INFORMATION AND COMPUTATION 121, 214–233 (1995)

Forward and Backward Simulations

I. Untimed Systems*

NANCY LYNCH

Massachusetts Institute of Technology, Laboratory for Computer Science, Cambridge, Massachusetts 02139
E-mail: lynch@theory.lcs.mit.edu

AND

FRITS VAANDRAGER

Centrum voor Wiskunde en Informatica, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands, and
University of Amsterdam, Programming Research Group, Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands
E-mail: fritsv@cwi.nl

A unified, comprehensive presentation of simulation techniques for verification of concurrent systems is given, in terms of a simple untimed automaton model. In particular, (1) refinements, (2) forward and backward simulations, (3) hybrid forward–backward and backward–forward simulations, and (4) history and prophecy relations are defined. History and prophecy relations are abstract versions of the history and prophecy variables of Abadi and Lamport, as well as the auxiliary variables of Owicki and Gries. Relationships between the different types of simulations, as well as soundness and completeness results, are stated and proved. Finally, it is shown how invariants can be incorporated into all the simulations. Even though many results are presented here for the first time, this paper can also be read as a survey (in a simple setting) of the research literature on simulation techniques. The development for untimed automata is designed to support a similar development for timed automata. Part II of this paper will show how the results of this paper can be carried over to the setting of timed automata. © 1995 Academic Press, Inc.

CONTENTS

1. INTRODUCTION

Much of the current work in verification of concurrent systems is based on the use of *simulation techniques*. A simulation proof involves establishing a correspondence known as a *simulation* between the states of two systems, A and B, where one (A) is regarded as an implementation and the other (B) is regarded as a specification. The correspondence is generally defined in terms of individual states and transitions, rather than in terms of entire executions. The existence of a simulation is used to show that any behavior that can be exhibited by A can also be exhibited by B; thus, if B solves some problem of interest, so does A. Typically, system A contains more details than B, or is an optimized or distributed version of B. Simulation techniques work for timing-based as well as untimed systems.

The research literature contains a large number of different types of simulations. Although all have the same general goals, there are many variations, some significant and some not, in their definitions and properties. An obstacle to the use of simulations in practice is that there has been, so far, no unified, comprehensive presentation of simulation methods. Our goal is to provide such a presentation: to identify the most important types of simulations, to express them in a common framework, to clarify the relationships between them, and to identify those properties

* This work was supported by ONR Contracts N00014-85-K-0168 and N00014-91-J-1988, by NSF Grants CCR-8915206 and CCR-9225124, by DARPA Contracts N00014-89-J-1988 and N00014-92-J-4033, and by ONR-AFOSR Contract F49620-94-1-0199. Part of this work took place while the second author was employed by the Ecole des Mines, CMA, Sophia Antipolis, France. The second author also received partial support from ESPRIT Basic Research Action 7166, CONCUR2. An earlier version of this paper (Parts I and II) appeared as [36].

0890-5401/95 $12.00

that are significant for verification purposes. We present our results for the untimed setting in this paper, and extend them to the timed setting in Part II [38].

Specifically, in this paper, we present forward and backward simulation techniques for proving trace inclusion relationships between concurrent systems. We describe all the simulation techniques in terms of a simple and general untimed automaton model that includes internal actions. Among the kinds of simulations we define are *refinements*, *forward simulations*, *backward simulations*, and hybrid versions that we call *forward–backward* and *backward–forward simulations*. We also define *history relations* and *prophecy relations*, which are abstract versions of the history and prophecy variables, respectively, of Abadi and Lamport [1]. We prove implication relationships among the different types of simulations, as well as soundness and completeness theorems. Finally, we show how invariants can be incorporated into all of the simulations.

The simplest simulations we consider are *refinements*. Refinements are similar to homomorphisms between automata in the sense of classical automata theory [10], and to the data refinements that are used in program development to replace abstract mathematical data structures by concrete structures that are more easily implemented [40, 15, 30, 18]. Lamport [28] advocates the use of refinements to prove that one concurrent program module implements another. A refinement from an automaton A to another automaton B is a function from states of A to states of B such that (a) the image of every start state of A is a start state of B, and (b) every step of A has a corresponding sequence of steps of B that begins and ends with the images of the respective beginning and ending states of the given step, and that has the same external actions. This notion of refinement implies that the traces of A are also traces of B. We give soundness and partial completeness results for refinements.

