Rabin Measures and Their Applications to Fairness and Automata Theory

Nils Klarlund*
Dexter Kozen**

TR 91-1201
April 1991

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

*The author 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.
**The author was partially supported by NSF grant CCR-8901061.

Rabin Measures and Their Applications to Fairness and Automata Theory

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

Dexter Kozen†
Department of Computer Science
Cornell University
Ithaca, NY 14853

Abstract

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 show how to determine whether a program satisfies a Rabin condition by reasoning about single transitions instead of infinite computations. We introduce a concept, called a *Rabin measure*, which in a precise sense expresses progress for each transition towards satisfaction of the Rabin condition.

When applied to termination problems under fairness constraints, Rabin measures constitute a simpler verification method than previous approaches, which often are syntax-dependent and require recursive applications of proof rules to syntactically transformed programs. Rabin measures also generalize earlier automata-theoretic verification methods. Combined with a result by Safra, our result gives a method for proving that a program satisfies a nondeterministic Büchi automaton specification.

1 Introduction

This paper is concerned with infinite sequences and properties of them 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, properties expressible as *Rabin conditions* [19], which are temporal conditions in a special disjunctive normal form, play a major role. In the theory of fairness, Rabin conditions describe *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.

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 [3,4,5,6,7,8,15,24], temporal logic [16,18], or automata theory [1,2,9,17, 25]. 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.

In this paper we address a fundamental question underlying several of the previous approaches: how can one explain in terms of local conditions that a graph satisfies a Rabin condition? This question is important, because a solution enables verification to take place locally, as opposed to globally, using assertional reasoning about program states and transitions. Our contribution is a concept, called a *Rabin measure*, which mathematically quantifies progress for transitions towards satisfaction of a Rabin condition. Rabin measures obey a relation, called the *Rabin relation*. This relation is used to express closeness to satisfaction of the Rabin condition in the same way a well-founded set is used to express closeness towards program termination.

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 Π^1_1-complete), the result is important for the following reasons:

1) Rabin measures can be used to verify properties expressible by temporal formulas in a certain

*E-mail: klarlund@watson.ibm.com. The author 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.

†E-mail: kozen@cs.cornell.edu. The author was partially supported by NSF grant CCR-8901061.

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 in a satisfactory way.

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 [7]) arise simply and naturally in this context and can be explained completely in these terms.

2) Our results extend [12], which describes a syntax-independent verification method for termination under *strong fairness*. Previous methods for fair termination rely on program transformations and thus depend on properties of the underlying programming language [7]. In this paper we show that for the *general fairness constraints* proposed in [6], programs can by annotated with Rabin measures that imply fair termination directly.

3) We obtain verification methods for specifications given by deterministic *Rabin automata* and nondeterministic *Büchi automata*. These methods extend earlier work based on Boolean combinations of deterministic automata [2] and $\forall$-*automata*, whose nondeterminism is interpreted as "and" [17].

2 Previous Work

Alpern and Schneider [1,2] used deterministic Büchi automata as a method of specification. They obtained a method of verification for nondeterministic Büchi automata using the fact that any such automaton can be converted to a Boolean combination in conjunctive normal form of deterministic Büchi automata.[1] Vardi [25] proposed a very general approach to verification where specifications are 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. By contrast, our results apply to the more complicated Rabin condition directly and explain helpful directions for such a condition. Moreover, we show how our results are easily carried over to automata with a Rabin acceptance condition.

Inspired by the ideas in [25], Manna and Pnueli gave verification conditions for $\forall$-automata [17], which are expressively equivalent to Büchi automata, although there is no known easy conversion of a Büchi automaton into an $\forall$-automaton. Sistla [23] 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.

For temporal logic specifications, assertional proof methods such as [16,18] are limited to quite simple temporal formulas that express properties like "leads-to" and "always." A general approach to verifying with finite temporal formulas was proposed in [20]. 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.

In the area of fairness, complete verification methods for termination were given in [6,7,8,15,14,24]. These methods are based on *helpful directions* and the iterative use of proof rules applied to syntactically transformed programs. In [7, 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.

