Proving Nondeterministically Specified Safety Properties Using Progress Measures

Nils Klarlund*
Fred B. Schneider**

TR 91-1204
(replaces TR 90-1167)
May 1991

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

*Supported by grants from the University of Aarhus, Denmark, the Danish Research Academy, and the Thanks to Scandinavia Foundation Inc. Current address: IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598.
**Supported in part by the Office of Naval Research under contract N00014-86-K-0092, the National Science Foundation under Grant No. CCR-8701103, and Digital Equipment Corporation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not reflect the views of these agencies. Current address: Dept. of Computer Science, Upson Hall, Cornell University, Ithaca, NY 14853.

Proving Nondeterministically Specified Safety Properties Using Progress Measures

Nils Klarlund* and Fred B. Schneider†

April, 1991

Abstract Using the notion of progress measures, we discuss verification methods for proving that a program satisfies a property specified by an automaton having finite nondeterminism. Such automata can express any safety property. Previous methods, which can be derived from the method presented here, either rely on transforming the program or are not complete. In contrast, our ND progress measures describe a homomorphism from the unaltered program to a canonical specification automaton and constitute a complete verification method. The canonical specification automaton is obtained from the classical subset construction and a new subset construction, called historization.

1 Introduction

Nondeterministic automata are a convenient mathematical abstraction for programs and specifications that define infinite sequences of events [Arn83, Par81, Sis89b, Var87]. A program is modelled as an automaton A_P, called the *program automaton*, which accepts a language $L(A_P)$ of infinite behaviors (words); a specification is modelled as an automaton A_S, called a *specification automaton*, which accepts the language $L(A_S)$. Both automata may be infinite-state. *A_P satisfies A_S* if every behavior of A_P is allowed by A_S; that is, if the containment $L(A_P) \subseteq L(A_S)$

*Supported by grants from the University of Aarhus, Denmark, the Danish Research Academy, and the Thanks to Scandinavia Foundation Inc. Current address: IBM T.J. Watson Research Center, PO BOX 704, Yorktown Heights, NY 10598

†Supported in part by the Office of Naval Research under contract N00014-86-K-0092, the National Science Foundation under Grant No. CCR-8701103, and Digital Equipment Corporation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not reflect the views of these agencies. Current address: Department of Computer Science, Upson Hall, Cornell University, Ithaca, New York 14853

holds. The *verification problem* is to find a method based on reasoning about states and transitions of A_P and A_S for establishing that A_P satisfies A_S.

In this paper we address the verification problem using the notion of *progress measure*, introduced in [Kla90]. A progress measure μ for establishing $L(A_P) \subseteq L(A_S)$ quantifies how a behavior of A_P converges towards a behavior accepted by A_S. More precisely, with each state p of A_P is associated information $\mu(p)$ about the progress of A_S, and the convergence is expressed in a mathematical, but non-numeric sense, by a *progress relation* $\rhd_S$, which depends on only A_S:

(VC) For any transition of A_P from state p to p' emitting symbol e, $\mu(p) \overset{e}{\rhd} \mu(p')$ holds.

(VC) is a verification condition since it is a local requirement about states and transitions. (In addition, some technical conditions relating the initial states of A_P and A_S must hold.)

A verification method based on progress measures for proving that A_P satisfies $L(A_S)$ is *sound* if the existence of a progress measure implies that $L(A_P) \subseteq L(A_S)$ holds. In that case, whenever $p_0, p_1, \ldots$ is a run over a behavior $\langle e_0, e_1, \ldots \rangle$ in $L(A_P)$,[1] the $\rhd$-related sequence $\mu(p_0) \overset{e_0}{\rhd}_S \mu(p_1) \overset{e_1}{\rhd}_S \cdots$ (whose existence is guaranteed by the verification condition above) gives rise to a run of A_S over $e_0, e_1, \ldots$ A method is *complete* if such a μ is guaranteed to exist whenever A_P satisfies A_S.

No sensible method based on progress measures can be sound and complete for specifications given by arbitrary nondeterministic automata [Sis89b]. Therefore, we restrict attention to specification automata that are infinite-state but whose nondeterminism is finite; such automata are called *safety automata*, because they express the class of safety properties[2] [AL88, Kla90]. Other papers, such as [AS89, CBK90, KK91], address the verification problem for (usually deterministic) automata that can also express liveness properties.

A particularly simple progress measure is the *automaton homomorphism*, known as a *refinement mapping*. For this kind of progress measure, the progress relation $\rhd_S$ is just the transition relation of A_S. Thus the verification condition expresses that μ maps any transition of A_P to a transition of A_S. (In addition, μ maps initial states of A_P to initial states of A_S.) Regrettably, there are simple situations where the inclusion $L(A_P) \subseteq L(A_S)$ holds but this cannot be proved by a homomorphism.

[1]A run of an automaton is a sequence of automaton states corresponding to a behavior accepted by that automaton.

[2]Informally, a *safety property* is one stating that some "bad thing" does not happen. Formally, a safety property is a closed set [AS85].

Abadi and Lamport [AL88] showed that there are two problems behind the inadequacy of homomorphisms. One problem, which we call the *prophecy problem*, stems from the nondeterminism of A_S and suggests that $\mu(p)$ must designate several states of A_S at once.[3] The other problem, which we call the *history problem*, is more intricate. Intuitively, it arises when A_P may visit the same state p in different ways, each giving rise to a different corresponding state of A_S; thus this problem also suggests that $\mu(p)$ must designate several states of A_S at once. Abadi and Lamport showed that these problems can be overcome by adding *prophecy* and *history variables*, respectively, to the program automaton so that a homomorphism μ can be constructed. The result is a sound and complete verification method, but at the expense of transforming the program automaton.

Often it is desirable to understand how each step executed by the unaltered program contributes to bringing the resulting computation closer to satisfying the specification. Therefore, some authors have proposed more powerful kinds of progress measures [Mer90, Lam83, LT87, Par81, Sis89a, Sta88], which we here call *prophecy measures* and *history measures*. These progress measures map a program state to a *set* of specification sets and can sometimes eliminate the need to transform the program automaton. A prophecy measure, for example, corresponds to a homomorphism from A_P to the deterministic automaton $\mathcal{D}A_S$ obtained by applying the classic subset construction [RS59], here denoted by $\mathcal{D}$, to the specification automaton. If prophecy measures are used in the method of Abadi and Lamport, the need for prophecy variables disappears. Yet, as we prove in this paper, no progress measure based on mappings to sets of specification states can form a complete verification method, unless transformations are used.

