INFORMATION AND COMPUTATION 107, 151–170 (1993)

Proving Nondeterministically Specified Safety Properties Using Progress Measures

NILS KLARLUND*

IBM T. J. Watson Research Center,
P.O. Box 704, Yorktown Heights, New York 10598

AND

FRED B. SCHNEIDER†

Department of Computer Science,
Cornell University, Ithaca, New York 14853

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, Sis89, 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,

* Supported by grants from the University of Aarhus, Denmark, the Danish Research Academy, and the Thanks to Scandinavia Foundation, Inc. Current address: Department of Computer Science, Aarhus University, Ny Munkegade, DK-8000 Aarhus C, Denmark.

† Supported in part by the Office of Naval Research under Contracts N00014-86-K-0092 and N00014-91-J-1219, the National Science Foundation under Grant CCR-8701103 and DARPA/NSF Grant CCR-9014363, 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.

0890-5401/93 $5.00

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 $L(A_P) \subseteq L(A_S)$ 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 nonnumeric sence, 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, $\mu(p) \overset{e}{\rhd}_S \mu(p')$ 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 $\rhd_S$-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 $\langle e_0, e_1, \ldots \rangle$. 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 [Sis89]. 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.)

[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].

Regrettably, there are simple situations where the inclusion $L(A_P) \subseteq L(A_S)$ holds but this cannot be proved by a homomorphism.

Abadi and Lamport [AL88] showed that there are two problems behind the indadequacy 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, Sis91, 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 $\mathscr{D}A_S$ obtained by applying the classic subset construction [RS59] for determinization, here denoted by $\mathscr{D}$, 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 shall see, 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 $\mathscr{H}$. The $\mathscr{H}$ construction allows us to explain history measures in terms of homomorphisms from A_P to $\mathscr{H}A_S$. Similarly, ND measures correspond to homomorphisms from A_P to $\mathscr{H}\mathscr{D}A_S$, which is the automaton obtained by first performing determinization and then performing historization. The significance of $\mathscr{H}\mathscr{D}A_S$ is that there is always a homomorphism from A_P to $\mathscr{H}\mathscr{D}A_S$ 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 $\mathscr{H}\mathscr{D}A_S$. Therefore, $\mathscr{H}\mathscr{D}A_S$ is *a canonical specification automaton.*

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

Moreover, as an alternative to constructing all of $\mathscr{HDA}_S$ at once and then using a homomorphism, the ND progress measure allows the description of only parts of $\mathscr{HDA}_S$. 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) $\langle e_0, e_1, \ldots \rangle$ 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 $u = \langle e_0, \ldots, e_n \rangle \in \Sigma^*$, $n \geqslant -1$, denote by $v \xrightarrow{u} v'$ that there exist $v_0, \ldots, v_{n+1} \in V$ 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 $v_0, v_1, \ldots$ such that $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_2} \cdots$, where $v = v_0$ and $w = \langle e_0, e_1, \ldots \rangle \in \Sigma^\omega$; if in addition $v_0 \in V^0$, then $v_0, v_1, \ldots$ 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 $\mathscr{R}_A(u)$. Note that $\mathscr{R}_A(\langle\ \rangle) = V^0$. 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 $\{v\} = \mathscr{R}_A(u)$. It is well known [AL88, Arn83, Kla90] that safety automata define the class of safety properties:[4]

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.*

A *dead-end* is a state v from which it is not possible to follow transitions ad infinitum, i.e., such that for no $w \in \Sigma^\omega$, $v \xrightarrow{w}$. Automaton A is *dead-end-free* if it has no dead-ends. Note that if a dead-end-free automaton has an initial state, then it accepts a non-empty language. Moreover, from every automaton $A = (\Sigma, V, \rightarrow, V^0)$, it is possible to obtain a dead-end-free automaton A' such that $L(A') = L(A)$ by deleting from V the dead-ends. 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 incomputability is usually not a problem, since predicates often can be used to describe the states occurring in infinite computations.

