee(\tau_P(p), \tau_S(h(p))))$, where $V \subseteq P$ are the states of $\mathcal{P}$ reachable by some finite computation and $(p, p') \in E$ if and only if $p \xrightarrow{a} p'$ for some a.

Theorem 12. (Progress Simulation Theorem) *Assume we have analytic $\mathcal{P} = (P, \rightarrow_{\mathcal{P}}, p^0, \neg WF\, \tau_{\mathcal{P}})$, coanalytic $\mathcal{Q} = (Q, \rightarrow_{\mathcal{Q}}, q^0, WF\, \tau_{\mathcal{Q}})$, and simulation $h : \mathcal{P} \rightarrow \mathcal{Q}$. Then $L^\circ(\mathcal{P}) \subseteq L^\circ(\mathcal{Q})$ if and only if there is a progress simulation from $\mathcal{P}$ to $\mathcal{Q}$ relative to h.*

Proof The proof follows from the WF Relation Lemma, Proposition 10, and Theorem 8. □

6.1 Suslin's Theorem

Corollary 13. (Suslin's Theorem) *Let L be a set of infinite sequences over ω that is both analytic and coanalytic. Then L is Borel.*

6.2 Finite Argument Theorem

A progress simulation can be viewed as an argument for why a program satisfies a specification. We show that for effective descriptions of program, specification, and simulation, there is an effective description of the progress measure. More precisely, let a *WF semi-measure* (μ, T) be a WF progress measure except that there is no requirement that T be well-founded. Then there is a

total recursive function calculating an index of a WF semi-measure (μ, T) for $merge^{\vee}(\tau_P(p), \tau_S(h(p)))$ given indices for $\mathcal{P}$, $\mathcal{Q}$, and h; moreover, (h, μ, T) is a progress simulation, i.e. T is well-founded, if and only if $\mathcal{P}$ satisfies $\mathcal{Q}$.

Theorem 12′ ***(Finite Argument Theorem)*** *A progress simulation can be obtained uniformly from indices of $\mathcal{P}$, $\mathcal{Q}$, and h.*

Proof By analyzing the proof of Theorem 12 for computational contents, one can obtain an explicit algorithm for calculating μ and T. □

Intuitively, the Finite Argument Theorem shows that there is a systematic (in fact computable) way of getting a finite argument of correctness about finite computations from the program and the specification (if a simulation exists, for example by assuming that program and specification are deterministic).

The verification method based on WF progress measures is optimal in the following sense. For specifications that are Σ_1^1, it is Π_2^1-complete to determine whether a Σ_1^1 program satisfies the specification. For example, determining whether $L(\mathcal{P}) \subseteq L(\mathcal{Q})$, where $\mathcal{P}$ and $\mathcal{Q}$ are recursively represented nondeterministic transition systems is Π_2^1-complete [25]. It is hardly imaginable that a reasonable verification method would not be in Σ_2^1, which allows one to guess relations and verify that they are well-founded. But even a Σ_2^1 method cannot possible solve the Π_2^1-complete verification problem for Σ_1^1 sets. In this sense the preceding results are optimal.

Finally, we observe that, just as Suslin's Theorem is a consequence of Theorem 12, the Finite Argument Theorem implies Kleene's Theorem, which states that there is a uniform way of obtaining an index in the hyperarithmetical hierarchy of a set L from a Π_1^1 and a Σ_1^1 index of L [23].

7 Borel Programs

To describe Borel sets in terms that are useful for verification of programs, we introduce a class of programs whose acceptance conditions are infinitary temporal logic formulas. This will also allow us to prove the Boundedness Theorem for Π_1^1 sets.

Definition 14. By transfinite induction, we define a *ranked formula* ϕ_γ, where γ is a countable ordinal, to be either a *temporal predicate* $\Box\Diamond\Phi$ ("infinitely often Φ") or $\Diamond\Box\Phi$ ("almost always Φ"), where Φ is a predicate on V, or a disjunction $\bigvee_{\gamma'<\gamma} \phi_{\gamma'}$ or a conjunction $\bigwedge_{\gamma'<\gamma} \phi_{\gamma'}$, where $\phi_{\gamma'}$ are ranked formulas.

