Chicago Journal of Theoretical Computer Science

The MIT Press

Volume 1995, Article 3

20 September 1995

ISSN 1073–0486. MIT Press Journals, 55 Hayward St., Cambridge, MA 02142 USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www-mitpress.mit.edu/jrnls-catalog/chicago.html*
- *http://www.cs.uchicago.edu/publications/cjtcs/*
- *gopher.mit.edu*
- *gopher.cs.uchicago.edu*
- anonymous *ftp* at *mitpress.mit.edu*
- anonymous *ftp* at *cs.uchicago.edu*

The *Chicago Journal of Theoretical Computer Science* is a peer-reviewed scholarly journal in theoretical computer science. The journal is committed to providing a forum for significant results on theoretical aspects of all topics in computer science.

Rabin Measures

Nils Klarlund Dexter Kozen

20 September, 1995

Abstract

Abstract-1 *Rabin conditions* are a general class of properties of infinite sequences that encompass most known automata-theoretic acceptance conditions and notions of fairness. In this paper, we introduce a concept, called a *Rabin measure*, which in a precise sense expresses progress for each transition toward satisfaction of the Rabin condition.

Abstract-2 We show that these measures of progress are linked to the Kleene-Brouwer ordering of recursion theory. This property is used in [Kla94b] to establish an exponential upper bound for the complementation of automata on infinite trees.

Abstract-3 When applied to termination problems under fairness constraints, Rabin measures eliminate the need for syntax-dependent, recursive applications of proof rules.

This article is a revised version of [KK91]. Compared to the earlier version, the notion of Rabin measure has been simplified in order to be consistent with the use of the present results in [Kla94b].

1 Introduction

1-1 This paper is concerned with infinite sequences and their properties that are true or false in the limit. Such properties arise in the study of fairness, temporal logic, and ω-automata. In all these areas, *Rabin conditions* [McN66, Rab69], which express temporal properties in a special disjunctive normal form, play a major role. (They are also called *pairs conditions*.) In the theory of fairness, Rabin conditions describe termination under *general* *fairness constraints*, which include *strong fairness* expressing that commands that are infinitely often enabled are infinitely often executed. In temporal logic, Rabin conditions can be used to model a variety of liveness properties. In the theory of ω-automata, Rabin conditions are pivotal, because they allow ω-regular languages to be expressed by deterministic automata [McN66].

1-2 Because temporal conditions are often crucial to understanding the behavior of concurrent and distributed programs, a large number of proof methods for Rabin conditions and simpler conditions have been proposed in the context of fairness [AO83, APS84, DH86, FK84, Fra86, GFMdR85, Mai89, SdRG89], temporal logic [MP84, OL82], or automata theory [AS87, AS89, Jon87, MP87, Var91]. The essence of some of the proposed methods is obscured by syntactic transformations; others are limited to simple temporal formulas or expressed in rather involved automata-theoretical terms.

1-3 In this paper we address a fundamental question underlying several of the previous approaches: How can one explain, in terms of local conditions and without transformations, that a graph satisfies a Rabin condition? Our contribution is a concept, called a *Rabin measure*, which mathematically quantifies progress for transitions toward satisfying a Rabin condition. Thus a Rabin measure expresses closeness to satisfaction of the Rabin condition in the same way a well-founded set expresses closeness to program termination.

1-4 The main result of our paper is that a graph satisfies a Rabin condition if and only if it has a Rabin measure. Although this is not surprising from a recursion-theoretic point of view (the problem is complete for Π^1_1), the result is important because Rabin measures can be used to verify properties expressible by temporal formulas in a certain disjunctive normal form. Previous research has concentrated on temporal formulas in conjunctive form; such formulas can be verified by verifying each conjunct separately. Disjunctive formulas are significantly more difficult, and no system to date has been able to handle them without transformations.

1-5 Our treatment exhibits a link between verification with Rabin conditions and certain constructs in classical recursion theory. Specifically, we show that the transfinite well-orders that arise in such verification problems are all obtained in a natural way, as the Kleene-Brouwer ordering on the set of paths in certain finite-path trees. The "helpful directions" (see [Fra86]) arise simply and naturally in this context and can be explained completely in these terms.

1-6 Rabin measures are useful for studying automata on infinite objects. In [Kla91], it is shown how the concept can be used to complement Streett automata on infinite words. For automata on infinite trees, the results of the present article are used to establish that complementation can take place with only an exponential blow-up [Kla94b]. These applications rely on the relationship to the Kleene-Brouwer ordering, and do not follow from previous attempts at explaining progress for Rabin conditions.

2 Related Work

2-1 Alpern and Schneider [AS87, AS89] used deterministic Büchi automata as a specification method. They obtained a verification method for nondeterministic Büchi automata using the fact that every such automaton can be converted to a Boolean combination, in conjunctive normal form, of deterministic Büchi automata. Vardi [Var91] proposed *rank predicates* as a very general approach to verification, where specifications were subjected to certain automata-theoretic transformations. The resulting automata define incorrect computations and have a Büchi-type acceptance condition, which yields an explanation of helpful directions for the transformed verification problem.

2-2 Inspired by the ideas in [Var91], Manna and Pnueli gave verification conditions for $\forall$-automata [MP87]. These conditions are expressively equivalent to Büchi automata, although there is no known easy conversion of a Büchi automaton into a $\forall$-automaton. Sistla [Sis87] considered deterministic automata with acceptance conditions given as temporal formulas on automaton states with the modalities $F^\infty(f)$ (infinitely often f) and $G^\infty(f)$ (almost always f). He showed that sound and complete verification conditions exist for automata that are in a special conjunctive normal form in which each conjunct is a particularly simple disjunction.

2-3 For temporal logic specifications, assertional proof methods such as [MP84, OL82] are limited to quite simple temporal formulas that express properties like "leads to" and "always." A general approach to verification with finite temporal formulas was proposed in [RFG88]. It is based on establishing a direct correspondence between the program and the temporal formula, assigning a well-founded ordering to every program state. The verification conditions depend on inductively defined predicates on temporal formulas, and are rather complicated.

2-4 In the area of fairness, complete verification methods for termination were given in [FK84, Fra86, GFMdR85, Mai89, LPS81, SdRG89]. These methods are based on *helpful directions*, and on the iterative use of proof rules applied to syntactically transformed programs. In [Fra86, Section 2.1], a variation of this method based on relativizing the proof rules to state predicates is presented. This avoids syntactic transformations, but the method is still dependent on repeated applications of proof rules. Similarly, the ranking arguments in [EJ88] are iterative. In both cases, our Rabin measures may be viewed as arranging the information of all steps into one mathematical structure. Explicit ranks for the *parity condition* [Mos84] (which is a restricted kind of Rabin condition) were defined in [EJ91].