If for all v in V there is exactly one finite behavior u such that $v \in \mathscr{R}_A(u)$, then A is *historical*; the intuition is that each state describes the finite behavior or *history* leading up to that state. Note that if A is historical and $L(A) \neq \varnothing$, then A is infinite-state.

In what follows, we consider a program automaton $A_P = (\Sigma, V_P, \rightarrow_P, V^0_P)$ 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 conditions are satisfied:

(REμ1) $p \in V^0_P \Rightarrow \mu(p) \in V^0_S$

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

(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.

[4] A safety property is a closed set, i.e., the complement of an open set $\{u_i \cdot w \mid i \geq 1 \text{ and } w \in \Sigma^\omega\}$, where $\{u_i\}_{i \geq 1}$ is a set of finite prefixes, denoting the set of "bad things."

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, 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. It is well known that 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 in Fig. 1, 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

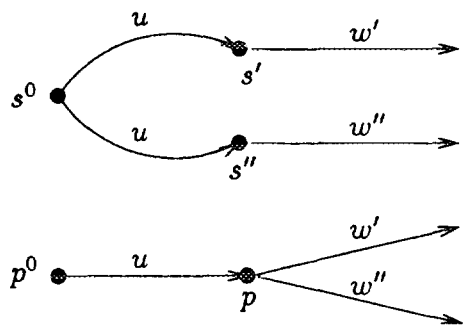

FIG. 1. Situation that calls for a prophecy set.

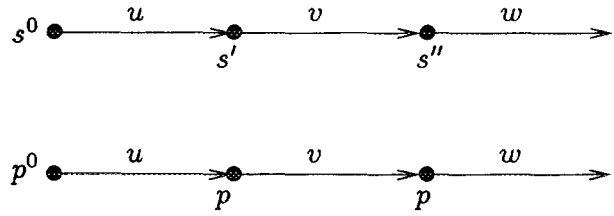

FIG. 2. Situation that calls for a history set.

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 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 where 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 shown in Fig. 2, $\mu(p)$ 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 as in Fig. 3, 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 $\rhd_S$ on $V_S = \{s_b, s_c\}$ yielding a *reasonable* verification method; such a method, we claim, would satisfy the criterion:

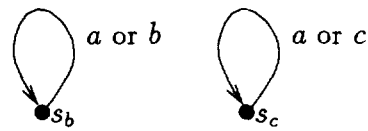

FIG. 3. Automaton A_S.

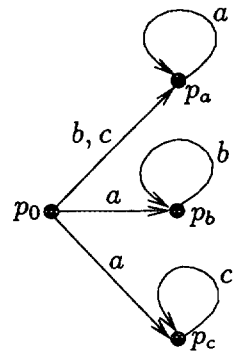

FIG. 4. Automaton A_P.

(M) 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 $s_0 \xrightarrow{e_0}_S s_1 \xrightarrow{e_1}_S \cdots$ is a run of A_S.

This criterion must hold for the method to be sound. Usually we would also expect a requirement relating initial states of A_P and A_S to C_0. In the case of A_S as defined above, however, there is no need for such a requirement since both states of A_S are initial.

Let the program automaton A_P be defined as shown in Fig. 4, where p_0 is the initial state. The infinite behaviors of A_P are $\langle b, a, a, \ldots \rangle$, $\langle c, a, a, \ldots \rangle$, $\langle a, b, b, \ldots \rangle$, and $\langle a, c, c, \ldots \rangle$. Observe that $L(A_P) \subseteq L(A_S)$. Since we assume that the hypothesized method is complete, there must then exist a progress measure μ that maps each state of A_P to a nonempty subset of $\{s_b, s_c\}$. There are only three such subsets. Thus μ is not injective. Since the verification condition is that $p \xrightarrow{e}_P p'$ implies $\mu(p) \overset{e}{\rhd}_S \mu(p')$, the case analysis in Table I gives a behavior w, which—according to (M)—must be a behavior of A_S. But in each case this contradicts the definition of A_S; thus no reasonable verification method exists.