A sequence $v_0 v_1 \cdots$ *satisfies* ϕ_γ, written $v_0 v_1 \cdots \models \phi_\gamma$, according to:

$v_0 v_1 \cdots \models \Diamond\Box\Phi$ if and only if $\exists H : \forall h > H : v_h \models \Phi$
$v_0 v_1 \cdots \models \Box\Diamond\Phi$ if and only if $\forall H : \exists h > H : v_h \models \Phi$
$v_0 v_1 \cdots \models \bigwedge_{\gamma'<\gamma} \phi_{\gamma'}$ if and only if $\forall \gamma' < \gamma : v_0 v_1 \cdots \models \phi_{\gamma'}$
$v_0 v_1 \cdots \models \bigvee_{\gamma'<\gamma} \phi_{\gamma'}$ if and only if $\exists \gamma' < \gamma : v_0 v_1 \cdots \models \phi_{\gamma'}$

Definition 15. A *Borel program* $\mathcal{P} = (P, \rightarrow, p^0, \phi_\nu)$ consists of a countable set of states P, a deterministic transition relation $\rightarrow \subseteq P \times \Sigma \times P$, an initial state p^0, and a ranked formula ϕ_ν.

Proposition 16. *The class of live languages accepted by Borel programs is the class of Borel sets.*

7.1 Proof of the Boundedness Theorem of Section 4

7.2 Borel Sets Are Analytic and Coanalytic

We show how to translate the temporal logic acceptance condition of a Borel program into a WF or $\neg$WF condition of a progress approximation defined on Σ^*. By this translation, program verification with temporal logic can take place by measuring progress using Theorem 8 or Theorem 12. The translation also proves that all Borel sets are analytic and coanalytic.

Theorem 17. *Let $\mathcal{P}$ be a Borel program. Then there exist progress approximations τ and τ' on Σ^* such that*

$$\lim_{WF}\ \tau = L^0(\mathcal{P})$$

$$\lim_{\neg WF}\ \tau' = L^0(\mathcal{P})$$

It is usually not possible to define τ as a function of the current state. Instead the whole history of states or actions must be used. In particular, a finite-state Borel program becomes infinite-state. (In contrast, note that Büchi conditions allow certain restricted third level properties to be expressed without going to infinite-state systems.)

In order to prove this theorem we need two lemmas. They show how to merge countably many sequences of finite trees into one such sequence that satisfies the WF condition if and only if all (respectively, one) of the original sequences satisfy the WF condition.

Lemma 18. *There is an operation $merge^\forall$ that merges any list of finite trees into a finite tree such that for any collection $(\tau_i^j)_i$, $j \in \omega$, of sequences of finite trees:*

$$WF\lim_{i\rightarrow\omega} merge^\forall(\tau_0^i, \ldots, \tau_i^i)$$
if and only if
$$\forall j : WF\lim_{i\rightarrow\omega} \tau_i^j$$

Lemma 19. *There is an operation $merge^\exists$ that merges any list of finite trees into a finite tree such that for any collection $(\tau_i^j)_i$, $j \in \omega$, of sequences of finite trees:*

$$WF\lim_{i\rightarrow\omega} merge^\exists(\tau_0^i, \ldots, \tau_i^i)$$
if and only if
$$\exists j : WF\lim_{i\rightarrow\omega} \tau_i^j$$

By Proposition 16, we have:

Corollary 20. *Borel sets are analytic and coanalytic.*

8 Progress Bisimulations

The full paper contains a formalization of the game outlined in Section 2.3. By Martin's result [19], one of the players has a winning strategy. If the responder has a winning strategy, then it can be described by a relation over finite computations and a progress measure for $Live_{\mathcal{P}} \Leftrightarrow Live_{\mathcal{Q}}$. This progress measure is formulated for the progress approximation $merge^{\Leftrightarrow}(\tau_{\mathcal{P}}(u), \tau_{\mathcal{Q}}(v))$, which is defined as $merge^{\wedge}(merge^{\vee}(\tau'_{\mathcal{P}}(u), \tau''_{\mathcal{Q}}(u)), merge^{\vee}(\tau''_{\mathcal{P}}(u), \tau'_{\mathcal{Q}}(u)))$.