The methods of *explicit schedulers* developed in [3,4,5] 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 [7].

Progress measures were introduced in [13] as a generic concept for quantifying how each step of a program contributes to bringing a computation closer to its specification. There it is shown that progress measures exist for a variety of program specifications, including those involving nondeterminism, fairness, and infinitary temporal logics. The paper [12] is concerned with a restricted form of Rabin measures called *fair termination measures* and the completeness proof given

[1] Probably the simplest conversion is to apply Safra's determinization method and write the Rabin acceptance condition in conjunctive normal form.

applies only to graphs that are trees. A completeness proof for arbitrary graphs requires the more sophisticated argument presented in this paper.

The verification method in [11] 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.

3 Rabin Conditions

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 write $v_0, v_1, \ldots \models (R, I)$ if there are infinitely many k such that $v_k \in R$ and only finitely many k such that $v_k \in I$.

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 write $v_0, v_1, \ldots \models \mathcal{C}$ if for some χ, $v_0, v_1, \ldots \models (R_\chi, I_\chi)$. We say that a graph $G = (V, E)$ satisfies a Rabin condition $\mathcal{C}$ on V and write $G \models \mathcal{C}$ if every infinite path $v_0, v_1, \ldots$ in G satisfies $\mathcal{C}$.

4 Pointer Trees

Rabin measures are based on pointer trees, also called direction trees, which are introduced in this section. 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. We visualize pointer trees as growing upwards; see Figure 1, where children are depicted from left to right in descending order. Any sequence of pointers $t^1, t^2, \ldots$ (finite or infinite) denotes a *path* $\langle\rangle, \langle t^1 \rangle, \langle t^1, t^2 \rangle, \ldots$

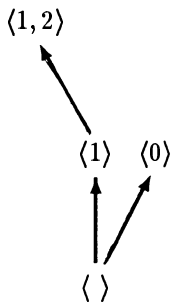

Figure 1: A pointer tree.

(finite or infinite) in T, provided each $\langle t^1, \ldots, t^\ell \rangle \in T$. The *level* $|t|$ of a node $t = \langle t^1, \ldots, t^\ell \rangle$ is the number ℓ; the level of $\langle\rangle$ is 0. The *prefix up to level* λ of $t = \langle t^1, \ldots, t^n \rangle$ is $\langle t^1, \ldots, t^{\min\{n,\lambda\}} \rangle$, denoted $t \downarrow \lambda$. The *height* of T is the maximum node level (if it exists). T is *finite-path* if there are no infinite paths in T. A *common ancestor* of nodes t and t' is a node $\tilde{t}$ that is a ancestor of both of t and of t'. The common ancestor at the highest level is the *highest common ancestor*. 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 any infinite sequence of nodes in a finite-path pointer tree T. Then there is a node t that is almost always a common ancestor and is infinitely often the highest common ancestor of t_k, t_{k+1}.

Proof 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 any t, t' in T' are eventually ancestors of t_k, and the ancestors of any node are linearly ordered. Since T contains only finite paths, T' has a unique maximal element, which satisfies the desired conditions. □

Definition 1 (Kleene-Brouwer Ordering) The ordering $\succ$ on T is defined by: $t \succ t'$ if there is a λ such that $t \downarrow \lambda = t' \downarrow \lambda$ and either $\lambda = |t| < |t'|$ or $\lambda < |t|, |t'|$ and $t^{\lambda+1} > t'^{\lambda+1}$. The ordering $\succeq$ is defined as $t \succeq t'$ 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). This is a total order on T.

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

Proof See [21]. □

Rabin measures are based on colored pointer trees, which are defined by:

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 $\mathrm{dom}\xi$.