TABLE I

Case Analysis

Case	Consequence	Behavior w
$\mu(p_0) = \mu(p_a)$	$\mu(p_0) \overset{b}{\rhd}_S \mu(p_a) = \mu(p_0) \overset{c}{\rhd}_S \mu(p_a) = \mu(p_0) \cdots$	$\langle b, c, b, c, \ldots \rangle$
$\mu(p_0) = \mu(p_b)$	$\mu(p_b) \overset{b}{\rhd}_S \mu(p_b) = \mu(p_0) \overset{c}{\rhd}_S \mu(p_a) \overset{a}{\rhd}_S \mu(p_a) \cdots$	$\langle b, c, a, a, \ldots \rangle$
$\mu(p_0) = \mu(p_c)$	$\mu(p_c) \overset{c}{\rhd}_S \mu(p_c) = \mu(p_0) \overset{b}{\rhd}_S \mu(p_a) \overset{a}{\rhd}_S \mu(p_a) \cdots$	$\langle c, b, a, a, \ldots \rangle$
$\mu(p_a) = \mu(p_b)$	$\mu(p_0) \overset{c}{\rhd}_S \mu(p_a) = \mu(p_b) \overset{b}{\rhd}_S \mu(p_b) \cdots$	$\langle c, b, b, \ldots \rangle$
$\mu(p_a) = \mu(p_c)$	$\mu(p_0) \overset{b}{\rhd}_S \mu(p_a) = \mu(p_c) \overset{c}{\rhd}_S \mu(p_c) \cdots$	$\langle b, c, c, \ldots \rangle$
$\mu(p_b) = \mu(p_c)$	$\mu(p_b) \overset{b}{\rhd}_S \mu(p_b) = \mu(p_c) \overset{c}{\rhd}_S \mu(p_c) \overset{c}{\rhd}_S \mu(p_c) \cdots$	$\langle b, c, c, \ldots \rangle$

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 dead-end-free 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, there is for every state p exactly one $u \in \Sigma^*$ such that $p \in \mathscr{R}_{A_P}(u)$. Because A_P is dead-end-free, there is a w such that $u \cdot w \in L(A_P)$. Therefore, $u \cdot w \in L(A_P) \subseteq L(A_S)$, whence $u \in L_*(A_S)$. Thus we can 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). ∎

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 prophency sets. First we define a progress relation on such sets.

Definition 2. The *prophecy relation* $\rhd_{PR}$ of a transition relation $\rightarrow$ on V is the transition relation on $\mathscr{P}V$ given as[5]

$$(\rhd_{PR}) \quad S \overset{e}{\rhd}_{PR} S' \text{ if } \forall s' \in S' : \exists s \in S : s \xrightarrow{e} s'.$$

An infinite $\rhd_{PR}$-related sequence of nonempty finite sets gives rise to an infinite $\rightarrow$-related sequence of states:

Lemma 1. (Prophecy Relation Lemma). *If $S_0 \overset{e_0}{\rhd}_{PR} S_1 \overset{e_1}{\rhd}_{PR} \cdots$ and $S_i \neq \varnothing$ 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 $i \geqslant 0$.*

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 $i \leqslant n$ 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

[5] $\mathscr{P}V$ denotes the set of all subsets of V.

trees. The forest is infinite, because for all n, it follows from $S_0 \overset{e_0}{\rhd}_{PR} S_1 \overset{e_1}{\rhd}_{PR} \cdots$ 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 $\mu : V_P \to \mathscr{P}V_S$ such that

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

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

(PRμ3) $\mu(p) \neq \varnothing$ and finite,

where $\rhd_{PR}$ is the prophecy relation of $\to_S$.

Prophecy measures give a sound verification method for nondeterministic automata:

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), $\mu(p_0) \overset{e_0}{\rhd}_{PR} \mu(p_1) \overset{e_1}{\rhd}_{PR} \cdots$, and by (PR$\mu$3), $\mu(p_i) \neq \varnothing$ 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 $A = (\Sigma, V, \to, V^0)$ be a safety automaton. Define $\mathscr{D}A = (\Sigma, \mathscr{F}V, \to_D, \{V^0\})$, where $U \xrightarrow{e}_D U'$ if $U' \neq \varnothing$ is maximal such that $U \overset{e}{\rhd}_{PR} U'$, i.e., if $U' = \{v' \mid \exists v \in U : v \xrightarrow{e} v'\}$ and $U' \neq \varnothing$. (Since A is a safety automaton, V_0 is a finite set and U' is always a finite set.[6]) $\mathscr{D}A$ is the *determinization* of A.