We next consider *forward simulations* and *backward simulations*, generalizations of refinements that allow a set of states of B to correspond to a single state of A. Forward simulations are similar to the simulations of [44, 19, 21], the possibilities mappings of [33, 35], the downward simulations of [17, 23, 13], the forward simulations of [22], and the history measures of [25]. The correspondence conditions (a) and (b) for refinements are generalized so that (a) every start state of A has *some* image that is a start state of B, and (b) every step of A and every state of B corresponding to the *beginning* state of the step yield a corresponding sequence of steps of B ending with an image of the *ending* state of the given step. Again, we give soundness and partial completeness results.

Backward simulations are similar to the upward simulations of [17, 23, 13], the prophecy mappings of [39], the backwards simulations of [21], and the prophecy measures of [25]. In the case of a backward simulation, conditions (a) and (b) for refinements are generalized so that (a) *all* images of every start state of A are start states of B, and (b) every step of A and every state of B corresponding to the *ending* state of the step yield a corresponding sequence of steps of B beginning with an image of the *beginning* state of the given step. Again, we give soundness and partial completeness results.

Next, we consider two combinations of forward and backward simulations, which we call *forward–backward* and *backward–forward* simulations, respectively. These are essentially compositions of one forward and one backward simulation, in the two possible orders. The definition of a forward–backward simulation was inspired by the work of Klarlund and Schneider [24, 25] for the case without internal actions. Forward–backward simulations are also similar to the subset-simulations of [22] and the simple failure simulations of [9]. Our new notion of a backward–forward simulation is suggested by symmetry with forward–backward simulations. We give soundness and completeness results; while some of the results for backward–forward simulations are symmetric with those for forward–backward simulations, others (notably certain completeness results) are different.

The final simulations we consider are *history relations* and *prophecy relations*. These are new and abstract versions of the *history* and *prophecy variables* of Abadi and Lamport [1]. The basic concept of history variables goes back at least as far as Lucas [32]. Owicki and Gries [43] defined history variables (which they called auxiliary variables) and used them in verifying parallel programs. Subsequently, Abadi and Lamport [1] gave a more abstract, language independent definition of history variables, and also introduced the dual concept of a prophecy variable. Severa authors observed that history and prophecy variables are closely related to forward and backward simulations, respectively, [39, 22, 25]. Inspired by this, we define in this paper the even more abstract notions of *history* and *prophecy relations*, and show their equivalence with the history, respectively prophecy, variables of [1]. According to our definitions, a history relation is simply a forward simulation whose inverse is a refinement, while a prophecy relation is simply a backward simulation whose inverse is a refinement. We prove some simple new characterizations; e.g., a forward simulation from A to B is equivalent to the combination of a history relation from A to some C and a refinement from C to B, and analogously for a backward simulation and a prophecy relation. We also give a simple new proof of a completeness result of Abadi and Lamport.

Finally, we address the issue of integrating invariants into simulation proofs. Our main development is carried out without mention of invariants, for the sake of simplicity. However, in actual verification examples using simulations, it is almost always the case that a preliminary collection of invariants is proved, then used where needed in proving the step correspondence. We state results showing how invariants can be used in conjunction with all the types of simulations.

We have crafted the development in this paper to be compatible with a similar development for timed systems; this work appears in Part II [38]. There, we define a new type of automaton called a *timed automaton*, and use it to define timed versions of all the simulations in this paper. Happily, the results for the timed setting turn out to be analogous to those for the untimed setting. In nearly all cases, the results for the timed setting are derived from those for the untimed setting, while in the few remaining cases, new proofs analogous to those in this paper are presented.