2-5 The methods of *explicit schedulers* developed in [AO83, APS84, DH86] 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 [Fra86].

2-6 The shuffling of colors in a Rabin measure is reminiscent of the *Later Appearance Record* of [GH82], which is used to explain how certain restricted memory strategies arise for infinite games played according to a Muller acceptance condition. In [Kla94b], it is shown how Rabin measures yield memoryless strategies for games with Rabin winning conditions.

2-7 *Progress measures* were introduced in [Kla90a] as a generic concept for quantifying how each step of a program contributes to bringing a computation closer to its specification, given in terms of a limit condition. There it is shown that progress measures exist for a variety of program specifications, including those involving nondeterminism, fairness, and infinitary temporal logics. For the general setting of the verification problem as studied in [Var91], rank predicates and progress measures are essentially equivalent [Var94] and their existence is in essence expressed by the Kleene-Suslin Theorem of descriptive set theory [Kla94a].

2-8 The paper [Kla92b] reformulates the notion of Rabin measure presented here, so that it can be better used in program verification. Also, the completeness proof presented in [Kla92b] is simplified for the case that the underlying program contains no cycles.

2-9 The verification method in [Kla92a] is based on the liminf concept, and also involves a variation on the Kleene-Brouwer ordering. Although applicable to a larger class of properties (in terms of the Borel classification), this method is dependent on characterizing finite computations, not states, and cannot be used without introducing history information. The method of [Kla92a] can be extended to general simulation and bisimulation concepts [Kla94a].

3 Rabin Conditions

3-1 A *graph* $G = (V, E)$ consists of a countable set of vertices (or states) V and a set of directed edges $E \subseteq V \times V$. 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 we write $v_0, v_1, \ldots \vDash (R, I)$, if there are infinitely many k such that $v_k \in R$ and only finitely many k such that $v_k \in I$.

3-2 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 finite set of *colors*, and Rabin pair (R_χ, I_χ) is said to have color χ. We assume that no pair in $\mathcal{C}$ is repeated, and say that $|\mathcal{C}| = |X|$ is the *size* of $\mathcal{C}$. The set R_χ is the set of χ-reconfirming states, and I_χ is the set of χ-invalidating states. For technical reasons, we always assume without loss of generality that $0 \in X$ and that $I_0 = \emptyset$ (one can always add the pair $(\emptyset, \emptyset)$ without changing the semantics of satisfaction defined next). We say that an infinite sequence $v_0, v_1, \ldots$ *satisfies* $\mathcal{C}$, and we write $v_0, v_1, \ldots \vDash \mathcal{C}$, if for some χ, $v_0, v_1, \ldots \vDash (R_\chi, I_\chi)$. We say that a graph $G = (V, E)$ satisfies a Rabin condition $\mathcal{C}$ on V, and we write $G \vDash \mathcal{C}$, if every infinite path $v_0, v_1, \ldots$ in G satisfies $\mathcal{C}$.

4 Pointer Trees

4-1 Rabin measures are based on pointer trees, also called direction trees. Let ω_1 be the set of countable ordinals. A *pointer tree* T is a countable prefix-closed subset of ω_1^*, the set of finite sequences of countable ordinals. Each sequence $t = \langle t^1, \ldots, t^\ell \rangle$ in T represents a *node*, which has *children* $t \cdot \langle d \rangle \in T$, where "$\cdot$" denotes concatenation of sequences. Here $d \in \omega_1$ is the *pointer* to $t \cdot \langle d \rangle$ from t. If t' is a prefix of $t \in T$, then t' is called an *ancestor* of t and we write $t' \mathrel{\dot{\le}} t$. We visualize pointer trees as growing upward; see Figure 1, where children are depicted from left to right in descending order. Every 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 $t = \langle t^1, \ldots, t^l \rangle$ is the number l; the level of $\langle\,\rangle$ is 0. The *prefix up to level* n of $t = \langle t^1, \ldots, t^l \rangle$ is $\langle t^1, \ldots, t^{\min\{l,n\}} \rangle$, denoted $t \restriction n$. The *height* of T is

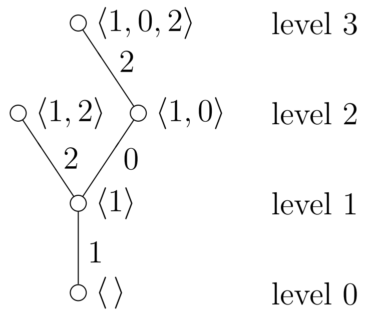

Figure 1: A pointer tree.

the maximum node level (if it exists). T is *path-finite* if there are no infinite paths in T. A *common ancestor* of nodes t and t' is a node $\tilde{t}$ that is an ancestor of both of t and of t'. The common ancestor at the highest level is the *highest common ancestor*, written $t \uparrow t'$.

4-2 A simple but central lemma about the infinite sequences of nodes in a pointer tree is:

Lemma 1 (Highest Common Ancestor Lemma) *Let $t_0, t_1, \ldots$ be an infinite sequence of nodes in a path-finite pointer tree T. Then there is a node t that is a common ancestor of t_k, t_{k+1} for almost all k, and the highest common ancestor for infinitely many k.*

Proof of Lemma 1 Let T' be the set of nodes that are almost always ancestors of t_k. Then T' contains $\langle\,\rangle$ and is linearly ordered by the prefix relation, since for all t, t' in T', there eventually exists t_k such that both t and t' are ancestors of t_k, and the ancestors of every node are linearly ordered. Since T contains only finite paths, T' has a unique maximal element, which satisfies the desired conditions.

$\square$

Definition 1 (Kleene-Brouwer Ordering) *The strict ordering $\succ$ on T is defined by: $t \succ t'$ if and only if either $t \mathrel{\dot{<}} t'$, or else the highest common ancestor of t and t' is at level l, t^{l+1} and t'^{l+1} are defined, and $t^{l+1} > t'^{l+1}$. The nonstrict ordering $\succeq$ is defined as $t \succeq t'$ if and only if $t \succ t'$ or $t = t'$.*

In other words $t \succeq t'$ if t is an ancestor of t' or if t' branches off to the right of t (assuming T is depicted as in Figure 1). $\succeq$ is a total order on T.

Lemma 2 (Kleene-Brouwer Ordering Lemma) *If T is path-finite, then $\succ$ is a well-ordering of T.*

Proof of Lemma 2 See [Rog67] or use Lemma 1.