5 Rabin Relation and Measure

We can now formulate the Rabin relation:

Definition 3 (Rabin Relation) For a colored pointer tree (T, ξ) and a Rabin condition $\mathcal{C}=\{(R_\chi, I_\chi) \mid \chi \in X\}$ on V, the *Rabin relation* $\triangleright \subseteq T \times V \times T$ is defined as:

$t \overset{v}{\triangleright} t'$ if

(R) $t \succ t'$ or $v \in R_{\xi(\hat{t})}$, where $\hat{t}$ is the highest common ancestor of t and t', and

(I) $v \notin I_{\xi(\tilde{t})}$ for each common ancestor $\tilde{t}$ of t and t'

Note that (I) implies that if $t \overset{v}{\triangleright} t'$ then every common ancestor must be colored by ξ. Intuitively, condition (R) enforces progress towards a reconfirming state of the Rabin pair designated by the highest common ancestor $\hat{t}$. Thus, $\hat{t}$ corresponds to the concept of "helpful direction" in previous work. Condition (I) enforces that for Rabin pairs designated by the common ancestors $\tilde{t}$, progress is not annulled by invalidating states.

When T is finite-path, the Rabin relation ensures in the limit that $\mathcal{C}$ is satisfied:

Lemma 3 (Rabin Relation Lemma) Let T be finite-path. If $t_0 \overset{v_1}{\triangleright} t_1 \overset{v_2}{\triangleright} \cdots$, then $v_0, v_1, \ldots \models \mathcal{C}$.

Proof By the Highest Common Ancestor Lemma, 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 $\chi = \xi(t)$ and let $\lambda = |t|$. We show that $v_0, v_1, \ldots \models (R_\chi, I_\chi)$.

First, by (a) and (I) it holds almost always that $v_k \notin I_\chi$.

Second, it holds infinitely often that $v_k \in R_\chi$; for suppose that the contrary is true, i.e. that for some K and for all $k \geq K$, $v_k \notin R_\chi$. Then we can by (a) assume that K is such that for all $k \geq K$, t is a common ancestor of t_k and t_{k+1}. Now let $k \geq K$. If t is the highest common ancestor of t_k and t_{k+1}, then $t_k \succ t_{k+1}$ by (R) and our supposition that $v_k \notin R_\chi$; in fact, $t_k \downarrow \lambda+1 \succ t_{k+1} \downarrow \lambda+1$. Otherwise, t is a common ancestor, but the not highest, of t_k and t_{k+1}; thus $|t_k|, |t_{k+1}| > |t|$ and $t_k \downarrow \lambda+1 = t_{k+1} \downarrow \lambda+1$. We conclude that $t_K \downarrow \lambda+1 \succeq t_{K+1} \downarrow \lambda+1 \succeq \cdots$ and that due to (b) infinitely many inequalities are strict. This contradicts the Kleene-Brouwer Ordering Lemma. Thus it holds infinitely often that $v_k \in R_\chi$. □

It is not hard to see that the Rabin Relation Lemma still holds if "the highest" is replaced by "some" in (R) of Definition 3, but for simplicity we stick to the original definition.

Definition 4 A *Rabin measure* $(\mu, (T, \xi))$ of $(G, \mathcal{C})$, where $G = (V, E)$ and $\mathcal{C} = \{(R_\chi, I_\chi) \mid \chi \in X\}$, is a colored finite-path pointer tree (T, ξ) and a mapping $\mu : V \to T$ respecting the edge relation; i.e. $(u, v) \in E$ implies $\mu(u) \overset{v}{\triangleright} \mu(v)$.

The main result of this article is:

Theorem 1 $G \models \mathcal{C}$ iff $(G, \mathcal{C})$ has a Rabin measure (of height $\leq |\mathcal{C}|$).

Proof ($\Leftarrow$) Let $v_0, v_1, \ldots$ be an infinite path in G. Then $\mu(v_0) \overset{v_1}{\triangleright} \mu(v_1) \overset{v_2}{\triangleright} \cdots$ Thus by Lemma 3, $v_0, v_1, \ldots \models \mathcal{C}$.

($\Rightarrow$) This is the subject of Section 6. □

6 Construction of Rabin Measures

We show how to construct a Rabin measure of $(G, \mathcal{C})$, 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

We need some definitions before stating a key lemma. A *color set assignment* CS is a map $V \to \mathcal{P}C$,[2] where C is a countable set of colors; CS associates a non-empty set of *enabled* colors $CS(v) \subseteq C$ to each vertex $v \in V$. A set $W \subseteq V$ is *χ-enabled* if for all $v \in W$, $\chi \in CS(v)$. An infinite path $v_0, v_1, \ldots$ is *eventually χ-enabled* if some suffix $v_k, v_{k+1}, \ldots$ is *χ-enabled*. A color set assignment is *permissible* if every infinite path is eventually χ-enabled for some χ.