This paper gives a new progress measure, called the *ND measure*, and a new kind of subset construction, which we call *historization* and denote by $\mathcal{H}$. The $\mathcal{H}$ construction allows us to explain history measures in terms of homomorphisms from A_P to $\mathcal{H}A_S$. Similarly, ND measures correspond to homomorphisms from A_P to $\mathcal{HD}A_S$, which is the automaton obtained by first performing determinization and then performing historization. The significance of $\mathcal{HD}A_S$ is that there is always a homomorphism from A_P to $\mathcal{HD}A_S$ when A_P satisfies A_S. The need for prophecy variables disappears because of determinization; the need for history variables disappears because of historization, and a homomorphism can be constructed from A_P to $\mathcal{HD}A_S$. Therefore, $\mathcal{HD}A_S$ is a *canonical specification automaton*. Moreover, as an alternative to constructing all of $\mathcal{HD}A_S$ at once and

[3] The notions of *prophecy* and *history* are derived from [AL88].

then using a homomorphism, the ND progress measure allows the description of only parts of $\mathcal{HDA}_S$. This approach to verification may be important in practice just as Abadi and Lamport showed that with their method, it is not always necessary to add variables to A_P that describe all the information inherent in their completeness proof.

The remainder of this paper is organized as follows. Section 2 contains definitions and describes basic properties of infinite-state automata. In Section 3 we consider some natural approaches to the verification problem, and we prove that the resulting methods are incomplete. Section 4 discusses completeness results for the methods of homomorphisms and prophecy progress measures. Section 5 describes the history and ND measures and their relationship to historization. Then Section 6 explains in detail how our results are related to existing verification methods from the literature. Section 7 relates our approach to recursion theory. Section 8 contains a summary.

2 Definitions and Basic Properties

Let Σ be a fixed countable alphabet of symbols called *events* (representing actions, communications, or observable parts of states). A *behavior* is a sequence (infinite if not otherwise stated) $\langle e_0, e_1, \dots \rangle$ of events. Σ^ω is the set of behaviors and Σ^* is the set of finite behaviors. We denote the concatenation of sequences u and w, where u is finite, by $u \cdot w$.

Let V be a set (countable if not otherwise stated) of *states*. A *transition relation on* V is a relation $\rightarrow \subseteq V \times \Sigma \times V$, where a *transition* $(v, e, v') \in \rightarrow$ is denoted $v \xrightarrow{e} v'$. An *automaton* $A = (\Sigma, V, \rightarrow, V^0)$ consists of an alphabet Σ, a state space V, a transition relation $\rightarrow$ on V, and a set of *initial states* $V^0 \subseteq V$.

For $v, v' \in V$ and $u = \langle e_0, \dots, e_n \rangle \in \Sigma^*$, $n \geq -1$, denote by $v \xrightarrow{u} v'$ that there exist $v_0, \dots, v_{n+1} \in V$ such that $v_0 \xrightarrow{e_0} \cdots \xrightarrow{e_n} v_{n+1}$, where $v = v_0$, $v' = v_{n+1}$; if in addition $v_0 \in V^0$, then v' is *reachable* over u. The set $L_*(A)$ of *finite behaviors of* A consists of all $u \in \Sigma^*$ such that some state v is reachable over u. In the infinite case, $v \xrightarrow{w}$ means that there exist $v_0, v_1, \dots$ such that $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_2} \cdots$, where $v = v_0$ and $w = \langle e_0, e_1, \dots \rangle \in \Sigma^\omega$; if in addition $v_0 \in V^0$, then $v_0, v_1, \dots$ is a *run of* A *over* w and we say that w is *accepted by* A or is a *behavior of* A. $L(A)$ is the set of behaviors of A. The *language* or *property* $L(A)$ *accepted by* A is the set of infinite behaviors of A.

The set of states reachable over $u \in \Sigma^*$ is denoted $\mathcal{R}_A(u)$. Note that $\mathcal{R}_A(\langle\rangle) = V^0$. A transition relation $\rightarrow$ has *finite nondeterminism* if for all $e \in \Sigma$ and all

$v \in V$, the set $\{v' \mid v \xrightarrow{e} v'\}$ is finite. Automaton $A = (\Sigma, V, \rightarrow, V^0)$ is a *safety automaton* if V^0 is finite and $\rightarrow$ has finite nondeterminism. And, if V^0 and all sets $\{v' \mid v \xrightarrow{e} v'\}$ have at most one element, then A is *deterministic*. Observe that if A is deterministic, then a *canonical mapping* $h_A : L_*(A) \rightarrow V$ is defined by $h_A(u) = v$, where $\{v\} = \mathcal{R}_A(u)$. It is well-known [AL88, Arn83, Kla90] that safety automata define the class of safety properties:

Proposition 1 The following are equivalent:

(a) S is a safety property.

(b) $S = L(A)$ for some safety automaton A.

(c) $S = L(A)$ for some deterministic automaton A.

Automaton A is *complete* if $V \neq \emptyset$ and every state $v \in V$ appears in some run. Note that a complete automaton accepts a non-empty language. Moreover, from every automaton $A = (\Sigma, V, \rightarrow, V^0)$ such that $L(A) \neq \emptyset$, it is possible to obtain a complete automaton A' such that $L(A') = L(A)$ by deleting from V those states that do not appear in any run. This procedure, however, is not computable, since it requires deciding whether there is an infinite path from a node in a graph—something that is Σ_1^1-complete for countable recursively represented graphs [Rog67]. In practice, this incomputatibility is usually not a problem, since predicates often can be used to describe the set of states occurring in infinite computations.

If for all v in V there is at most one finite behavior u such that $v \in \mathcal{R}_A(u)$, then A is *historical*; the intuition is that each state corresponds to at most one finite behavior or *history* leading up to that state.

In what follows, we consider a program automaton $A_P = (\Sigma, V_P, \rightarrow_P, V_P^0)$ and a specification automaton $A_S = (\Sigma, V_S, \rightarrow_S, V_0^S)$.

Definition 1 A *homomorphism* or *refinement mapping* μ for (A_P, A_S) is a mapping $\mu : V_P \rightarrow V_S$ such that the following verification condtions are satisfied:

(REμ1) $\quad p \in V_P^0 \;\Rightarrow\; \mu(p) \in V_S^0$

(REμ2) $\quad p \xrightarrow{e}_P p' \;\Rightarrow\; \mu(p) \xrightarrow{e}_S \mu(p')$

(REμ1) stipulates that μ maps initial states of A_P to initial states of A_S, and (REμ2) stipulates that for every transition $p \xrightarrow{e}_P p'$ of A_P, there is a transition $\mu(p) \xrightarrow{e}_S \mu(p')$ of A_S.

Proposition 2 If there is a homomorphism μ for (A_P, A_S), then $L(A_S) \subseteq L(A_S)$.