□

4-3 Rabin measures are based on coloring pointer trees.

Definition 2 *A* colored pointer tree (T, ξ) *is a pointer tree T with a partial mapping $\xi : T \rightharpoonup X$, where X is a set of* colors, *assigning a color $\xi(t) \in X$ to each node t in* $\operatorname{dom} \xi$ *so that:*

- *the root is colored 0, i.e., $\xi(\langle\,\rangle) = 0$;*
- *each node that is not a leaf receives some color; and*
- *all colors along a path starting in the root are distinct.*

Thus, a colored pointer tree has height at most $|X|$.

5 Rabin Measures

5-1 A Rabin measure for a graph with a Rabin condition consists of a colored pointer tree and a mapping that assigns a node in the tree to each vertex in the graph. These nodes, or *progress values*, quantify progress toward the Rabin condition if they are related in a certain way on every transition.

Definition 3 *Let $G = (V, E)$ be a graph, and let $\mathcal{C} = \{\, (R_\chi, I_\chi) \mid \chi \in X \,\}$ be a Rabin condition on G. A* Rabin progress measure *(or just* Rabin measure*) for $(G, \mathcal{C})$ is a triple (μ, T, ξ), where $\mu : V \to T$ and (T, ξ) is a colored pointer tree, such that:*

(I) for all $v \in V$ and all $\chi \in \mu(v)$, $v \notin I_\chi$, and

(R) for all $(u, v) \in E$, $u \rhd_\mu v$,

where $u \rhd_\mu v$ if and only if

$\mu(u) \succ \mu(v)$, or

there exists a common ancestor $t \mathrel{\dot\leq} \xi(\mu(u) \uparrow \mu(v))$ such that $v \in R_{\xi(t)}$.

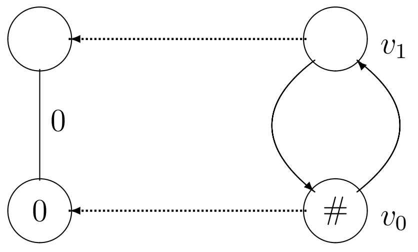

Figure 2: A colored pointer tree and a graph with Rabin measure.

The value $\mu(v)$ designates a path in T from the root to the node $\mu(v)$. The sequence of colors on this path indicates a prioritization of hypotheses about Rabin pairs: The closer it is to the bottom, the more likely a pair is to be the one satisfied in the limit.

5-2 Requirement (I) states that all hypotheses correspond to pairs that are not invalidated. Requirement (R) states that on a transition from u to v, either the progress value decreases from u to v or one of the pairs described by the common part of $\mu(u)$ and $\mu(v)$ is reconfirmed. The colors at or below the highest common ancestor indicate the "helpful directions," that is, the pairs that have been singled out for satisfaction in the limit. It is shown below how (R) insures that some color is reconfirmed infinitely often, and how (I) insures that the color is invalidated only finitely many times.

5-3 In Figure 2, we have shown a graph G (to the right) with vertices v_0 and v_1. The Rabin condition $\mathcal{C}$ of the graph consists of the pair (R_0, I_0), where the set X is $\{0\}$, and $R_0 = \{v_0\}$ consists of the node marked #, and $I_0 = \emptyset$. Thus, every infinite path in G runs through a reconfirming state infinitely often, and $G \models \mathcal{C}$. This fact can also be established by the existence of a Rabin measure. We use the colored pointer tree shown to the left. Its root is labeled with the color 0 and the node $\langle 0 \rangle$ is not labeled with a color. The measure is as indicated by the dotted arrows, i.e., $\mu(v_0) = \langle\,\rangle$ and $\mu(v_1) = \langle 0 \rangle$. On the transition from v_0 to v_1, the value of the measure decreases according to the Kleene-Brouwer ordering, and on the transition from v_1 to v_0, the value of the measure also decreases, since the root is a common ancestor with a color that is reconfirmed.

5-4 Another more complicated situation involving two colors ($X = \{0, 1\}$) is shown in Figure 3. Here $R_0 = \{v_0\}$, $I_0 = \emptyset$, $R_1 = \{v_1\}$ (the node marked +), and $I_1 = \{v_3\}$ (the node marked $-$).

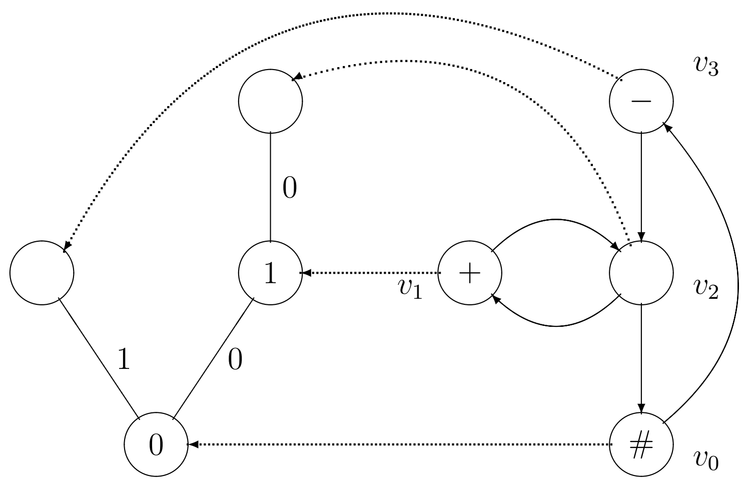

Figure 3: Tree, graph, measure, with two nontrivial colors.

5-5 The main result of this article is:

Theorem 1 *$G \models \mathcal{C}$ if and only if $(G, \mathcal{C})$ has a Rabin measure.*

Proof of Theorem 1-1 **Proof of Theorem 1 ($\Longleftarrow$)** Let $v_0, v_1, \ldots$ be an infinite path in G and let $t_k = \mu(v_k)$. By Lemma 1 (Highest Common Ancestor), there is a node t that is (a) almost always a common ancestor and (b) infinitely often the highest common ancestor of t_k, t_{k+1}. Let $l = |t|$. Next, we prove that there exists $\hat{t} \underset{\cdot}{<} t$ such that $v_0, v_1, \ldots$ satisfies the Rabin pair $(R_{\xi(\hat{t})}, I_{\xi(\hat{t})})$.

Proof of Theorem 1-2 First, assume for a contradiction that for all $\hat{t} \underset{\cdot}{\leq} t$, $v_k \in R_{\xi(\hat{t})}$ holds only finitely often. Thus by (a) there is a K such that for all $k \geq K$, either (1) there is a t'_k with $t \underset{\cdot}{<} t'_k \underset{\cdot}{\leq} t_k \uparrow t_{k+1}$ and with $v_k \in R_{\xi(t'_k)}$, or (2) $t_k \succ t_{k+1}$.