[2] $\mathcal{P}C$ denotes the class of subsets of C.

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 4 Let $G = (V, E)$ be a countable graph. If $CS : V \to \mathcal{P}C$ 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 non-empty 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 4) For every finite path u in G, define $B_u =$

$$\{u \cdot w \mid u \cdot w \text{ is a path (finite or infinite) in } G\}$$

Let $M = \{w \mid w \text{ is a path (finite or infinite) in } G\}$. The B_u's form a subbasis for a topology over M, where the open sets are unions of B_u's. It can be seen that M is a complete metric space.[3]

For u a finite path 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 $F_{u,\chi}$, where χ is arbitrary, 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 $M = \bigcup_{u,\chi} F_{u,\chi}$.

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. □

6.2 Colorings

Lemma 4 can be applied transfinitely to a graph to yield a stronger result. We need a few definitions. A *coloring* c of a set V is a total mapping $c : V \to C$. A coloring c *obeys* a color set assignment CS 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, $\chi = c(v_i)$. 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 5 Let $G = (V, E)$ be a graph. If CS is a permissible color set assignment, then there is an eventually stable coloring $c : V \to C$ obeying CS and a partition $\mathcal{S} = \{W_\theta\}_{\theta < \gamma}$ of V, where γ is a countable ordinal, such that

(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 We apply Lemma 4 transfinitely. More precisely, Lemma 4 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 4 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.

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. □

6.3 Proof of Theorem 1 "⇒"

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 2. 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.

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 $\widehat{W} = W(t) \backslash R_\chi$ is the set of states in $W(t)$ that are not χ-reconfirming.

[3] The metric can be given as for example:

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

AssignRabin(t, Y, χ):

1. $\xi(t) := \chi$.
   $\widehat{W} := W(t) \backslash R_\chi$.
2. Use Lemma 5 on $G|\widehat{W}$ with color set assignment $CS^{\overline{Y \cup \{\chi\}}}$ (explained in the text) to obtain a coloring c and a partition $\mathcal{S} = \{W_\theta | \theta < \gamma\}$ of $\widehat{W}$.
3. For each class W_θ of $\mathcal{S}$
   (a) $T := T \cup \{t \cdot \langle\theta\rangle\}$.
   (b) $W(t \cdot \langle\theta\rangle) := W_\theta$.
4. For each χ'-colored class W_θ of $\mathcal{S}$, where $\chi' \in X$:
   AssignRabin$(t \cdot \langle\theta\rangle, Y \cup \{\chi\}, \chi')$.

Figure 2: Algorithm **AssignRabin**.

In Step 2 Lemma 5 is used to obtain a partition $\mathcal{S} = \{W_\theta | \theta < \gamma\}$ 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 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 $\{\perp_v\}$, where $\perp_v$ is a distinct dummy color, different from any color defined elsewhere. Thus the set C of all colors is $X \cup \{\perp_v | v \in V\}$.

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 labelled W_θ.

Finally in Step 4 descendants of each child not assigned a dummy color are constructed by further applications of **AssignRabin**.

To explain the construction of μ and to prove that Lemma 5 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 $|Y| = |t|$;
- $\chi \in X \backslash Y$;
- $W(t)$ is Y-nonreconfirming; and
- $W(t)$ is $(Y \cup \{\chi\})$-noninvalidating.

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

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 $|Y| = |t|$, $\chi \in X \backslash Y$, and $W(t)$ is $(Y \cup \{\chi\})$-noninvalidating and Y-nonreconfirming.