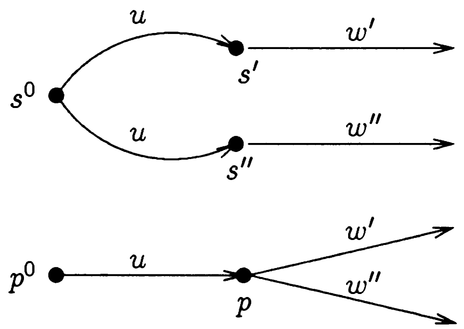
Proof Given a homomorphism μ and a behavior $\langle e_0, e_1, \ldots \rangle$ of A_P; then there is a run $p_0 \xrightarrow{e_0}_P p_1 \xrightarrow{e_1}_P \cdots$ of A_P, and from (REμ1) and (REμ2) it follows that $\mu(p_0) \xrightarrow{e_0}_S \mu(p_1) \xrightarrow{e_1}_S \cdots$ is a run of A_S, whence $\langle e_0, e_1, \ldots \rangle$ is a behavior of A_P. Thus $L(A_S) \subseteq L(A_S)$. □

3 Incompleteness of Some Previous Approaches

In this section we consider some candidate progress measures for showing that $L(A_P) \subseteq L(A_S)$ holds. This leads to our proof that there can be no sound and complete verification method based on a progress measure that maps states of A_P either to states of A_S or to sets of states of A_S.

3.1 Incompleteness of Homomorphisms

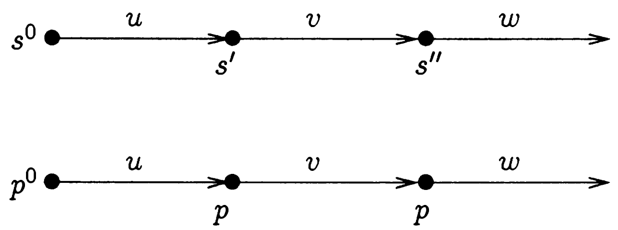
Homomorphisms (refinement mappings) do not yield a complete method for nondeterministic automata. Even for finite-state automata A_P and A_S such that A_P satisfies A_S, a homomorphism may not exist. To see this, assume that A_P satisfies A_S and that this can be proved by some homomorphism μ. Consider the situation:



where states p of A_P and s', s'' of A_S are all the states reachable over some finite u. Also assume that there exist w' and w'' such that $u \cdot w'$ and $u \cdot w''$ are different behaviors that allow only the runs depicted. Suppose that $p_0, \ldots, p, p'_0, p'_1, \ldots$ is the run of A_P over $u \cdot w'$ and that $p_0, \ldots, p, p''_0, p''_1, \ldots$ is the run over $u \cdot w''$. Thus $\mu(p_0), \ldots, \mu(p), \mu(p'_0), \mu(p'_1), \ldots$ must be the run of A_S over $u \cdot w'$ and $\mu(p_0), \ldots, \mu(p), \mu(p''_0), \mu(p''_1), \ldots$ must be the run of A_S over $u \cdot w''$, because A_P satisfies A_S. However, this is impossible because for the run over $u \cdot w'$, it must be the case that $\mu(p) = s'$, and for the run over $u \cdot w''$, it must be the case that $\mu(p) = s''$.

To avoid the incompleteness inherent in homomorphisms, we might consider a progress measure that maps program states to sets of specification states. For the situation above, we would define $\mu(p) = \{s', s''\}$, where $\{s', s''\}$ is called a *prophecy set* because it predicts that either s' or s'' is the state of the specification automaton corresponding to p.

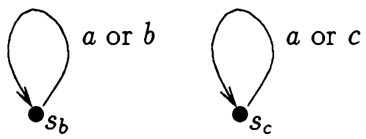
In the simple case that A_S is deterministic, the need for prophecy sets does not arise, and we might hope that a homomorphism for (A_P, A_S) would always exist. This is not always so, however. In the situation below, $\mu(p)$ would have to be both s' and s'' at the same time:



Again in this case, it would be natural to let $\mu(p)$ be a set, namely $\{s', s''\}$, which we call a *history set* since each state corresponds to a different finite behavior or history leading up to the state. Above, u and $u \cdot v$ are two such histories leading up to p.

3.2 Incompleteness of Measures Mapping to Sets of States

We now show that in the general case, where A_P and A_S both are nondeterministic (but only finite-state), not even progress measures that map to sets of specification states can form a complete verification method. Consider an automaton A_S given by



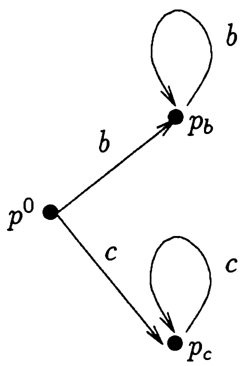
where both s_b and s_c are initial states. The behaviors defined by A_S are the sequences that consist of either a's and b's or a's and c's (i.e. the ω-regular language $(a+b)^\omega \cup (a+c)^\omega$). We first show that there can be no progress relation $\triangleright_S$ on $V_S = \{s_b, s_c\}$ yielding a *reasonable* verification method; such a method, we claim, would satisfy two criteria:

i. If $C_0 \overset{e_0}{\rhd}_S C_1 \overset{e_1}{\rhd}_S C_2 \overset{e_2}{\rhd}_S \cdots$, where $C_0, C_1, \ldots$ are sets of specification states, then there are $s_0 \in C_0$, $s_1 \in C_1, \ldots$ such that $s_0 \xrightarrow{e_1}_S s_1 \xrightarrow{e_2}_S \cdots$ is a run of A_S.

ii. If $L(A_P) \subseteq L(A_S)$ then a progress measure μ exists such that a state s need only be in $\mu(p)$ if there is a $u \in \Sigma^*$ such that both p and s are reachable over u and for some $w \in \Sigma^\omega$, $p \xrightarrow{w}_P$ and $s \xrightarrow{w}_S$ hold.

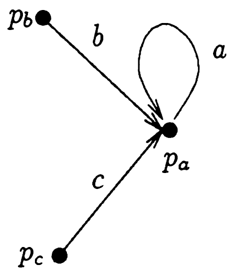
Criterion i must hold for the method to be sound; note that there need not be any condition on C_0 with respect to initial states because both states of A_S are initial. Criterion ii is an assumption that if $L(A_P) \subseteq L(A_S)$, then a progress measure μ exists such that $\mu(p)$ only contains states that actually occur in runs of A_S when p occurs in a corresponding run of A_P.