Proof of Theorem 1-3 In case (1), $t_k, t_{k+1} \underset{\cdot}{>} t$ and $t_k \restriction (l+1) = t_{k+1} \restriction (l+1)$. In case (2), which holds infinitely often by assumption and by (b), $t_k \restriction (l+1) \succ t_{k+1} \restriction (l+1)$, since $t_k \succ t_{k+1}$ and t is the highest common ancestor of t_k and t_{k+1}. Thus $t_K \restriction (l{+}1) \succeq t_{K+1} \restriction (l{+}1) \succeq \cdots$ and infinitely many inequalities are strict. This contradicts the Kleene-Brouwer Ordering Lemma. Thus it holds infinitely often that $v_k \in R_{\xi(\hat{t})}$ for some $\hat{t} \underset{\cdot}{\leq} t$.

Proof of Theorem 1-4 Second, by (I) it follows that $v_k \notin I_{\xi(\hat{t})}$ holds from some point on due to (a).

Proof of Theorem 1 ($\Longleftarrow$) □

We prove the ($\Longrightarrow$) portion in Section 6.3.

6 Construction of Rabin Measures

6-1 We show how to construct a Rabin measure for $(G, \mathcal{C})$, where $G = (V, E)$ is a graph that satisfies a Rabin condition $\mathcal{C} = \{ (R_\chi, I_\chi) \mid \chi \in X \}$.

6.1 Color Set Assignments

6.1-1 We need some definitions before stating a key lemma. Let $\mathcal{P}C$ denote the class of subsets of C. A *color set assignment* is a map $CS\colon V \to \mathcal{P}C$, where C is a countable set of colors; CS associates a nonempty set of *enabled* colors $CS(v) \subseteq C$ to each vertex $v \in V$. A set $W \subseteq V$ is χ-*enabled* if and only if, for all $v \in W$, $\chi \in CS(v)$. An infinite path $v_0, v_1, \ldots$ is *eventually* χ-*enabled* if and only if there is a χ-*enabled* suffix $v_k, v_{k+1}, \ldots$. A color set assignment is *permissible* if and only if every infinite path is eventually χ-enabled for some χ.

6.1-2 The set of *descendants* $\mathcal{R}(v)$ of a vertex v in a graph G is the set of all v' such that there is path from v to v'. Note that $v \in \mathcal{R}(v)$. The key lemma is:

Lemma 3 *Let $G = (V, E)$ be a countable graph. If $CS\colon V \to \mathcal{P}C$ is a permissible color set assignment, then there is a vertex v and a color χ such that $\mathcal{R}(v)$ is χ-enabled.*

Before proving the lemma, we recall that a set Z is *nowhere dense* if there is no nonempty open set O such that $O \cap Z$ is dense in O. We will use:

Theorem 2 (The Baire Category Theorem) *Let M be a complete metric space. Then M is not a countable union of nowhere-dense sets. In particular, M is not a countable union of closed sets that contain no basic open sets.*

Proof of Lemma 3-1 **Proof of Lemma 3** For every finite path u in G, define B_u and M by

$$\begin{aligned} B_u &= \{\, u \cdot w \mid u \cdot w \text{ is a path (finite or infinite) in } G \,\} \\ M &= \{\, w \mid w \text{ is a path (finite or infinite) in } G \,\} \end{aligned}$$

Proof of Lemma 3-2 The B_us form a subbasis for a topology over M, where the open sets are unions of B_us. M is a complete metric space.[1]

Proof of Lemma 3-3 For a finite path u in G, and for $\chi \in C$, define the set

$$F_{u,\chi} = \{\, u \cdot w \mid w \text{ is } \chi\text{-enabled} \,\}$$

which is closed. Every finite path u is contained in $F_{u,\chi}$ for all χ, and every infinite path w is in $F_{u,\chi}$, for some u and χ, by the assumption that w is eventually χ-enabled. Thus $M = \bigcup_{u,\chi} F_{u,\chi}$.

Proof of Lemma 3-4 By the Baire Category Theorem, some $F_{u,\chi}$ contains a basic open set, i.e., contains some B_v. Consequently, $\mathcal{R}(v)$ is χ-enabled.

Proof of Lemma 3 □

6.2 Colorings

6.2-1 Lemma 3 can be applied transfinitely to a graph to yield a stronger result. We need a few definitions. Let C be a countable set of colors. A *C-coloring* c of a set V is a total mapping $c\colon V \to C$. A coloring c *obeys* a color set assignment CS if and only if, for all $v \in V$, $c(v) \in CS(v)$. An infinite path $v_0, v_1, \ldots$ is *eventually χ-stable* with respect to c, where $\chi \in C$, if for almost all i, $c(v_i) = \chi$. A coloring c is *eventually stable* if every infinite path is eventually χ-stable for some χ. A set $W \subseteq V$ is *monochromatic* with respect to c if there is $\chi \in C$ such that for all $v \in W$, $c(v) = \chi$.

Lemma 4 *Let $G = (V, E)$ be a graph. If CS is a permissible color set assignment, then there is an eventually stable coloring $c\colon V \to C$ obeying CS and a partition $\mathcal{S} = \{\, W_\theta \mid \theta < \gamma \,\}$ of V, where γ is a countable ordinal, such that:*

[1]The metric can be given, for example, as:

$$d(w, w') = \begin{cases} 0 & \text{if } w = w' \\ 1/i & \text{if } w_i \neq w'_i \text{ and } w_j = w'_j \text{ for } 1 \leq j < i \\ 1/(i+1) & \text{if } w \text{ is a prefix of } w' \text{ of length } i \\ 1/(i+1) & \text{if } w' \text{ is a prefix of } w \text{ of length } i \end{cases}$$

(a) each W_θ is monochromatic; and

(b) if $(v, v') \in E$, $v \in W_\theta$, and $v' \in W_{\theta'}$, then $\theta \geq \theta'$.

Proof of Lemma 4-1 **Proof of Lemma 4** We apply Lemma 3 transfinitely. More precisely, Lemma 3 is first applied to yield a vertex v in G such that $\mathcal{R}(v)$ is χ-enabled for some χ. Define $W_0 = \mathcal{R}(v)$ and $c(v) = \chi$ for $v \in W_0$. Then remove W_0 from G and apply Lemma 3 again to define W_1 in a similar manner. By transfinite induction, this process induces a partition of G into γ classes, where γ is a countable ordinal.