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, $\chi' \in X \backslash (Y \cup \{\chi\})$. 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 $W_\theta \subseteq \widehat{W} = W \backslash R_\chi$), it follows that $W(t \cdot \langle\theta\rangle) = W_\theta$ is $(Y \cup \{\chi\})$-nonreconfirming. □

To see that Lemma 5 is applicable in Step 2 of **AssignRabin**, we prove:

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

Proof 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. Let now $v_0, v_1, \ldots$ 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 $G \models \mathcal{C}$, $v_0, v_1, \ldots \models (R_{\hat{\chi}}, I_{\hat{\chi}})$ for some $\hat{\chi} \in X \backslash (Y \cup \{\chi\})$. 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 $\widehat{W}$. □

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 any $t \in T$ with $|t| < |X|$,

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

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

(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 (a) holds for $t = \langle\rangle$, because $W(\langle\rangle) = V$ and $\widehat{W}$ formed in `AssignRabin` is $W(\langle\rangle) \backslash R_{\xi(\langle\rangle)} = V \backslash R_0$; thus Lemma 5 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 $W(\langle\rangle) \backslash R_{\xi(\langle\rangle)}$. Similarly, by induction it can been seen that (a) holds for all $t \in T$ with $|t| < |X|$.

Part (b) follows from (a). □

Using Claim 3(b), we define $\mu : V \to T$ by $\mu(v) = \langle t^1, \ldots, t^n \rangle$. Now let $(u, v) \in E$ and let us prove that $\mu(u) \overset{v}{\rhd} \mu(v)$.

To prove (I), we note that if $\tilde{t}$ is a common ancestor of $\mu(u)$ and $\mu(v)$, then $\xi(\tilde{t})$ is defined, because otherwise $W(\tilde{t})$, which contains both u and v, consists of a node $u = v$ colored with a dummy color—this contradicts Claim 2 that for the path $u, u, \ldots$ the color assignment is enabled for some $\widehat{\chi} \in X$. By Claim 1 and by the definition of $\xi(\tilde{t})$ in Step 1, $W(\tilde{t})$ is $\xi(\tilde{t})$-noninvalidating. Since $v \in W(\tilde{t})$ by the definition of $\mu(v)$, it follows that $v \notin I_{\xi(\tilde{t})}$.

To prove (R), let $\hat{t}$ be the highest common ancestor of $\mu(u)$ and $\mu(v)$. If $v \in R_{\xi(\hat{t})}$, then (R) is trivially satisfied, so assume that $v \notin R_{\xi(\hat{t})}$. Then $|\mu(v)| > |\hat{t}|$ by Claim 3(a). Consider the application `AssignRabin`$(\hat{t}, Y, \xi(t))$.

If $u \in R_{\xi(\hat{t})}$, then $\mu(u) = \hat{t}$ because $u \notin \widehat{W} = W(t) \backslash R_{\xi(\hat{t})}$. Thus $\mu(u) = \hat{t} \succ \mu(v)$, because $|\mu(v)| > |\hat{t}|$. Otherwise, if $u \notin R_{\xi(\hat{t})}$, then by Claim 3(a), 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 5, $\theta > \theta'$, because $(u, v) \in E$. Therefore, $\mu(u) \succ \mu(v)$. Thus in all cases (R) holds. □

```
*[ a: x=0              → y:=y+1
   b: x=0 ∧ even(y)    → x:=1
   c: x≠0 ∧ y≠0        → y:=y-1
   d: x≠0 ∧ y≠0        → z := z + 1]
```

Figure 3: The program P_{ex}.

7 Application to Fairness

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 [6] or [7, 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. A *computation* $p_0, p_1, \ldots$ of a program P is a sequence of program states such that p_0 is an initial state and each p_{i+1} is related to p_i by an atomic transition. The *computation graph* $G(P)$ of P is the graph whose vertices are states of computations of P and whose edges correspond to the atomic transactions of P. 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 P *$\mathcal{C}$-fairly terminates* if every infinite computation of P is unfair, i.e. if $G(P) \models \mathcal{C}$. Thus to show fair termination of P, we can use Theorem 1.