The usefulness of refinement mappings, history variables, and forward simulations in proving correctness has been well demonstrated. Abstraction mappings, which are essentially refinement mappings, comprise a basic proof method for implementations of abstract data types [30, 18]. They are also widely used in the verification of concurrent and reactive systems. Some typical examples can be found in [28, 14]. There is also a long tradition of using history variables in program verification [32, 6, 43, 45]. Often history variables are used together with refinements, see for instance [27]. Forward simulations combine refinement mappings with history variables. Typical examples of their use appear in [19, 31, 34, 29, 42]. Bisimulations, which combine in a single relation forward simulations in two directions, play a vital role in the theory of process algebras [44, 41, 4]. Backward simulations have so far been much less widely used. Abadi and Lamport [1] demonstrate the usefulness of prophecy variables (and hence backward simulations), with some simple examples, while [29] contains a somewhat more practical example. There has not been much work on applying the hybrid forward and backward methods.

We consider the main contribution of this paper to be the unified presentation, in terms of a simple and general automaton model, of a wide range of important simulation techniques, together with their basic soundness and completeness properties. Some features of our presentation are: (a) It parallels and supports a similar development for timed systems. (b) We present the simulations in a "bottom-up" order, starting with simple ones such as forward and backward simulations and building up to more complicated simulations such as forward-backward simulations and history relations. The proofs of many of the results for complicated simulations rest on the results for the simpler simulations. (c) We separate out the treatment of invariants. We make no mention of invariants (or even of state reachability) in our main development, but only incorporate them at the end. The results involving invariants can be proved using the results without invariants.

In addition, there are several new definitions and theorems, notably, (a) the abstract definitions of history and prophecy relations, and the accompanying characterization and completeness theorems, and (b) the definition and properties of backward–forward simulations.

The rest of this paper is organized as follows. Section 2 contains basic definitions and results for untimed automata. Section 3 contains the development of the basic simulation techniques: refinements, forward simulations and backward simulations. Section 4 contains the development of the hybrid techniques: forward–backward and backward–forward simulations. Section 5 contains the results on history and prophecy relations. Section 6 shows how invariants can be included in the simulations. Section 7 contains some conclusions and a discussion of related work. Finally, Appendix A contains some mathematical preliminaries, and Appendix B gives a glossary of conventions followed in the paper.

2. UNTIMED AUTOMATA AND THEIR BEHAVIORS

In this section, we present the basic definitions and results for untimed automata. We also define certain restricted kinds of automata that are useful in our proofs, and define various sets of traces that automata can generate.

2.1. *Automata*

We begin with the definition of an (untimed) automaton. An *automaton* A consists of:

- a set $states(A)$ of states,
- a nonempty set $start(A) \subseteq states(A)$ of start states,
- a set $acts(A)$ of actions that includes a special element τ, and
- a set $steps(A) \subseteq states(A) \times acts(A) \times states(A)$ of steps.

All these components should be completely self-explanatory.

We let s, s', u, u', ... range over states, and a, ... over actions. We let $ext(A)$, the *external actions*, denote $acts(A) - \{\tau\}$. We call τ the *internal action*. The term *event* refers to an occurrence of an action in a sequence. If γ is a sequence of actions then $\hat{\gamma}$ is the sequence obtained by deleting all τ events from γ. We write $s' \xrightarrow{a}_A s$, or just $s' \xrightarrow{a} s$ if A is clear from the context, as a shorthand for $(s', a, s) \in steps(A)$. In this paper (Part I), $A, B, ...$ range over automata.

An *execution fragment* of A is a finite or infinite alternating sequence, $s_0 a_1 s_1 a_2 s_2 \cdots$, of states and actions of A, beginning with a state, and if it is finite also ending with a state, such that for all i, $s_i \xrightarrow{a_{i+1}} s_{i+1}$. We denote by $frag^*(A)$, $frag^\omega(A)$, and $frag(A)$ the sets of finite, infinite, and all execution fragments of A, respectively. An *execution* of A is an execution fragment that begins with a start state. We denote by $execs^*(A)$, $execs^\omega(A)$, and $execs(A)$ the sets of finite, infinite, and all executions of A, respectively. A state s of A is *reachable* if $s = last(\alpha)$ for some finite execution α of A.