Proof of Lemma 4-2 Then (a) is satisfied according to the definition of c. Also, (b) is satisfied, because every vertex is removed along with all its successors in the remaining graph.

Proof of Lemma 4 $\square$

6.3 Proof of Theorem 1 ($\Longrightarrow$)

6.3-1

Proof of Theorem 1-5 **Proof of Theorem 1 ($\Longrightarrow$)** Let $G = (V, E)$ be a graph that satisfies a finite Rabin condition $\mathcal{C}$. To construct a measure $(\mu, (T, \xi))$ of $(G, \mathcal{C})$, we use the algorithm **AssignRabin** in Figure 4. The algorithm builds the tree (T, ξ) and labels each node $t \neq \langle\,\rangle$ of T with a set $W(t) \subseteq V$, which is to be the set of vertices mapped by μ to nodes having t as an ancestor. The purpose of **AssignRabin**(t, Y, χ) is to assign color χ to node t and to define the children $t \cdot \langle\theta\rangle$ of node t. Each child receives a label $W(t \cdot \langle\theta\rangle)$, which is a subset of $W(t)$. The set Y denotes the colors that have already been assigned to the proper ancestors of t.

Proof of Theorem 1-6

- Initially, **AssignRabin** is applied with parameters $(\langle\,\rangle, \emptyset, 0)$, and the tree T consists of only the root $\langle\,\rangle$ with label $W(\langle\,\rangle) = V$.
- In Step 1 of **AssignRabin**, node t is assigned color χ, and $\widehat{W} = W(t) \setminus R_\chi$ is the set of states in $W(t)$ that are not χ-reconfirming.
- In Step 2, Lemma 4 is used to obtain a partition $\mathcal{S} = \{\, W_\theta \mid \theta < \gamma \,\}$ of $\widehat{W}$ and a coloring c of $\widehat{W}$. Informally, the color set assignment $CS^{\overline{Y \cup \{\chi\}}}$ expresses the set of colors that are not invalidating and that

```
**AssignRabin**$(t, Y, \chi)$ :

  1. $\xi(t) := \chi$

     $\widehat{W} := W(t) \setminus R_\chi$

  2. Use Lemma 4 on $G \setminus \widehat{W}$ with color set assignment $CS^{\overline{Y \cup \{\chi\}}}$ (explained
     in the text) to obtain a coloring $c$ and a partition $\mathcal{S} = \{\, W_\theta \mid \theta < \gamma \,\}$
     of $\widehat{W}$.

  3. For each class $W_\theta$ of $\mathcal{S}$:

     (a) $T := T \cup \{t \cdot \langle\theta\rangle\}$
     (b) $W(t \cdot \langle\theta\rangle) := W_\theta$

  4. For each $\chi'$-colored class $W_\theta$ of $\mathcal{S}$, where $\chi' \in X$:

         **AssignRabin**$(t \cdot \langle\theta\rangle, Y \cup \{\chi\}, \chi')$
```

Figure 4: Algorithm **AssignRabin**.

have not already been considered. More precisely, the color set assignment $CS^{\overline{Y \cup \{\chi\}}}$ assigns to vertex v the set of colors $\chi' \in X$ such that $\chi' \notin Y \cup \{\chi\}$ and $v \notin I_{\chi'}$; however, if this set is empty, then the color set assigned is $\{\bot_v\}$, where $\bot_v$ is a distinct dummy color, different from any color defined elsewhere. Thus the set C of all colors is $X \cup \{ \bot_v \mid v \in V \}$.

- In Step 3, a child $t \cdot \langle\theta\rangle$ is added to t for each class W_θ of $\mathcal{S}$ and $t \cdot \langle\theta\rangle$ is labeled W_θ.
- Finally, in Step 4, descendants of each child not assigned a dummy color are constructed by further applications of **AssignRabin**.

Proof of Theorem 1-7 To explain the construction of μ and to prove that Lemma 4 can indeed always be used in Step 2, we need some terminology. We say that a subset $W \subseteq V$ is χ-*nonreconfirming* if $W \cap R_\chi = \emptyset$ and that W is χ-*noninvalidating* if $W \cap I_\chi = \emptyset$. A subset $W \subseteq V$ is Y-*nonreconfirming* if it is χ-nonreconfirming for each χ in Y. Similarly, a subset $W \subseteq V$ is Y-*noninvalidating* if it is χ-noninvalidating for each χ in Y.

Claim 1 *For each application of the inductive procedure* ***AssignRabin***(t, Y, χ) *starting with* ***AssignRabin***$(\langle\,\rangle, \emptyset, 0)$, *the following hold:*

- $Y \subseteq X$ *and* $|Y| = |t|$*;*
- $\chi \in X \setminus Y$*;*
- $W(t)$ *is* Y*-nonreconfirming; and*
- $W(t)$ *is* $(Y \cup \{\chi\})$*-noninvalidating.*

Proof of Claim 1-1 **Proof of Claim 1** (By induction) This is true for the first application **AssignRabin**$(\langle\,\rangle, \emptyset, 0)$ because $|Y| = |\emptyset| = |\langle\,\rangle| = |t| = 0$; $\chi = 0$; and $W(t) = W(\langle\,\rangle) = V$, which is $\emptyset$-nonreconfirming and, since $I_0 = \emptyset$, $\{0\}$-noninvalidating.

Proof of Claim 1-2 When **AssignRabin**$(t \cdot \langle\theta\rangle, Y \cup \{\chi\}, \chi')$ is applied from within **AssignRabin** in Step 4, it may be assumed by the induction hypothesis that $|Y| = |t|$, $\chi \in X \setminus Y$, and $W(t)$ is $(Y \cup \{\chi\})$-noninvalidating and Y-nonreconfirming.

Proof of Claim 1-3 It follows that $Y \cup \{\chi\} \subseteq X$ and $|Y \cup \{\chi\}| = |t \cdot \langle\theta\rangle|$. Also, by definition of the color set assignment in Step 2, $\chi' \in X \setminus (Y \cup \{\chi\})$. Furthermore, since W_θ is χ'-noninvalidating by this definition, W_θ is $(Y \cup \{\chi\} \cup \{\chi'\})$-noninvalidating. Finally, since W_θ is χ-nonreconfirming (because $W_\theta \subseteq \widehat{W} = W \setminus R_\chi$), it follows that $W(t \cdot \langle\theta\rangle) = W_\theta$ is $(Y \cup \{\chi\})$-nonreconfirming.