Example

Program P_{ex} shown in Figure 3 is taken from [8] (and can also be found in [7]). The variables `x`, `y`, and `z` take on non-negative integer values. Program P_{ex} terminates under the assumption of *strong fairness:* for any 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), $\ell = a, b, c, d$, denotes that command ℓ is unfairly executed. Thus ϕ_ℓ is the guard of ℓ and ψ_ℓ is a predicate denoting that ℓ was the last

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 towards termination.

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

$$\{\langle\rangle, \langle 0\rangle, \langle 1\rangle, \ldots, \langle\omega\rangle, \langle\omega, 0\rangle, \langle\omega, 1\rangle\}$$

and the coloring ξ is defined by

$$\xi(\langle\rangle) = t,\ \xi(\langle\omega\rangle) = b, \text{and } \xi(\langle k\rangle) = c, k < \omega.$$

The mapping μ is defined by

$$(\star)\quad \mu(x, y) = \begin{cases} \langle\omega, y + 1 \bmod 2\rangle & \text{if } x = 0 \\ \langle y\rangle & \text{if } x \neq 0 \end{cases}$$

where x and y denote the values of variables $\mathbf{x}$ and $\mathbf{y}$, respectively.

In general the progress measure can be described by means of *program assertions*, which state the value of the progress measure as a function of the program variables. For example, P_{ex} could be annotated with the formula $(\star)$, thus indicating the value of the progress measure before execution of the loop. The coloring ξ can also be described using assertional techniques, see [12].

For the program P_{ex}, it can be verified that with each iteration of the loop, the value of μ changes according to (R) and (I). For example, consider the case when $\mathbf{x} = 0$, $\mathbf{y}$ is even, and ℓ_a is executed. Then μ changes from $\langle\omega, 1\rangle$ to $\langle\omega, 0\rangle$ and thus (R) holds, since $\langle\omega, 1\rangle \succ \langle\omega, 0\rangle$; also (I) is satisfied: the common ancestors are $\{\langle\rangle, \langle\omega\rangle\}$ and $\psi_{\xi(\langle\rangle)} = \psi_t$ is always false and $\psi_{\xi(\langle\omega\rangle)} = \psi_b$ is false, since ℓ_a is executed.

The termination proof in [7,8] of the program P_{ex} involves reasoning not only about the original program, but also about two transformed programs.

8 Automata-Theoretic Applications

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.

In this section we show that 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 [22], this yields a verification method for specifications given as nondeterministic finite-state Büchi automata.

8.1 Verification with Deterministic Rabin Automata

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—or is a *behavior of* 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.

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 $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots$ 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.

We can now formulate verification conditions for showing that $L(A_P) \subseteq L(A_S)$, 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. The verification conditions are formulated in terms of a *variant* $\mathcal{I}$, which relates the program and specification automata together with progress information for $\mathcal{C}$; formally, $\mathcal{I}$ is a relation on $V_P \times V_S \times T$, where (T, ξ) is a colored pointer tree of height $|\mathcal{C}|$. The verification conditions for $\mathcal{I}$ are:

$$(\mathrm{V1_R})\quad p \in V_P^0 \ \Rightarrow\ \exists t : \mathcal{I}(p, s^0, t)$$

$$(\mathrm{V2_R})\quad p \xrightarrow{e}_P p' \ \wedge\ \mathcal{I}(p, s, t) \ \Rightarrow$$

$$\exists s', t' : s \xrightarrow{e}_S s' \ \wedge\ t \overset{s'}{\triangleright} t' \ \wedge\ \mathcal{I}(p', s', t')$$

For an automata-theoretic formulation of Theorem 1, we define a a state v to be *reachable* over $e_0, \ldots, e_n$ if there is a partial run $v_0 \xrightarrow{e_0} \cdots \xrightarrow{e_n} v_{n+1}$ with $v = v_{n+1}$. The set of states reachable over $u = e_0, \ldots, e_n$ is denoted $\mathcal{R}_A(u)$. A *dead-end* state of an automaton is a reachable state that does not occur in any run.

Corollary 1 (of Theorem 1) Let A_P be an automaton with no dead-end states and let A_S be a deterministic Rabin automaton. Then $L(A_P) \subseteq L(A_S)$ iff there is a variant $\mathcal{I}$ satisfying $(V1_R)$ and $(V2_R)$.