Suppose $\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ is an execution fragment of A. Let γ be the sequence consisting of the actions in α: $\gamma = a_1 a_2 \cdots$. Then $trace(\alpha)$ is defined to be the sequence $\hat{\gamma}$. A finite or infinite sequence β of external actions is a *trace* of A if A has an execution α with $\beta = trace(\alpha)$. We write $traces^*(A)$, $traces^{\omega}(A)$, and $traces(A)$ for the sets of finite, infinite and all traces of A, respectively. These notions induce three *preorders* (i.e., reflexive and transitive relations). For A and B automata, we define $A \leqslant_{*\mathrm{T}} B \triangleq traces^*(A) \subseteq traces^*(B)$, $A \leqslant_{\omega\mathrm{T}} B \triangleq traces^{\omega}(A) \subseteq traces^{\omega}(B)$, and $A \leqslant_{\mathrm{T}} B \triangleq traces(A) \subseteq traces(B)$. Recall that the *kernel* of a preorder $\sqsubseteq$ is the equivalence $\equiv$ defined by $x \equiv y \triangleq x \sqsubseteq y \wedge y \sqsubseteq x$. We denote by $\equiv_{*\mathrm{T}}$, $\equiv_{\omega\mathrm{T}}$, and $\equiv_{\mathrm{T}}$ the respective kernels of the preorders $\leqslant_{*\mathrm{T}}$, $\leqslant_{\omega\mathrm{T}}$, and $\leqslant_{\mathrm{T}}$.

Suppose A is an automaton, s' and s are states of A, and β is a finite sequence over $ext(A)$. We say that (s', β, s) is a *move* of A, and write $s' \overset{\beta}{\Longrightarrow}_A s$, or just $s' \overset{\beta}{\Longrightarrow} s$ when A is clear, if A has a finite execution fragment α with $first(\alpha) = s'$, $trace(\alpha) = \beta$ and $last(\alpha) = s$.

Example 2.1. The automata A_1 and A_2 of Fig. 1 illustrate the difference between $\leqslant_{*\mathrm{T}}$ and $\leqslant_{\mathrm{T}}$. Each has a linear sequence of states. A_1 has a single start state, and a step from each state to its right neighbor, while A_2 has all states as start states, and a step from each state to its left neighbor. Every finite sequence of a's is a trace of each of A_1 and A_2; in addition, the sequence consisting of infinitely many a's is a trace of A_1 but not of A_2. Therefore, $A_1 \equiv_{*\mathrm{T}} A_2$, $A_2 \leqslant_{\mathrm{T}} A_1$, and $A_1 \nleqslant_{\mathrm{T}} A_2$.

2.2. *Restricted Kinds of Automata*

Now we describe three restricted kinds of automata that are useful in our proofs.

First, automaton A is *deterministic* if $|start(A)| = 1$, and for any state s' and any finite sequence β over $ext(A)$, there is at most one state s such that $s' \overset{\beta}{\Longrightarrow} s$. A deterministic automaton is characterized uniquely by the properties that $|start(A)| = 1$, every τ step is of the form (s, τ, s) for some s, and for all states s' and all actions a there is at most one state s such that $s' \xrightarrow{a}_A s$.

Second, A has *finite invisible nondeterminism* (*fin*) if $start(A)$ is finite, and for any state s' and any finite sequence β over $ext(A)$, there are only finitely many states s such that $s' \overset{\beta}{\Longrightarrow}_A s$.

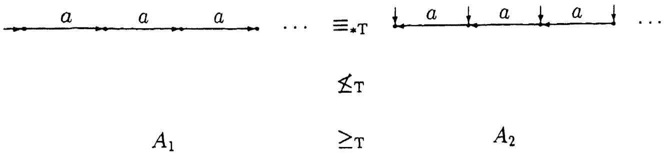

FIG. 1. $\leqslant_{*\mathrm{T}}$ versus $\leqslant_{\mathrm{T}}$.

Third, A is a *forest* if, for each state s of A, there is a unique execution that leads to s. A forest is characterized uniquely by the property that all states of A are reachable, start states have no incoming steps and each of the other, states has exactly one incoming step.