Our proof that there is no complete verification method based on a progress relation $\rhd_S$ satisfying the criteria involves two programs. The first is A_{P1}



where state p^0 is the initial state. There are two infinite behaviors of A_{P1}, namely $\langle b, b, b, \ldots \rangle$ and $\langle c, c, c, \ldots \rangle$, i.e. A_{P1} satisfies A_S. Thus since we assume that the hypothesized method is complete, there must exist a progress measure μ. By Criterion i, $\mu(p^0)$ must contain state s_b, because $p^0 \xrightarrow{b}_{P1} p_b \xrightarrow{b}_{P1} p_b \xrightarrow{b}_{P1} \cdots$ is a run of A_{P1} and the only corresponding run of A_S is $s_b \xrightarrow{b}_S s_b \xrightarrow{b}_S \cdots$. Similarly, $\mu(p^0)$ must contain s_c, $\mu(p_b)$ must contain s_b, and $\mu(p_c)$ must contain s_c. By Criterion ii, $\mu(p_b)$ does not contain s_c, and $\mu(p_c)$ does not contain s_b. Thus $\mu(p^0) = \{s_b, s_c\}$, $\mu(p_b) = \{s_b\}$, and $\mu(p_c) = \{s_c\}$. Since $p^0 \xrightarrow{b}_{P1} p_b \xrightarrow{b}_{P1} p_b \xrightarrow{b}_{P1} \cdots$ is a run of A_{P1}, $\mu(p^0) \overset{b}{\rhd}_S \mu(p_b) \overset{b}{\rhd}_S \mu(p_b) \overset{b}{\rhd}_S \cdots$ must hold. In particular, $\{s_b, s_c\} \overset{b}{\rhd}_S \{s_b\}$ must hold. By an analogous argument, $\{s_b, s_c\} \overset{c}{\rhd}_S \{s_c\}$ must hold.

The second program is A_{P2}



whose initial states are p_b and p_c. The behaviors of this program are $\langle b, a, a, a, \ldots\rangle$ and $\langle c, a, a, a, \ldots\rangle$. Thus A_{P2} satisfies A_S. By arguments similar to those above, $\mu(p_b) = \{s_b\}$, $\mu(p_c) = \{s_c\}$, and $\mu(p_a) = \{s_b, s_c\}$ hold. Thus $\{s_b\} \overset{b}{\triangleright}_S \{s_b, s_c\}$ and $\{s_c\} \overset{c}{\triangleright}_S \{s_b, s_c\}$ must hold.

From A_{P1} and A_{P2}, we conclude that there is a sequence $\{s_b, s_c\} \overset{c}{\triangleright}_S \{s_c\} \overset{c}{\triangleright}_S \{s_b, s_c\} \overset{b}{\triangleright}_S \{s_b\} \overset{b}{\triangleright}_S \{s_b, s_c\} \overset{c}{\triangleright}_S \cdots$. However, this contradicts Criterion i because A_S does not allow the behavior $\langle c, c, b, b, c, c, b, b, \cdots\rangle$. (Note that A_{P1} and A_{P2} are complete and that A_S is a safety automaton; thus the inadequacy result here is not related to the recursion-theoretic discussion in Section 7.)

4 Homomorphisms and Prophecy Measures

In this section we use progress measures to explain two well-known verification methods for establishing $L(A_P) \subseteq L(A_S)$, namely homomorphisms and prophecy measures.

4.1 Homomorphisms

By imposing restrictions on A_P and A_S, a complete verification method based on homomorphisms can be obtained:

Proposition 3 Let A_P be a historical complete automaton and let A_S be a deterministic automaton. If $L(A_P) \subseteq L(A_S)$ then (A_P, A_S) has a homomorphism.

Proof Assume $L(A_P) \subseteq L(A_S)$. Since A_P is historical and complete, every state p appears in some run and there is exactly one $u \in \Sigma^*$ such that $p \in \mathcal{R}_{A_P}(u)$. Define $\mu(v) = h_{A_S}(u)$, where h_{A_S} is the canonical mapping of the deterministic automaton A_S. It can be verified that μ so defined satisfies (REμ1) and (REμ2). $\square$

4.2 Prophecy Measures and the Classical Subset Construction

Similarly, it is not difficult to obtain a complete method based on mapping program states to prophecy sets. First we define a progress relation on such sets.

Definition 2 The *prophecy relation* $\triangleright_{\mathrm{PR}}$ of a transition relation $\rightarrow$ on V is the transition relation on $\mathcal{P}V$ given as:[4]

$(\triangleright_{\mathrm{PR}})\qquad S \overset{e}{\triangleright}_{\mathrm{PR}} S'$ if $\forall s' \in S' : \exists s \in S : \ s \xrightarrow{e} s'$

An infinite $\triangleright_{\mathrm{PR}}$-related sequence of non-empty finite sets gives rise to an infinite $\rightarrow$-related sequence of states:

Lemma 1 (Prophecy Relation Lemma) If $S_0 \overset{e_0}{\triangleright}_{\mathrm{PR}} S_1 \overset{e_1}{\triangleright}_{\mathrm{PR}} \cdots$ and $S_i \neq \emptyset$ is finite for all i, then there exists a sequence $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$ with $s_i \in S_i$ for $i \geq 0$.

Proof Construct a forest as follows. Each node is of the form $s_0 \xrightarrow{e_0} \cdots \xrightarrow{e_{n-1}} s_n$ such that $s_i \in S_i$ for $i \leq n$ and $s_i \xrightarrow{e_i} s_{i+1}$ for $i < n$; in particular, the roots are elements of S_0. The edges are of the form

$$(s_0 \xrightarrow{e_0} \cdots \xrightarrow{e_{n-1}} s_n\, ,\, s_0 \xrightarrow{e_0} \cdots \xrightarrow{e_n} s_{n+1}).$$

Since S_i is finite, the forest is a finite collection of finitely branching trees. The forest is infinite, because for all n, it follows from $S_0 \overset{e_0}{\triangleright}_{\mathrm{PR}} S_1 \overset{e_1}{\triangleright}_{\mathrm{PR}} \cdots$ and $S_i \neq 0$ that there are some $s_0, \ldots, s_n$ such that $s_0 \xrightarrow{e_0} \cdots \xrightarrow{e_{n-1}} s_n$ is a node. Hence by König's Lemma, there is an infinite path through one of the trees. This path defines $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$. □

Next we define a prophecy progress measure, which maps each program state to a finite prophecy set of specification states:

Definition 3 A *prophecy measure* μ for (A_P, A_S) is a mapping $\mu : V_P \rightarrow \mathcal{P}V_S$ such that:

(PRμ1) $\ p \in V_P^0 \ \Rightarrow \ \mu(p) \subseteq V_S^0$

(PRμ2) $\ p \xrightarrow{e}_P p' \ \Rightarrow \ \mu(p) \overset{e}{\triangleright}_{\mathrm{PR}} \mu(p')$

(PRμ3) $\ \mu(p) \neq \emptyset$ and finite

where $\triangleright_{\mathrm{PR}}$ is the prophecy relation of $\rightarrow_S$.