Proof "$\Leftarrow$": Let $p_0 \xrightarrow{e_0}_P p_1 \xrightarrow{e_1} \cdots$ be a run of A_P. By $(\mathrm{V1_R})$ and $(\mathrm{V2_R})$, there are $s_0, s_1, \ldots$ and $t_0, t_1, \ldots$ such that $s_0 = s^0$, $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$, and $t_0 \overset{s_1}{\triangleright} t_1 \overset{s_2}{\triangleright} \cdots$. It follows that $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$ is a run of A_S, which is accepting by Lemma 3.

"$\Rightarrow$" Assume that $L(A_P) \subseteq L(A_S)$. Construct the joint graph $G = (V, E)$ defined by

$$V = \{(p,s) \mid p \in V_P \wedge s \in V_S \wedge \exists u \in \Sigma^* : (p \in \mathcal{R}_{A_P}(u) \wedge s \in \mathcal{R}_{A_S}(u))\}$$
$$E = \{((p,s),(p',s') \mid \exists e \in \Sigma : p \xrightarrow{e}_P p' \wedge s \xrightarrow{e}_S s'\}$$

Note that the assumption that A_P has no dead-end states implies that for any joint state (p,s), if there are e, p' such that $p \xrightarrow{e}_P p'$, then there is an s' such that $s \xrightarrow{e}_S s'$.

From the Rabin condition $\mathcal{C}$ on V_S, define a Rabin condition, also called $\mathcal{C}$, on G in the natural way. From the assumption $L(A_P) \subseteq L(A_S)$, it follows that $G \models \mathcal{C}$. Thus by Theorem 1 there is a Rabin measure $(\mu, (T, \xi))$ of G. Define $\mathcal{I}$ by

$$\mathcal{I}(p,s,t) \text{ iff } (p,s) \in V \wedge \mu(p,s) = t.$$

Then it is not hard to see that both $(\mathrm{V1_R})$ and $(\mathrm{V2_R})$ are fulfilled. □

8.2 Verification with Nondeterministic Büchi Automata

A Büchi automaton $A = (\Sigma, V, \rightarrow, V^0, V^F)$ is a nondeterministic automaton $(\Sigma, V, \rightarrow, V^0)$ with a set $V^F \subseteq V$ of *accepting states*. A run $v_0, v_1, \ldots$ of A over a behavior $e_0, e_1, \ldots$ is *accepting* if there is a state in V^F that occurs infinitely often in $v_0, v_1, \ldots$ The language $L(A)$ *accepted by* A is the set of behaviors that allow an accepting run.

Using Safra's result [22], we define a method of verification for finite-state Büchi automata. Safra showed how to construct a deterministic Rabin automaton $\mathcal{S}(A_S)$ with $2^{O(n \log n)}$ states and $O(n)$ pairs given a nondeterministic Büchi automaton A_S with n states. When applying our method of verification to $\mathcal{S}(A_S)$, we see that the variant becomes a relation $\mathcal{I}(p,r,t)$ where r is a state of $\mathcal{S}(A_S)$ and t is a node in a colored pointer tree of height n.

Corollary 2 (of Theorem 1) The verification conditions $(\mathrm{V1_R})$ and $(\mathrm{V2_R})$ applied to $\mathcal{S}(A_S)$ are sound and complete for verifying that $L(A_P) \subseteq L(A_S)$, where A_P is a nondeterministic automaton with no dead-end states and A_S is a finite-state Büchi automaton.

For further applications see [10], where the method for Rabin automata is extended to the $\forall$-automata of [17] and applied to simplify the method of [2].

References

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

[2] 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.

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

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

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

[8] 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.

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

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

[11] N. Klarlund. Liminf progress measures. *Mathematical Foundations of Programming Semantics 1991*. To appear in LNCS, 1991.

[12] N. Klarlund. Progress measures for fair termination. Submitted for publication, 1991.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

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

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

[22] S. Safra. On complexity of ω-automata. In *Proc. Foundations of Computer Science.* IEEE, 1988.

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

[24] 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.

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