The relation $after(A)$ consists of the pairs (β, s) for which there is a finite execution of A with trace β and last state s:

$$after(A) \triangleq \{(\beta, s) \mid \exists \alpha \in execs^*(A): \\ trace(\alpha) = \beta \text{ and } last(\alpha) = s\}.$$

We also define $past(A)$ to be the inverse of $after(A)$, $past(A) \triangleq after(A)^{-1}$; this relates a state s of A to the traces of finite executions of A that lead to s.

Lemma 2.2.

1. *If A is deterministic then $after(A)$ is a function from $traces^*(A)$ to $states(A)$.*
2. *If A has fin then $after(A)$ is image-finite.*[1]
3. *If A is a forest then $past(A)$ is a function from $states(A)$ to $traces^*(A)$.*

Example 2.3. In Fig. 1, automaton A_1 is deterministic (and so has fin), and is a forest. Automaton A_2 has none of these three properties.

2.3. *Trace Properties*

In this subsection, we define "trace properties," the structures that are used as external behaviors for automata. We also prove some basic properties of trace properties and some lemmas relating trace properties to automata.

A *trace property* P is a pair (K, L), where K is a set and L is a nonempty, prefix closed set of (finite or infinite) sequences over K. We will refer to the constituents of P as $sort(P)$ and $traces(P)$, respectively. Also, we write $traces^*(P) \triangleq K^* \cap L$ and $traces^{\omega}(P) \triangleq K^{\omega} \cap L$. For P and Q trace properties, we define $P \leqslant_{*\mathrm{T}} Q \triangleq traces^*(P) \subseteq traces^*(Q)$, $P \leqslant_{\omega\mathrm{T}} Q \triangleq traces^{\omega}(P) \subseteq traces^{\omega}(Q)$, and $P \leqslant_{\mathrm{T}} Q \triangleq traces(P) \subseteq traces(Q)$. With $\equiv_{*\mathrm{T}}$, $\equiv_{\omega\mathrm{T}}$, and $\equiv_{\mathrm{T}}$, we denote the kernels of the preorders $\leqslant_{*\mathrm{T}}$, $\leqslant_{\omega\mathrm{T}}$, and $\leqslant_{\mathrm{T}}$, respectively. A trace property P is *limit-closed* if an infinite sequence is in $traces(P)$ whenever all its finite prefixes are.

Lemma 2.4. *Suppose P and Q are trace properties with Q limit-closed. Then $P \leqslant_{*\mathrm{T}} Q \Leftrightarrow P \leqslant_{\mathrm{T}} Q$.*

The *behavior* of an automaton A, $beh(A)$, is defined by $beh(A) \triangleq (ext(A), traces(A))$.

[1] See Appendix A for the definition of image-finite.

LEMMA 2.5.

1. *$beh(A)$ is a trace property.*
2. *If A has fin then $beh(A)$ is limit-closed.*
3. *$A \leqslant_{*\mathrm{T}} B \Leftrightarrow beh(A) \leqslant_{*\mathrm{T}} beh(B)$, $A \leqslant_{\omega\mathrm{T}} B \Leftrightarrow beh(A) \leqslant_{\omega\mathrm{T}} beh(B)$, and $A \leqslant_{\mathrm{T}} B \Leftrightarrow beh(A) \leqslant_{\mathrm{T}} beh(B)$.*

Proof. It is easy to see that $beh(A)$ is a trace property.

For Part 2, suppose A has fin. We use Lemma A.1 to show that $beh(A)$ is limit-closed. Suppose β is an infinite sequence over $ext(A)$ such that all finite prefixes of β are in $traces(A)$. Consider the digraph G whose nodes are pairs $(\gamma, s) \in after(A)$, where γ is a finite prefix of β; there is an edge from node (γ', s') to node (γ, s) exactly if γ is of the form $\gamma' a$, where $a \in ext(A)$, and where $s' \overset{a}{\Longrightarrow}_A s$. Then G satisfies the hypotheses of Lemma A.1, which implies that there is an infinite path in G starting at a root. This corresponds directly to an execution α having $trace(\alpha) = \beta$. Hence, $\beta \in traces(A)$.