Prophecy measures give a sound verification method for nondeterministic automata:

[4] $\mathcal{P}V$ denotes the set of all subsets of V

Proposition 4 If (A_P, A_S) has a prophecy measure, then $L(A_P) \subseteq L(A_S)$.

Proof Assume that (A_P, A_S) has a prophecy measure μ. Let $p_0 \xrightarrow{e_0}_P p_1 \xrightarrow{e_1}_P \cdots$ be a run of A_P. By (PRμ2), $\mu(p_0) \overset{e_0}{\triangleright}_{\mathbf{PR}} \mu(p_1) \overset{e_1}{\triangleright}_{\mathbf{PR}} \cdots$, and by (PR$\mu$3), $\mu(p_i) \neq \emptyset$ and finite for $i \geq 0$. We can use the Prophecy Relation Lemma to obtain a sequence $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$, where $s_0 \in \mu(p_0)$. By (PRμ1), $s_0 \in \mu(p_0) \subseteq V_S^0$, whence $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$ is a run of A_S. □

A completeness result for prophecy measures can easily be obtained by taking advantage of the classical subset construction, which is:

Definition 4 Let $A = (\Sigma, V, \rightarrow, V^0)$ be a safety automaton. Define $\mathcal{D}A = (\Sigma, \mathcal{F}V, \rightarrow_D, \{V^0\})$, where $U \xrightarrow{e}_D U'$ if $U' \neq \emptyset$ is maximal such that $U \overset{e}{\triangleright}_{\mathbf{PR}} U'$, i.e. if $U' = \{v' \mid \exists v \in U : v \xrightarrow{e} v'\}$ and $U' \neq \emptyset$. (Since A is a safety automaton, V_0 is a finite set and U' is always a finite set.)[5]

Note that if A is finite-state with n states, then $\mathcal{D}A$ has 2^n states, and if A's state space is countably infinite, then $\mathcal{D}A$ has still only countably many states.

Proposition 5 $L(A) = L(\mathcal{D}A)$.

Proof The mapping μ defined by $\mu(v) = \{v\}$ is a homomorphism for $(A, \mathcal{D}A)$; thus by Proposition 2, $L(A) \subseteq L(\mathcal{D}A)$. Similarly, the mapping μ_D defined by $\mu(S) = S$ is a prophecy measure for (A_D, A); thus by Proposition 4, $L(\mathcal{D}A) \subseteq L(A)$. □

Proposition 6 Let A_P be a historical complete automaton and let A_S be a safety automaton. If $L(A_P) \subseteq L(A_S)$, then

- there exists a homomorphism for $(A_P, \mathcal{D}A_S)$, and
- there exists a prophecy measure for (A_P, A_S).

Proof A homomorphism μ for $(A_P, \mathcal{D}A_S)$ exists by Proposition 3 and from the definitions of prophecy measure and $\mathcal{D}A_S$, it follows that μ is a prophecy measure. □

According to the discussion in Section 3.1, the method of prophecy measures is not complete if the restriction that A_P be historical is removed.

[5] $\mathcal{F}V$ denotes the set of finite subsets of V.

5 History and ND Measures

In this section we first show how to prove $L(A_P) \subseteq L(A_S)$ when the specification automaton A_S is deterministic. The progress measures that arise are called history measures and the completeness of the resulting method stems from a new subset construction that we call *historization*. Next, we define ND measures as a natural generalization of history measures.

5.1 History Measures

Assume A_S is deterministic and consider a program state p. It can be reached by different finite behaviors. Let the progress measure $\mu(p)$ be the *history set*—the set of specification states that are reached by these finite behaviors (there is one such state per behavior because A_S is deterministic). On a transition $p \xrightarrow{e}_P p'$ and for each state $s \in \mu(p)$, there must be a state $s' \in \mu(p')$ such that $s \xrightarrow{e}_S s'$; this ensures that every partial run of A_S can be extended. Thus we define:

Definition 5 The *history relation* $\triangleright_{\mathrm{HI}}$ of a transition relation $\rightarrow$ on V is the transition relation on $\mathcal{P}V$ given as:

$$(\triangleright_{\mathrm{HI}}) \qquad C \overset{e}{\triangleright}_{\mathrm{HI}} C' \text{ if } \forall s \in C : \exists s' \in C' : \ s \xrightarrow{e} s'$$

The history relation of $\rightarrow$ has the following property:

Lemma 2 (History Relation Lemma) If $C_0 \overset{e_0}{\triangleright}_{\mathrm{HI}} C_1 \overset{e_1}{\triangleright}_{\mathrm{HI}} \cdots$, then for all $s_0 \in C_0$, there exists a sequence such that $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$ with $s_i \in C_i$ for all i.

Proof Let s_0 be any state in C_0. Then by definition of $\triangleright_{\mathrm{HI}}$, there is a state s_1 in C_1 such that $s_0 \xrightarrow{e_0} s_1$. By iterating this argument, we obtain $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$ such that for all i, $s_i \in C_i$. □

A history measure maps a program state to a possibly infinite set of specification states:

Definition 6 A *history measure* μ for (A_P, A_S) is a mapping $\mu : V_P \rightarrow \mathcal{P}V_S$ such that

$$(\mathrm{HI}\mu 1) \quad p \in V_P^0 \ \Rightarrow \ \exists s \in \mu(p) : s \in V_S^0$$

$$(\mathrm{HI}\mu 2) \quad p \xrightarrow{e}_P p' \ \Rightarrow \ \mu(p) \overset{e}{\triangleright}_{\mathrm{HI}} \mu(p')$$

where $\triangleright_{\mathrm{HI}}$ is the history relation of $\rightarrow_S$.

History measures also give a sound verification method for nondeterministic automata:

Proposition 7 If (A_P, A_S) has a history measure then $L(A_P) \subseteq L(A_S)$.

Proof Let $p_0 \xrightarrow{e_0}_P p_1 \xrightarrow{e_1}_P \cdots$ be a run of A_P. By (HIμ1) there is an $s_0 \in \mu(p_0)$ such that $s_0 \in V_S^0$. By (HIμ2), $\mu(p_0) \overset{e_0}{\rhd}_{\mathrm{HI}} \mu(p_1) \overset{e_1}{\rhd}_{\mathrm{HI}} \cdots$. Thus by the History Relation Lemma, there is a run $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$ of A_S. $\square$

Definition 7 Let $A = (\Sigma, V, \rightarrow, V^0)$ be an automaton. Define $\mathcal{H}A = (\Sigma, \mathcal{P}V - \emptyset, \rhd_{\mathrm{HI}}, \{C \subseteq V \mid C \cap V^0 \neq \emptyset\})$.