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

PROPOSITION 5. $L(A) = L(\mathscr{D}A)$.

Proof. To see that $L(A) \subseteq L(\mathscr{D}A)$, let $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \cdots$ be a run of A and define $U_0 = V_0$, $U_i = \{v' \mid \exists v \in U_{i-1} : v \xrightarrow{e_{i-1}} v'\}$, $i > 0$. By induction, $v_i \in U_i$. Thus $U_0 \xrightarrow{e_0}_D U_1 \xrightarrow{e_1}_D \cdots$, which is a run of $\mathscr{D}A$.

[6] $\mathscr{F}V$ denotes the set of finite subsets of V.

On the other hand, $L(\mathscr{D}A) \subseteq L(A)$ holds by Proposition 4 since the mapping μ_D defined by $\mu(S) = S$ is a prophecy measure for $(\mathscr{D}A, A)$. ∎

PROPOSITION 6. *Let A_P be a historical dead-end-free automaton and let A_S be a safety automaton. If $L(A_P) \subseteq L(A_S)$, then*

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

Proof. A homomorphism μ for $(A_P, \mathscr{D}A_S)$ exists by Proposition 3 and from the definitions of prophecy measure and $\mathscr{D}A_S$, 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 must be historical is removed.

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* $\rhd_{HI}$ of a transition relation $\rightarrow$ on V is the transition relation on $\mathscr{P}V$ given as

$(\rhd_{HI})$ $C \overset{e}{\rhd}_{HI} C'$ 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 $C_0 \overset{e_0}{\rhd}_{HI} C_1 \overset{e_1}{\rhd}_{HI} \cdots$, 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 $\rhd_{\mathrm{HI}}$, 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 $(A_{\mathrm{P}}, A_{\mathrm{S}})$ is a mapping $\mu : V_{\mathrm{P}} \to \mathscr{P}V_{\mathrm{S}}$ such that

(HIμ1) $p \in V_{\mathrm{P}}^0 \Rightarrow \exists s \in \mu(p) : s \in V_{\mathrm{S}}^0$

(HIμ2) $p \xrightarrow{e}_{\mathrm{P}} p' \Rightarrow \mu(p) \overset{e}{\rhd}_{\mathrm{HI}} \mu(p')$,

where $\rhd_{\mathrm{HI}}$ is the history relation of $\rightarrow_{\mathrm{S}}$.

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

PROPOSITION 7. *If $(A_{\mathrm{P}}, A_{\mathrm{S}})$ has a history measure, then $L(A_{\mathrm{P}}) \subseteq L(A_{\mathrm{S}})$.*

Proof. Let $p_0 \xrightarrow{e_0}_{\mathrm{P}} p_1 \xrightarrow{e_1}_{\mathrm{P}} \cdots$ be a run of A_{P}. By (HIμ1) there is an $s_0 \in \mu(p_0)$ such that $s_0 \in V_{\mathrm{S}}^0$. 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 $s_0 \xrightarrow{e_0}_{\mathrm{S}} s_1 \xrightarrow{e_1}_{\mathrm{S}} \cdots$ of A_{S}. ∎

To obtain a completeness result for history measures, we define a new automaton construction, which is the dual of the classical subset construction.

DEFINITION 7. Let $A = (\Sigma, V, \rightarrow, V^0)$ be an automaton. Define $\mathscr{H}A = (\Sigma, \mathscr{P}V, \rhd_{\mathrm{HI}}, \{C \subseteq V \mid C \cap V^0 \neq \varnothing\})$. $\mathscr{H}A$ is the *historization* of A.

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

PROPOSITION 8. $L(A) = L(\mathscr{H}A)$.