Proof of Claim 1 □

Proof of Theorem 1-8 To see that Lemma 4 is applicable in Step 2 of **AssignRabin**, we prove:

Claim 2 *In every application of* ***AssignRabin****, $CS^{\overline{Y \cup \{\chi\}}}$ is permissible for $\widehat{W}$; in fact, every infinite path in W is eventually $\widehat{\chi}$-enabled for some $\widehat{\chi} \in X \setminus (Y \cup \{\chi\})$.*

Proof of Claim 2-1 **Proof of Claim 2** Consider an application **AssignRabin**(t, Y, χ). By Claim 1, it follows that $\widehat{W}$ defined in Step 1 of **AssignRabin** is $(Y \cup \{\chi\})$-noninvalidating and $(Y \cup \{\chi\})$-nonreconfirming. Now let $v_0, v_1, \ldots$ be an infinite path in $\widehat{W}$. It is $(Y \cup \{\chi\})$-nonreconfirming because $\widehat{W}$ is $(Y \cup \{\chi\})$-nonreconfirming. Hence by the assumption that $G \vDash \mathcal{C}$, $v_0, v_1, \ldots \vDash (R_{\hat{\chi}}, I_{\hat{\chi}})$ for some $\hat{\chi} \in X \setminus (Y \cup \{\chi\})$. In particular, $v_0, v_1, \ldots$ is eventually $\hat{\chi}$-enabled with respect to the color set assignment $CS^{\overline{Y \cup \{\chi\}}}$ of $\widehat{W}$.

Proof of Claim 2 □

Proof of Theorem 1-9 Now, to show that all nodes constructed are at level $\leq |X| = |\mathcal{C}|$, we note that if $|t| = |X| - 1$, then since $|Y| = |t|$ and $\chi \notin Y$ (by Claim 1, $Y \cup \{\chi\} = X$). In that case, the color set assignment of Step 2 assigns to each vertex v only the dummy color $\perp_v$; thus, **AssignRabin** is not further applied in Step 4. It follows that **AssignRabin** defines a tree T of height at most $|X|$. The tree has the following properties:

Claim 3 *(a) For every $t \in T$ with $|t| < |X|$,*

$$\{\, W(t \cdot \langle\theta\rangle) \mid t \cdot \langle\theta\rangle \in T \,\}$$

is a partition of $W(t) \setminus R_{\xi(t)}$.

(b) For each $v \in V$ there is a unique maximal list $t = \langle t^1, \ldots, t^n\rangle$ with $1 \leq n \leq |X|$ such that $v \in W(t)$.

Proof of Claim 3 (a) holds for $t = \langle\,\rangle$, because $W(\langle\,\rangle) = V$ and $\widehat{W}$ formed in **AssignRabin** is $W(\langle\,\rangle) \setminus R_{\xi(\langle\,\rangle)} = V \setminus R_0$; thus Lemma 4 in Step 2 and the definition of children in Step 4 yield a partition $\{\, W(\langle\theta\rangle) \mid \langle\theta\rangle \in T \,\}$ of $W(\langle\,\rangle) \setminus R_{\xi(\langle\,\rangle)}$. Similarly, by induction it can been seen that (a) holds for all $t \in T$ with $|t| < |X|$. Part (b) follows from (a).

Proof of Claim 3 □

Proof of Theorem 1-10 Using Claim 3b, we define $\mu: V \to T$ by $\mu(v) = \langle t^1, \ldots, t^n\rangle$. Now let us prove that (I) and (R) are satisfied.

(I) Note that if $t \,\dot{\leq}\, \mu(v)$ and $\chi = \xi(t)$ is defined, then by Claim 1, $W(t)$ is χ-noninvalidating. Thus in particular, $v \notin I_\chi$.

(R) Let $(u, v) \in E$. If there is a color χ of a common ancestor of $\mu(u)$ and $\mu(v)$ such that $v \in R_\chi$, then (R) is trivially satisfied. So we may assume that for the highest common ancestor $\hat{t}$, $v \notin R_{\xi(\hat{t})}$. Then $|\mu(v)| > |\hat{t}|$ by Claim 3a. Consider the application **AssignRabin**$(\hat{t}, Y, \xi(\hat{t}))$.

If $u \in R_{\xi(\hat{t})}$, then $\mu(u) = \hat{t}$ because $u \notin \widehat{W} = W(t) \setminus R_{\xi(\hat{t})}$. Thus $\mu(u) = \hat{t} \succ \mu(v)$, because $|\mu(v)| > |\hat{t}|$. Otherwise, if $u \notin R_{\xi(\hat{t})}$, then by Claim 3a, there are θ and θ' such that $u \in W_\theta$ and $v \in W_{\theta'}$. Since $\hat{t} \cdot \theta$ is a prefix of $\mu(u)$, $\hat{t} \cdot \theta'$ is a prefix of $\mu(v)$, and $\hat{t}$ is the highest common ancestor of $\mu(u)$ and $\mu(v)$, it follows that $\theta \neq \theta'$. Then by Lemma 4, $\theta > \theta'$, because $(u, v) \in E$. Therefore, $\mu(u) \succ \mu(v)$. Thus in all cases (R) holds.

Proof of Theorem 1 ($\Longrightarrow$) □

7 Application to Fairness

7-1 Our results apply to proving program termination under a general fairness constraint $\mathcal{C} = \{(\phi_1, \psi_1), \ldots, (\phi_N, \psi_N)\}$, which is defined in [FK84] and [Fra86, p. 112]. Each (ϕ_χ, ψ_χ), $1 \leq \chi \leq N$ consists of an *enabling condition* ϕ_χ and an *action condition* ψ_χ, both of which are program-state predicates. An infinite computation of P is *unfair* if it satisfies $\mathcal{C}$ regarded as a Rabin condition, i.e., if for some χ, the enabling condition ϕ_χ is satisfied infinitely often and the action condition ψ_χ is satisfied only finitely often. A program

$$
\begin{array}{lllllll}
*[& a: & x=0 & & & \rightarrow & y := y+1 \\
 & b: & x=0 & \wedge & even(y) & \rightarrow & x := 1 \\
 & c: & x \neq 0 & \wedge & y \neq 0 & \rightarrow & y := y-1 \\
 & d: & x \neq 0 & \wedge & y \neq 0 & \rightarrow & z := z+1 \quad]
\end{array}
$$

Figure 5: The program P_{ex}.