Note that if A is finite-state with n states, then $\mathcal{H}$ has 2^n states, and if A's state set is countably finite, then $\mathcal{H}A$ has $2^{\aleph_0}$ many states, i.e. uncountably many states.

Proposition 8 $L(A) = L(\mathcal{H}A)$.

Proof The mapping μ defined by $\mu(v) = \{v\}$ is a homomorphism for $(A, \mathcal{H}A)$; thus by Proposition 2, $L(A) \subseteq L(\mathcal{H}A)$. The mapping μ_H defined by $\mu_H(C) = C$ is a history measure for $(\mathcal{H}A, A)$; thus by Proposition 7, $L(\mathcal{H}A) \subseteq L(A)$. $\square$

The following is the key proposition of our paper:

Proposition 9 Let A_P be a complete automaton and let A_S be a deterministic automaton. If $L(A_P) \subseteq L(A_S)$, then

- there exists a homomorphism for $(A_P, \mathcal{H}A_S)$, and
- there exists a history measure for (A_P, A_S).

Proof Assume $L(A_P) \subseteq L(A_S)$ and let s^0 be the initial state of A_S. Define $\mu(p) = \{s \mid \exists u : p \in \mathcal{R}_{A_P}(u) \text{ and } s^0 \xrightarrow{u}_S s\}$. To prove (HI$\mu$1), let $p \in V_P^0$. Since $p \in \mathcal{R}_{A_P}(\langle\rangle)$, it follows that $s^0 \in \mu(p^0)$.

To see that (HIμ2) holds, let p, e, p', and s be such that $p \xrightarrow{e}_P p'$ and $s \in \mu(p)$. Thus there is a finite behavior u such that $s^0 \xrightarrow{u}_S s$ and $p \in \mathcal{R}_{A_P}(u)$. By assumption that A_S is complete, there is an s', which is unique, such that $s \xrightarrow{e}_S s'$. Then $s' \in \mu(p')$, because $p' \in \mathcal{R}_{A_P}(u \cdot \langle e \rangle)$. Thus (HI$\mu$2) holds and μ is a history measure for (A_P, A_S).

In addition, it follows from the definition of $\mathcal{H}A$ that μ is a homomorphism for $(A_P, \mathcal{H}A_S)$. $\square$

According to the results of Section 3.1, history measures do not constitute a complete verification method for A_S that are safety automata.

5.2 ND Measures

We have discussed progress relations that give complete methods for two special cases above: prophecy relations when A_P is historical and history relations when A_S is deterministic. Our solution to the general case consists of combining these relations: the ND progress relation is the history relation of the prophecy relation.

Definition 8 The *ND relation* $\triangleright_{\mathbf{ND}}$ on $\mathcal{PF}V$ of a transition relation $\rightarrow$ on V is defined as:

$$(\triangleright_{\mathbf{ND}}) \qquad C \overset{e}{\triangleright}_{\mathbf{ND}} C' \quad \text{if} \quad \begin{array}{l} \forall S \in C : \exists S' \in C' : \\ \quad \forall s' \in S' : \exists s \in S : s \xrightarrow{e} s' \end{array}$$

An immediate consequence of the two preceding lemmas is:

Lemma 3 (ND Relation Lemma) If $C_0 \overset{e_0}{\triangleright}_{\mathbf{ND}} C_1 \overset{e_1}{\triangleright}_{\mathbf{ND}} \cdots$, $S_0 \in C_0$, and $\emptyset \notin C_i$ for all i, then there is a sequence $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$ with $s_0 \in S_0$.

Proof As $S_0 \in C_0$ and as $C_0 \overset{e_0}{\triangleright}_{\mathbf{HI}} C_1 \overset{e_1}{\triangleright}_{\mathbf{HI}} \cdots$, there is by the History Relation Lemma a sequence $S_0 \overset{e_0}{\triangleright}_{\mathbf{PR}} S_1 \overset{e_1}{\triangleright}_{\mathbf{PR}} \cdots$ with $S_i \in C_i$.

Moreover since $\emptyset \notin C_i$, i.e. $S_i \neq \emptyset$, and since S_i is finite for all i, it follows by the Prophecy Relation Lemma that there is a sequence $s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots$ such that $s_0 \in S_0$. □

An ND measure μ associates with each program state a (history) set of (prophecy) sets of specification states:

Definition 9 An ND measure μ for (A_P, A_S) is a mapping $\mu : V_P \rightarrow \mathcal{PF}V_S$ such that

(NDμ1) $p \in V_P^0 \;\Rightarrow\; \exists S \in \mu(p) : S \subseteq V_S^0$

(NDμ2) $p \xrightarrow{e}_P p' \;\Rightarrow\; \mu(p) \overset{e}{\triangleright}_{\mathbf{ND}} \mu(p')$

(NDμ3) $\emptyset \notin \mu(p)$

Proposition 10 If (A_P, A_S) has an ND measure then $L(A_P) \subseteq L(A_S)$.

Proof Follows from the ND Relation Lemma. □

Our main result is:

Theorem 1 Let A_P be a complete automaton and let A_S be a safety automaton. If $L(A_P) \subseteq L(A_S)$, then

- there exists a homomorphism for $(A_P, \mathcal{HD}A_S)$, and
- there exists an ND measure for (A_P, A_S).

Proof Assume $L(A_P) \subseteq L(A_S)$. By Proposition 9, there is a homomorphism μ for $(A_P, \mathcal{HD}A_S)$. In addition, it is not hard to see that μ is also an ND progress measure for (A_P, A_S). □

6 Derivation of Previous Methods

In this section we derive from our ND measures Abadi and Lamport's method [AL88] as applied to safety properties. We also show how to obtain the verification methods of [Mer90, Sis89a].

The goal of [AL88] is to show $L(A_P) \subseteq L(A_S)$ by means of a homomorphism. This is done by adding *history* and *prophecy information* to the program automaton before the refinement mapping is constructed. This information is such that one can verify locally that the language $L(A_P)$ accepted does not shrink when it is added. The main result of [AL88] is that $L(A_P) \subseteq L(A_S)$ if and only if there is an automaton $A_{P''}$—obtained by adding first history, then prophecy information to A_P—and there is a refinement measure of $(A_{P''}, A_S)$. The work in [Mer90, Sis89a] uses prophecy measures and constitutes what can be regarded as an intermediate approach between ours and that of [AL88]. In [Sis89a] it was shown that a complete method results from modifying the program automaton and using a prophecy measure: $L(A_P) \subseteq L(A_S)$ if and only if there is an automaton $A_{P'}$—obtained by adding history information to A_P—and there is a prophecy measure of $(A_{P'}, A_S)$.