Proof. The mapping μ defined by $\mu(v) = \{v\}$ is a homomorphism for $(A, \mathscr{H}A)$; thus by Proposition 2, $L(A) \subseteq L(\mathscr{H}A)$. The mapping μ_{H} defined by $\mu_{\mathrm{H}}(C) = C$ is a history measure for $(\mathscr{H}A, A)$; thus by Proposition 7, $L(\mathscr{H}A) \subseteq L(A)$. ∎

The following is the key proposition of our paper:

PROPOSITION 9. *Let A_{P} be a dead-end-free automaton and let A_{S} be a deterministic automaton. If $L(A_{\mathrm{P}}) \subseteq L(A_{\mathrm{S}})$, then*

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

Proof. Asume $L(A_P) \subseteq L(A_S)$ and let s^0 be the initial state of A_S. Define $\mu(p) = \{s \mid \exists u : p \in \mathscr{R}_{A_P}(u) \text{ and } s^0 \xrightarrow{u}_S s\}$. Note that $\mu(p)$ is possibly empty.

To prove (HIμ1), let $p \in V_P^0$. Since $p \in \mathscr{R}_{A_P}(\langle\ \rangle)$, it follows that $s^0 \in \mu(p)$.

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 $p \in \mathscr{R}_{A_P}(u)$. By the assumption that A_P is dead-end-free, there exists w such that $p' \xrightarrow{w}$, hence $u \cdot \langle e \rangle \cdot w \in L(A_P)$. Since $L(A_P) \subseteq L(A_S)$ and A_S is deterministic, there is an s' such that $s \xrightarrow{e}_S s'$. Then $s' \in \mu(p')$, because $p' \in \mathscr{R}_{A_P}(u \cdot \langle e \rangle)$. Thus (HI$\mu$2) holds and μ is a history measure for (A_P, A_S).

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

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* $\rhd_{ND}$ on $\mathscr{P}\mathscr{F}V$ of a transition relation $\rightarrow$ on V is defined as

$$(\rhd_{ND}) \quad C \overset{e}{\rhd}_{ND} C' \qquad \text{if} \quad \forall S \in C : \exists S' \in C' : \forall s' \in S' : \exists s \in S : s \xrightarrow{e} s'.$$

An immediate consequence of the two preceding lemmas is:

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

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

Moreover since $\varnothing \notin C_i$, i.e., $S_i \neq \varnothing$, 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 $\mu : V_P \to \mathscr{P}\mathscr{F}V_S$ such that

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

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

(NDμ3) $\varnothing \notin \mu(p)$.

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 dead-end-free automaton and let A_S be a safety automaton. If $(A_P) \subseteq L(A_S)$, then*

- *there exists a homomorphism for $(A_P, \mathscr{K}\mathscr{D}A_S)$, 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 $(A_P, \mathscr{K}\mathscr{D}A_S)$. 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 [Ai88] as applied to safety properties. We also show how to obtain the verification methods of [Mer90, Sis91].

The goal of [AL88] is to show that $L(A_P) \subseteq L(A_S)$ 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 for $(A_{P'}, A_S)$. The work in [Mer90, Sis91] uses prohecy measures and constitutes what can be regarded as an intermediate approach between ours and that of [AL88]. In [Sis91] 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, Sis91] 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) $p \in V^0_P \Rightarrow \exists i : (p, i) \in V^0_{P'}$

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

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 hat $A_{P'}$ is not necessarily a dead-end-free 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, Sis91] now follows from Theorem 1:

PROPOSITION 12. *Let A_P be a dead-end-free 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. "⇐" By Proposition 11, $L(A_P) \subseteq L(A_{P'})$, and by Proposition 4, $L(A_{P'}) \subseteq L(A_S)$.