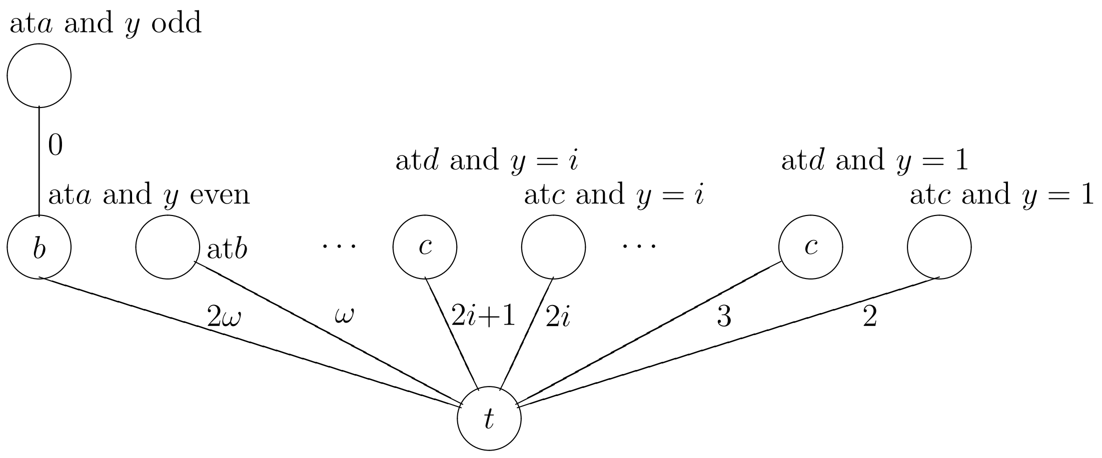

Figure 6: The pointer tree, its coloring, and the measure.

P fairly terminates if every infinite computation of P is unfair, i.e., if P viewed as a graph (where nodes are states and edges are transitions) satisfies $\mathcal{C}$. Thus to show fair termination of P, we can use Theorem 1.

7-2

Example 1-1 **Example 1** Program P_{ex}, shown in Figure 5, is taken from [GFMdR85] and can also be found in [Fra86]. The variables x, y, and z take on non-negative integer values. Program P_{ex} terminates under the assumption of *strong fairness*: For every infinite computation there is some guarded command l that is unfairly executed, i.e., infinitely often enabled but only finitely often executed. Thus the fairness constraint $\mathcal{C}$ can be written

$$\{(\phi_a, \psi_a), (\phi_b, \psi_b), (\phi_c, \psi_c), (\phi_d, \psi_d), (\phi_t, \psi_t)\}$$

where the pair (ϕ_l, ψ_l), $l = a, b, c, d$, denotes that command l is unfairly executed. Thus ϕ_l is the guard of l, and ψ_l is a predicate denoting that

l is the guarded command to be executed. The additional pair $(\phi_t, \psi_t) = (\mathit{false}, \mathit{false})$ is introduced for technical reasons to account for progress toward termination.

Example 1-2 To prove that $G(P_{ex}) \models \mathcal{C}$, we define a progress measure $(\mu, (T, \xi))$. The tree T is

$$\{\langle\,\rangle, \langle 2\rangle, \langle 3\rangle, \ldots, \langle\omega\rangle, \langle 2\omega\rangle, \langle 2\omega, 0\rangle\}$$

and the coloring ξ is defined by

$$\xi(\langle\,\rangle) = t, \xi(\langle 2\omega\rangle) = b, \text{ and } \xi(\langle k\rangle) = c, \text{ where } k < \omega \text{ and } k \text{ is odd}$$

See Figure 6, where also the mapping μ is indicated. Formally, μ is defined by:

$$\mu(x,y) = \begin{cases} \langle 2\omega, 0\rangle & \text{if at}a \text{ and } y \text{ is odd} \\ \langle 2\omega\rangle & \text{if at}a \text{ and } y \text{ is even} \\ \langle\omega\rangle & \text{if at}b \\ \langle 2y+1\rangle & \text{if at}d \\ \langle 2y\rangle & \text{if at}c \end{cases}$$

where y denotes the value of program variable y and at$l$ denotes that the program counter is at l.

Example 1-3 For the program P_{ex}, it can be verified that (I) holds. For example, the colors of nodes on the path to $\langle 2\omega, 0\rangle$ consists of b alone, and ψ_b is certainly false when a is selected for execution. With each iteration of the loop, the value of μ changes according to (R). For example, consider the case when $x = 0$, y is odd, and a is executed with a also being the new value of the program counter. Then μ changes from $\langle 2\omega, 0\rangle$ to $\langle 2\omega\rangle$ and b is the label of a common ancestor and $\phi_b = (x = 0 \wedge \mathit{even}(y))$ is satisfied in the new state, whence (R) holds.

Example 1 □

The termination proof of the program P_{ex} in [Fra86, GFMdR85] involves reasoning, not only about the original program, but also about two transformed programs.

7-3 In practice, it is convenient to reformulate the notion of Rabin measure so that condition (I) expresses that on a transition from u to v only the colors of common ancestors of u and v should not be invalidating. Other changes make it possible to define concisely both T and μ by assertions in the program text as described in [Kla92b].

8 Automata-Theoretic Applications

8-1 The verification problem for automata has been widely studied and can be formulated as follows. An automaton A_P, called a *program automaton*, satisfies an automaton A_S, called a *specification automaton*, if the language $L(A_P)$ defined by A_P is a subset of the language $L(A_S)$ defined by A_S, i.e., if every behavior of A_P is a behavior of A_S.

8-2 In this section we indicate how Rabin measures solve the problem of finding verification conditions for proving that a program satisfies a specification given by a deterministic Rabin automaton. Combined with Safra's result [Saf88], this yields a verification method for specifications given as nondeterministic finite-state Büchi automata.

8-3 An *automaton* $A = (\Sigma, V, \rightarrow, V^0)$ consists of a countable *alphabet* Σ, a countable *state space* V, a *transition relation* $\rightarrow \subseteq V \times \Sigma \times V$, and a set of *initial states* $V^0 \subseteq V$. If V^0 and all sets $\{\, v' \mid v \xrightarrow{e} v' \,\}$ have at most one element, then A is *deterministic*. A *run* (computation) of A over a behavior $e_0, e_1, \ldots$ is an infinite sequence of states $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots$ with $v_0 \in V^0$. A behavior $e_0, e_1, \ldots$ is *accepted by* A if there is a run of A over $e_0, e_1, \ldots$. The *language* or *property* $L(A)$ *accepted by* A is the set of behaviors of A.

8-4 A *deterministic Rabin automaton* $A = (\Sigma, V, \rightarrow, \{v^0\}, \mathcal{C})$ is defined as a deterministic automaton, but in addition it is equipped with a Rabin condition $\mathcal{C}$ on V. A run $v_0 \xrightarrow{e_o} v_1 \xrightarrow{e_1} \cdots$ of a Rabin automaton A over a behavior $e_0, e_1, \ldots$ is *accepting* if $v_0, v_1, \ldots \models \mathcal{C}$. The language $L(A)$ *accepted by* A is the set of behaviors whose run is accepting.