6.1 Adding History Information

Using ND measures, we can derive the methods of [Mer90, Sis89a] as follows.

Definition 10 $A_{P'} = (\Sigma, V_{P'}, \rightarrow_{P'}, V^0_{P'})$ is obtained from A_P by *adding* history information if $V_{P'} \subseteq V_P \times I$—with I countable—and

(HI1) $\quad p \in V^0_P \;\Rightarrow\; \exists i : (p,i) \in V^0_{P'}$

(HI2) $\quad p \rightarrow_P p' \;\wedge\; (p,i) \in V_{P'} \;\Rightarrow\; \exists i' : (p,i) \xrightarrow{e}_{P'} (p', i')$

The projection $\pi_{A_P} : V_{P'} \rightarrow V_P$ is defined by $\pi_{A_P}(p,i) = p$.

Note that (HI1) and (HI2) are equivalent to saying that the inverse projection $\pi^{-1}_{A_P} : V_P \rightarrow V_{P'}$ (defined by $\pi^{-1}_{A_P}(p) = \{(p,i) \mid (p,i) \in V_{P'}\}$) is a history measure for $(A_P, A_{P'})$. Also observe that $A_{P'}$ is not necessarily a complete automaton or a safety automaton, even if A_P is.

Proposition 11 If $A_{P'}$ is obtained from A_P by adding history information, then $L(A_P) \subseteq L(A_{P'})$.

Proof As noted above, $\pi^{-1}_{A_P}$ is a history measure for $(A_P, A_{P'})$. Thus by Proposition 7, $L(A_P) \subseteq L(A_{P'})$ holds. □

(The original definition of adding history variables in [AL88] is stronger and, as a result, implies that $L(A_P) = L(A_S)$.) The method of [Mer90, Sis89a] now follows from Theorem 1:

Proposition 12 Let A_P be a complete automaton and let A_S be a safety automaton. Then, $L(A_P) \subseteq L(A_S)$ if and only if there is an automaton $A_{P'}$—obtained by adding history information to A_P—and there is a prophecy measure for $(A_{P'}, A_S)$.

Proof "$\Leftarrow$" By Proposition 11, $L(A_P) \subseteq L(A_{P'})$, and by Proposition 4, $L(A_{P'}) \subseteq L(A_S)$.

"$\Rightarrow$" By Proposition 9, there is an homomorphism μ_{HD} for $(A_P, \mathcal{D}A_S)$. Let $I = \mathcal{F}V_S$, $V_{P'} = \{(p,S) \mid S \in \mu_{\mathrm{HD}}(p)\}$, $V^0_{P'} = \{(p,S) \in V_{P'} \mid p \in V^0_P, S \subseteq V^0_S\}$, and $(p,S) \xrightarrow{e}_{P'} (p', S')$ if $p \xrightarrow{e}_P p'$ and $S \overset{e}{\triangleright}_{\mathrm{PR}} S'$. Then $A_{P'}$ is obtained from A_P by adding history information, because $\pi^{-1}_{A_P} : p \mapsto (p, \mu_{\mathrm{HD}}(p))$ is a history measure for $(A_P, A_{P'})$.

Define $\mu(p,S) = S$. Then it can be seen that μ is a prophecy measure for $(A_{P'}, A_S)$. □

The completeness proofs in [AL88, Sis89a] rely on changing A_P into an infinite-state automaton by adding information that records the past history of states. In contrast, the analysis above shows that if A_P and A_S are finite-state, then $A_{P'}$

can be chosen to be finite-state; for in the proof of Proposition 12, the number of different history sets is finite when V_P is finite. In light of this observation, the concepts of history measure and history information are slightly misleading. Distinguishing among histories of the program automaton is not the crucial point—what matters is to distinguish among prophecy sets of the specification automaton.

6.2 Adding Prophecy Information

To obtain the verification method of [AL88], we define:

Definition 11 $A_{P'} = (\Sigma, V_{P'}, V^0_{P'}, \rightarrow_{P'})$ is obtained from A_P by *adding* prophecy information if $V_{P'} \subseteq V_P \times I$—with I countable—and

(PR1) $p \in V^0_P \;\wedge\; (p,i) \in V_{P'} \;\Rightarrow\; (p,i) \in V^0_{P'}$

(PR2) $p \xrightarrow{e}_P p' \;\wedge\; (p',i') \in V_{P'} \;\Rightarrow\; \exists i : (p,i) \xrightarrow{e}_{P'} (p',i')$

(PR3) $\emptyset \neq \{i \mid (p,i) \in V_{P'}\}$ is finite

The projection $\pi_{A_P} : V_{P'} \rightarrow V_P$ is defined by $\pi_{A_P}(p,i) = p$.

Note that this definition is equivalent to requiring that the inverse projection $\pi^{-1}_{A_P} : V_P \rightarrow V_{P'}$ is a prophecy measure for $(A_P, A_{P'})$. Also observe that $A_{P'}$ is not necessarily a safety automaton nor is it necessarily complete, even if A_P has these properties.

Proposition 13 If $A_{P'}$ is a safety automaton obtained from A_P by adding prophecy information, then $L(A_P) \subseteq L(A_{P'})$.

Proof This follows from Proposition 4 and the observation above that $\pi^{-1}_{A_P}$ is a prophecy measure for $(A_P, A_{P'})$. □

A version of Theorem 2 of [AL88] follows from Theorem 1:

Proposition 14 Let A_P be an automaton (assumed complete) and let A_S be a safety automaton. Then $L(A_P) \subseteq L(A_S)$ if there is a safety automaton $A_{P''}$—obtained by adding first history, then prophecy information to A_P—and there is a homomorphism for $(A_{P''}, A_S)$.

Proof "$\Leftarrow$" By Proposition 11 and Proposition 13, $L(A_P) \subseteq L(A_{P''})$. By Proposition 2, $L(A_{P''}) \subseteq L(A_S)$.

"$\Rightarrow$" Assume $L(A_P) \subseteq L(A_S)$. By Proposition 9 there is a homomorphism μ_{HD} of $(A_P, \mathcal{HD}A_S)$. Let $A_{P''} = (\Sigma, V_{P''}, \rightarrow_{P''}, V^0_{P''})$, where $V_{P''}, V^0_{P''} \subseteq V_P \times \mathcal{F}V_S \times V_S$ are given by:

$$\begin{aligned} V_{P''} &= \{(p, S, s) \mid s \in S \in \mu_{\mathrm{HD}}(p)\} \\ V^0_{P''} &= \{(p, S, s) \mid s \in S \in \mu_{\mathrm{HD}}(p) \ \wedge \ p \in V^0_P \ \wedge \ S \subseteq V^0_S\} \end{aligned}$$

and $(p, S, s) \xrightarrow{e}_{P''} (p', S', s')$ if $p \xrightarrow{e}_P p'$, $S \overset{e}{\triangleright}_{\mathrm{PR}} S'$ and $s \xrightarrow{e}_S s'$.