"⇒" By Proposition 9, there is a homomorphism μ_{HD} for $(A_P, \mathscr{H}\mathscr{S}A_S)$. Let $I = \mathscr{F}V_S$, $V_{P'} = \{(p, S) \mid S \in \mu_{HD}(p)\}$, $V^0_{P'} = \{(p, S) \in V_{P'} \mid p \in V^0_P, S \subseteq V^0_S\}$, and $(p, S) \xrightarrow{e}_{P'} (p', S')$ if $p \xrightarrow{e}_P p'$ and $S \overset{e}{\rhd}_{PR} S'$. Then $A_{P'}$ is obtained from A_P by adding history information, because $\pi^{-1}_{A_P} : p \mapsto \{(p, S) \mid S \in \mu_{HD}(p)\}$ 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, Sis91] 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) $\varnothing \neq \{i \mid (p, i) \in V_{P'}\}$ 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 $\pi^{-1}_{A_P} : V_P \rightarrow V_{P'}$ be 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 a dead-end-free automaton 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. "⇐" By Proposition 11 and Proposition 13, $L(A_P) \subseteq L(A_{P''})$. By Proposition 2, $L(A_{P''}) \subseteq L(A_S)$.

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

$$V_{P''} = \{(p, S, s) \mid s \in S \in \mu_{HD}(p)\}$$

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

and $(p, S, s) \xrightarrow{e}_{P''} (p', S', s')$ if $p \xrightarrow{e}_P p'$, $S \overset{e}{\rhd}_{PR} S'$ 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, they use stuttering automata. 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 [Sis91]). 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 may only have finite nondeterminism and that the program automaton must be dead-end-free. The restriction to finite nondeterminism is also imposed in previous methods. As discussed in [Sis89], there are recursion-theoretic arguments showing that there does not exist any sensible verification method for automata having infinite nondeterminism; in fact, the question $L(A_P) \subseteq L(A_S)$ is Π_2^1-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 Σ_2^1 (corresponding to guessing a mapping and then verifying well-foundedness and first order conditions).

The restriction to dead-end-free 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 $L(A_P) \subseteq \varnothing$) 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 following form: $L(A_P) \subseteq L(A_S)$ if and only if there is a relation R such that some first-order conditions hold.[7] Thus the methods are essentially Σ_1^1 and therefore cannot even be used for the general problem $L(A_P) \subseteq \varnothing$ where A_P is not assumed

[7] 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.

TABLE II

Progress Measures for Sound and Complete Verification Methods

	A_S	
A_P (dead-end-free)	Deterministic	Safety
Historical	Refinement	Prophecy
Nondeterministic	History	ND

dead-end-free. 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_1^0-complete.[8]

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 in Table II constitutes a sound and complete verification method for showing $L(A_P) \subseteq L(A_S)$. 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

[8] *Proof sketch.* Assume that the states of A_P and A_S are natural numbers and that their transition relations and sets of initial states are recursively represented. In addition, there are 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., $s \xrightarrow{c}_S s' \Rightarrow s' \leqslant b(s)$. The problem $L(A_P) \subseteq L(A_S)$ is in Π_1^0 because it can be written $\forall u \in \Sigma^* : P(u)$—where the recursive 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_1^0. A_S is a 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_1^0-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.

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 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.

RECEIVED September 21, 1989; FINAL MANUSCRIPT RECEIVED September 10, 1991

Note added in proof. Since the present paper was submitted, it was reported in [AAM92] that a construction resembling our historization is used in a technical report [Car70], published in 1970, to obtain a characterization of minimal nondeterministic automata equivalent to a given automaton.

In the same year, and probably independently, a similar construction was published in [KW70] as part of an algorithm to search for a minimal nondeterministic automaton. We are grateful to A. Potthoff at the University of Kiel for providing us with this information. He also located a technical report appearing a couple of years later [Kim74], which further investigated the issue.

This earlier work did not address the verification issue. Thus concepts such as automata accepting infinite words and mappings from states to sets of states (or sets of sets of states) were not considered.

REFERENCES

[AAM92] ARNOLD, A., DICKY, A., AND NIVAT, M. (1992), A note about minimal nondeterministic automata, *Bulletin of the EATCS* **47**, 166–169.

[AL88] ABADI, M., AND LAMPORT, L. (1991), The existence of refinement mappings, *Theoret. Comput. Sci.* **2**(82), 253–284.

[AL89] ABADI, M., AND LAMPORT, L. (1989), Private communication.