8-5

We can now use Rabin measures to solve the verification problem $L(A_P) \subseteq L(A_S)$, where the program automaton $A_P = (\Sigma, V_P, \rightarrow_P, V_P^0)$ is nondeterministic and the specification automaton $A_S = (\Sigma, V_S, \rightarrow_S, \{s^0\}, \mathcal{C})$ is a deterministic Rabin automaton. In fact, it suffices to notice that $L(A_P) \subseteq L(A_S)$ holds if and only if all paths in the reachable part of the joint state graph (formed by the automata-theoretic product of $L(A_P)$ and $L(A_S)$) satisfy the Rabin condition $\mathcal{C}$.

Corollary 1 (of Theorem 1) *Let A_P be an automaton and let A_S be a deterministic Rabin automaton. Then $L(A_P) \subseteq L(A_S)$ if and only if there is progress measure for $\mathcal{C}$ on the jointly reachable states.*

8-6 For further applications see [Kla90b], where the method for Rabin automata is extended to the $\forall$-automata of [MP87] and applied to simplify the method of [AS89].

9 Acknowledgments

9-1 Thanks to two anonymous referees for useful comments. We are also grateful to the managing editor, Michael J. O'Donnell, for his careful help in polishing the article.

Acknowledgement of support: Nils Klarlund was partially supported by a research stipend from the University of Aarhus, Denmark; an Alice & Richard Netter Scholarship of the Thanks to Scandinavia Foundation, Inc., USA; and Forskerakademiet, Denmark. Part of this research was carried out while the author was with IBM T. J. Watson Research Center, NY, USA. Dexter Kozen was partially supported by NSF grant CCR-8901061.

References

[AO83] K. R. Apt and E.-R. Olderog. Proof rules and transformations dealing with fairness. *Science of Computer Programming*, 3:65–100, 1983.

[APS84] K. R. Apt, A. Pnueli, and J. Stavi. Fair termination revisited with delay. *Theoretical Computer Science*, 33:65–84, 1984.

[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 and Systems*, 11(1):147–167, January 1989.

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

[EJ88] E. A. Emerson and C. S. Jutla. Complexity of tree automata and logics of programs. In *Proceedings of the 29th Symposium on Foundations of Computer Science*, pages 328–337, Los Alamitos, CA, 1988. Institute of Electrical and Electronics Engineers.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 368–377, Los Alamitos, CA, 1991. Institute of Electrical and Electronics Engineers.

[FK84] N. Francez and D. Kozen. Generalized fair termination. In *Proceedings of the 11th Symposium on Principles of Programming Languages, Salt Lake City*, pages 46–53. Association for Computing Machinery, January 1984.

[Fra86] N. Francez. *Fairness*. Springer-Verlag, Berlin, Heidelberg, 1986.

[GFMdR85] 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.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the 14th Symposium on Theory of Computing*, pages 60–65. Association for Computing Machinery, 1982.

[Jon87] B. Jonsson. Modular verification of asynchronous networks. In *Proceedings of the 6th Symposium on the Principles of Distributed Computing*, pages 152–166. Association for Computing Machinery, 1987.

[KK91] N. Klarlund and D. Kozen. Rabin measures and their applications to fairness and automata theory. In *Proceedings of the 6th Symposium on Logic in Computer Science*, pages 256–265, Los Alamitos, CA, 1991. Institute of Electrical and Electronics Engineers.

[Kla90a] N. Klarlund. *Progress Measures and Finite Arguments for Infinite Computations*. PhD thesis, Cornell University, August 1990. Available as Technical Report TR-1153.

[Kla90b] N. Klarlund. Verification conditions for ω-automata and applications to fairness. Technical Report TR-1080, Cornell University, February 1990.

[Kla91] N. Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *Proceedings of the 32nd Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 358–367, Los Alamitos, CA, 1991. Institute of Electrical and Electronics Engineers.

[Kla92a] N. Klarlund. Liminf progress measures. In S. Brookes, M. Main, A. A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 7th International Conference, March 1991*, volume 598 of *Lecture Notes in Computer Science*, pages 477–491, Berlin, Heidelberg, 1992. Springer-Verlag.

[Kla92b] N. Klarlund. Progress measures and stack assertions for fair termination. In *Proceedings of the 11th Symposium on Principles of Distributed Computing, Vancouver*, pages 229–240, Los Alamitos, CA, 1992. Institute of Electrical and Electronics Engineers.

[Kla94a] N. Klarlund. The limit view of infinite computations. In *Proceedings CONCUR '94: Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 351–368, Berlin, Heidelberg, 1994. Springer-Verlag.

[Kla94b] N. Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Journal of Pure and Applied Logic*, 69:243–268, 1994. A preliminary version appeared in *Proceedings of the 7th Symposium on Logic in Computer Science*, 1992.

[LPS81] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: the ethics of concurrent termination. In *Proceedings of the 8th Colloquium on Automata, Language, and Programming*, volume 115 of *Lecture Notes in Computer Science*, pages 264–277, Berlin, Heidelberg, 1981. Springer-Verlag.

[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.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[Mos84] A. W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory, Proceedings of the 5th Symposium*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168, Berlin, Heidelberg, 1984. Springer-Verlag.

[MP84] 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.

[MP87] Z. Manna and A. Pnueli. Specification and verification of concurrent programs by $\forall$-automata. In *Proceedings of the 14th Symposium on Principles of Programming Languages, Munich, West Germany*, pages 1–12. Association for Computing Machinery, 1987.

[OL82] S. Owicki and L. Lamport. Proving liveness of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455–495, 1982.

[Rab69] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[RFG88] R. Rinat, N. Francez, and O. Grumberg. Infinite trees, markings and well-foundedness. *Information and Computation*, 79:131–154, 1988.

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

[Saf88] S. Safra. On complexity of ω-automata. In *Proceedings of the 29th Symposium on Foundations of Computer Science*, pages 319–327. Institute of Electrical and Electronics Engineers, 1988.

[SdRG89] 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.

[Sis87] A. P. Sistla. On using automata in the verification of concurrent programs. Technical report, Computer and Intelligent Systems Laboratory, GTE Laboratories Inc, 1987.

[Var91] M. Vardi. Verification of concurrent programs: The automata-theoretic framework. *Annals of Pure and Applied Logic*, 51:79–98, 1991.

[Var94] M. Vardi. Rank predicates vs. progress measures in concurrent-program verification. Technical Report RJ 9879, IBM Research Division, 1994.