9 Conclusion

We have used a limit view of finite computations to show that concepts of simulation and bisimulation can be be generalized to account also for very general liveness properties. The two generalized concepts establish a strong connection to two major theorems in descriptive set theory. The limit conditions presented here probably have only theoretical interest, however.

In practice, the mathematical challenge needed to establish even simple bisimulations for transitions systems with Büchi acceptance conditions seems quite difficult. Further investigation may reveal whether the concepts presented in this article may be sufficiently simplified for finite-state systems to be of use in practice.

Acknowledgements

Thanks to Dexter Kozen and Moshe Vardi for valuable comments on an earlier version of this article.

References

1. M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.
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 and E.-R. Olderog. Proof rules and transformations dealing with fairness. *Science of Computer Programming*, 3:65–100, 1983.
5. I. Dayan and D. Harel. Fair termination with cruel schedulers. *Fundamenta Informatica*, 9:1–12, 1986.
6. N. Francez and D. Kozen. Generalized fair termination. In *Proc. 11th POPL, Salt Lake City*. ACM, January 1984.
7. 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.

8. Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings 14th Symp. on Theory of Computing*. ACM, 1982.
9. D. Harel. Effective transformations on infinite trees with applications to high undecidability, dominos, and fairness. *Journal of the ACM*, 33(1):224–248, 1986.
10. N. Klarlund. Liminf progress measures. In *Proc. of Mathematical Foundations of Programming Semantics 1991*. LNCS.
11. N. Klarlund. Progress measures and stack assertions for fair termination. In *Proc. Eleventh Symp. on Princ. of Distributed Computing*, pages 229–240. IEEE, 1992.
12. N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. In *Proc. Seventh Symp. on Logic in Computer Science*, 1992.
13. 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.
14. N. Klarlund and F.B. Schneider. Proving nondeterministically specified safety properties using progress measures. *Information and Computation*, 107(1):151–170, 1993.
15. Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations*. PhD thesis, TR-1153, Cornell University, August 1990.
16. 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.
17. 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.
18. Z. Manna and A. Pnueli. Specification and verification of concurrent programs by ∀-automata. In *Proc. Fourteenth Symp. on the Principles of Programming Languages*, pages 1–12. ACM, 1987.
19. D.A. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.
20. Yiannis N. Moschovakis. *Descriptive Set Theory*, volume 100 of *Studies in Log. and the Found. of Math.* North-Holland, 1980.
21. L. Priese. Fairness. *EATCS Bulletin*, 50, 1993.
22. R. Rinat, N. Francez, and O. Grumberg. Infinite trees, markings and well-foundedness. *Information and Computation*, 79:131–154, 1988.
23. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill Book Company, 1967.
24. 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.
25. A.P. Sistla. On verifying that a concurrent program satisfies a nondeterministic specification. *Information Processing Letters*, 32(1):17–24, July 1989.
26. L. Staiger. Recursive automata on infinite words. In P. Enjalbert, A. Finkel, and K.W. Wagner, editors, *Proc. 10th Annual Symp. on Theoretical Computer Science (STACS), LNCS 665*. Springer Verlag, 1993.
27. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. MIT Press/Elsevier, 1990.
28. M. Vardi. Verification of concurrent programs: The automata-theoretic framework. *Annals of Pure and Applied Logic*, 51:79–98, 1991.
29. M. Vardi. Private communication, 1992.