Then it is not hard to see that $A_{P''}$ is obtained by adding prophecy information to $A_{P'}$ from the proof of Proposition 12. Also, it can be seen that μ defined by $\mu(p, S, s) = s$ is a homomorphism for $(A_{P''}, A_S)$. □

Although the approaches of Abadi and Lamport [AL88] can be derived from ours, their work is more general in two respects. First, they show how safety and liveness issues can be separated by using automata that are equipped with auxiliary liveness properties (see also [Mer90]). Second, stuttering automata are used. A *stuttering automaton* is one in which repetition of events is considered a single event. Stuttering is important when multiple steps of the program automaton correspond to a single step of the specification automaton (see also [Sis89a]). For simplicity we have not considered this issue here. In [AL89], translations between the method of [AL88] and our method (originally described in [KS89]) were first outlined.

7 Discussion

Our verification methods hinge on two restrictions: that the specification automaton has only finite nondeterminism and that the program automaton is complete. The restriction to finite nondeterminism is also imposed in previous methods. As discussed in [Sis89b], there are recursion-theoretic arguments showing that there does not exist any sensible verification method for automata having unfinite nondeterminism; in fact, the question $L(A_P) \subseteq L(A_S)$ is Π^1_2-complete and we should expect a verification method to be at most in Σ^1_1 (corresponding to guessing a mapping and verifying first order conditions) or perhaps in Π^1_2 (corresponding to guessing a mapping and then verifying well-foundedness and first order conditions).

The restriction to complete program automata is also rooted in the laws of recursion theory. Just to determine if an effectively presented nondeterministic automaton A_P defines the empty set (i.e. that $L(A_P) \subseteq \emptyset$) is Π^1_1-complete, because there is a reduction from the Π^1_1-complete problem of determining whether an effectively represented tree has only finite paths. On the other hand, all the methods described here involve a second order existential quantification; i.e. each method is of the form: $L(A_P) \subseteq L(A_S)$ if and only if there is a relation R such that some first-order conditions hold.[6] Thus the methods are essentially Σ^1_1 and therefore cannot even be used for the general problem $L(A_P) \subseteq \emptyset$ where A_P is nondeterministic. One can lower the computational complexity by reformulating the verification problem. We say that A_P *simulates* A_S if each finite and infinite behavior of A_P is a behavior of A_S. Perhaps surprisingly, the problem of determining whether nondeterministic A_P simulates safety automaton A_S—something that looks stronger than $L(A) \subseteq L(S)$—is computationally much easier. In fact this problem can be shown to be Π^0_1-complete.[7]

8 Summary

We have described a verification method based on our ND progress measure for nondeterministic automata. Unlike previous complete methods, ours is direct in the sense that it requires modifying neither the program nor the specification. Progress measures also have allowed us to classify the applicability of previous methods that do not depend on program transformations. According to whether A_P is historical or not, or whether A_S is deterministic or safety, the progress measure indicated below constitutes a sound and complete verification method for showing $L(A_P) \subseteq L(A_S)$:

	A_S	
A_P	*deterministic*	*safety*
historical	refinement	prophecy
nondeterministic	history	ND

[6]An ND measure μ can be defined by $S \in \mu(p)$ if and only if $R(p, \#S)$, where $\#S$ is a number encoding the finite set S.

[7]Proof sketch: Assume the states of A_P and A_S are natural numbers and their transition relations are recursive. In addition, there is a maximal initial state, known to A_S, and a recursive function b, known to A_S, bounding the branching of A_S, i.e. $s \xrightarrow{e}_S s' \Rightarrow s' \leq b(s)$. The problem $L(A_P) \subseteq L(A_S)$ is in Π^0_1 because it can be written $\forall u \in \Sigma^* : P(u)$, where the predicate $P(u)$ is "u is a finite behavior of $A_P \Rightarrow u$ is a finite behavior of A_S", which can be shown to be Π^0_1; then $P(u)$ is a recursive predicate when A_P and A_S are recursively represented, and A_S is safety automaton ensures that $(\forall u \in \Sigma^* : P(u))$ implies $L(A_P) \subseteq L(A_S)$. Also, by standard recursion theoretic techniques, the Π^0_1-complete problem "Does M_x not halt on x?" (where M_x, $x = 0, 1, \ldots$ is an enumeration of Turing machines) can be reduced to a question of the form $\Sigma^\omega \subseteq L(A)$, where $\Sigma = \{0\}$ and A is a deterministic automaton that allows the finite behavior 0^n if and only if M_x does not halt on x after n steps.

Unfortunately, the most powerful progress measure, the ND measure, is rather complex since it maps program states to sets of sets of specification states. This complexity is inherent in the verification problem. No method based on just mapping program states to sets of specification states can be both sound and complete for nondeterministic automata.

Acknowledgments

We would like to thank M. Abadi, B. Alpern, D. Kozen, L. Lamport, M. Merritt, A. Zwarico, and an anonymous referee for their very helpful comments on earlier versions of this paper.

References

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

[AL89] M. Abadi and L. Lamport. Private communication, November 1989.

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

[AS85] B. Alpern and F.B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, Oct. 1985.

[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.

[CBK90] E. M. Clark, I.A. Browne, and R.P Kurshan. A unified approach for showing language containment and equivalence between various types of ω-automata. In A. Arnold, editor, *CAAP, LNCS 431*, pages 103–116, 1990.

[KK91] N. Klarlund and D. Kozen. Rabin measures and their applications to fairness and automata theory. In *Proc. Sixth Symp. on Logic in Computer Science*. IEEE, 1991. To appear.

[Kla90] Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations.* PhD thesis, TR-1153, Cornell University, August 1990.

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

[Lam83] L. Lamport. Specifying concurrent program modules. *ACM Transactions on Programming Languages and Systems*, 5(2):190–222, 1983.

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

[Mer90] M. Merritt. Completeness theorems for automata. In *Stepwise Refinement of Distributed Systems*, pages 544–560. Springer-Verlag, 1990. Lecture Notes in Computer Science, Vol. 430, Proceedings of the REX Workshop, Mook, The Netherlands, May/June 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. 5th GI conference*, pages 167–183, 1981. In Lecture Notes in Computer Science 104.

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

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research*, 3(2):115–125, 1959.

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

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

[Sta88] E. Stark. Proving entailment between conceptual state specifications. *Theoretical Computer Science*, 56:135–154, 1988.

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