[Arn83] ARNOLD, A. (1983), Topological characterizations of infinite behaviors of transition systems, *in* "Proceedings, 10th Colloquium Automata, Languages and Programming," pp. 490–510, Lecture Notes in Computer Science, Vol. 154, Springer-Verlag, Berlin/New York.

[AS85] ALPERN, B., AND SCHNEIDER, F. B. (1985), Defining liveness, *Inform. Process. Lett.* **21**, 181–185.

[AS89] ALPERN, B., AND SCHNEIDER, F. B. (1989), Verifying temporal properties without temporal logic, *ACM Trans. Programming Languages Systems* **11**(1), 147–167.

[Car70] CARREZ, C. (1970), On the minimalization of non-deterministic automata, Technical report, Laboratoire de Calcul de la Faculté des Sciences de l'Université de Lille.

[CBK90] CLARK, E. M., BROWNE, I. A., AND KURSHAN, R. P. (1990), A unified approach for showing language containment and equivalence between various types of ω-automata, *in* "CAAP" (A. Arnold, Ed.), pp. 103–116, Lecture Notes in Computer Science, Vol. 431, Springer-Verlag, Berlin/New York.

[Kim74] KIM, J. (1974), State minimization of nondeterministic machines, Technical Report RC4896 (#21805), IBM T. J. Watson Research Center.

[KK91] KLARLUND, N., AND KOZEN, D. (1991), Rabin measures and their applications to fairness and automata theory, *in* "Proceedings, Sixth Symposium on Logic in Computer Science," IEEE, New York.

[Kla90] KLARLUND, N. (1990), "Progress Measures and Finite Arguments for Infinite Computations," Ph.D. thesis and Technical Report TR-1153, Cornell University.

[KS89] KLARLUND, N., AND SCHNEIDER, F. B. (1989), Verifying safety properties using infinite-state automata, Technical Report TR-1036, Cornell University.

[KW70] KAMEDA, T., AND WEINER, P. (1970), On the state minimization of nondeterministic automata, *Trans. on Computers*, C-19 **7**, 617–627.

[Lam83] LAMPORT, L. (1983), Specifying concurrent program modules, *ACM Trans. Programming Languages Systems* **5**(2), 190–222.

[LT87] LYNCH, N., AND TUTTLE, M. (1987), Hierarchical correctness proofs for distributed algorithms, *in* "Proceedings, Sixth Symposium on the Principles of Distributed Computing," pp. 137–151, Assoc. Comput. Math., New York.

[Mer90] MERRITT, M. (1990), Completeness theorems for automata, *in* "Stepwise Refinement of Distributed Systems," Proceedings of the REX Workshop, Mook, The Netherlands, May/June 1989, pp. 544–560, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.

[Par81] PARK, D. (1981), Concurrency and automata on infinite sequences, *in* "Proceedings, 5th GI Conference" (P. Deussen, Ed.), pp. 167–183, Lecture Notes in Computer Science, Vol. 104, Springer-Verlag.

[Rog67] ROGERS, H., JR. (1967), "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York.

[RS59] RABIN, M. O., AND SCOTT, D. (1959), Finite automata and their decision problems, *IBM J. Res.* **3**(2), 115–125.

[Sis89] SISTLA, A. P. (1989), On verifying that a concurrent program satisfies a nondeterministic specification, *Inform. Process. Lett.* **32**(1), 17–24.

[Sis91] SISTLA, A. P. (1991), Proving correctness with respect to nondeterministic safety specifications, *Inform. Process. Lett.* **39**(1).

[Sta88] STARK, E. (1988), Proving entailment between conceptual state specifications, *Theoret. Comput. Sci.* **56**, 135–154.

[Var87] VARDI, M. (1987), Verification of concurrent programs: The automata-theoretic framework, *in* "Proceedings, Symposium on Logic in Computer Science," IEEE, New York.

Printed in Belgium
Uitgever: Academic Press, Inc.
Verantwoordelijke uitgever voor België:
Hubert Van Maele
Altenastraat 20, B-8310 Sint-Kruis