Part 3 is immediate from the definitions. ▮

PROPOSITION 2.6. *If B has fin then $A \leqslant_{*\mathrm{T}} B \Leftrightarrow A \leqslant_{\mathrm{T}} B$.*

Proof. Immediate from Lemma 2.4 and Lemma 2.5. ▮

EXAMPLE 2.7. Recall that, in Fig. 1, $A_1 \leqslant_{*\mathrm{T}} A_2$ but $A_1 \not\leqslant_{\mathrm{T}} A_2$. This is consistent with Lemma 2.6, because A_2 does not have fin.

We close this section with the construction of the *canonical automaton*[2] for a given trace property. For P a trace property, the associated *canonical* automaton $can(P)$ is the structure A given by

- $states(A) = traces^*(P)$,
- $start(A) = \{\lambda\}$,
- $acts(A) = sort(P) \cup \{\tau\}$, and
- for $\beta', \beta \in states(A)$ and $a \in acts(A)$, $\beta' \xrightarrow{a}_A \beta \Leftrightarrow a \in ext(A) \wedge \beta' a = \beta$.

LEMMA 2.8.

1. *$can(P)$ is a deterministic forest.*
2. *$beh(can(P)) \equiv_{*\mathrm{T}} P$.*
3. *$beh(can(P)) \geqslant_{\mathrm{T}} P$.*
4. *If P is limit-closed then $beh(can(P)) \equiv_{\mathrm{T}} P$.*

Proof. Parts 1 and 2 follow easily from the definitions. Since $can(P)$ is deterministic it certainly has fin, so it follows by Lemma 2.5 that $beh(can(P))$ is limit-closed. Now 3 and 4 follow by combination of 2 and Lemma 2.4. ▮

LEMMA 2.9.

1. *$can(beh(A))$ is a deterministic forest.*
2. *$can(beh(A)) \equiv_{*\mathrm{T}} A$.*
3. *$can(beh(A)) \geqslant_{\mathrm{T}} A$.*
4. *If A has fin then $can(beh(A)) \equiv_{\mathrm{T}} A$.*

Proof. By combining Lemma 2.5 and Lemma 2.8. ▮

3. BASIC SIMULATIONS

In this section, we develop the basic simulation techniques for untimed automata: refinements and forward and backward simulations.

3.1. *Refinements*

The simplest type of simulation we consider is a *refinement*. A *refinement* from A to B is a function r from states of A to states of B that satisfies the following two conditions:

1. If $s \in start(A)$ then $r(s) \in start(B)$.
2. If $s' \xrightarrow{a}_A s$ then $r(s') \overset{\hat{a}}{\Longrightarrow}_B r(s)$.

We write $A \leqslant_{\mathrm{R}} B$ if there exists a refinement from A to B.

This notion is similar to that of a *homomorphism* in classical automata theory; see for instance Ginzberg [10]. Besides our additional treatment of internal actions, a difference between the two notions is that the classical notion involves a mapping between the action sets of the automata, whereas our refinements do not.

EXAMPLE 3.1. Figure 2 presents some examples of automata that are and are not related by $\leqslant_{\mathrm{R}}$. Automata A_3 and A_4 have the same traces, $A_3 \leqslant_{\mathrm{R}} A_4$ and $A_4 \not\leqslant_{\mathrm{R}} A_3$. Likewise, automata A_5 and A_6 have the same traces, $A_5 \leqslant_{\mathrm{R}} A_6$ and $A_6 \not\leqslant_{\mathrm{R}} A_5$.

The following technical lemma is a straightforward consequence of the definition of a refinement.

LEMMA 3.2. *Suppose r is a refinement from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. Then $r(s') \overset{\beta}{\Longrightarrow}_B r(s)$.*

PROPOSITION 3.3. *$\leqslant_{\mathrm{R}}$ is a preorder (i.e., is transitive and reflexive).*

Proof. The identity function $id(states(A))$ is a refinement from A to itself. This implies that $\leqslant_{\mathrm{R}}$ is reflexive. Using Lemma 3.2, transitivity follows from the observation that if r is a refinement from A to B and r' is a refinement from B to C, then $r' \circ r$[3] is a refinement from A to C. ▮

The important property of refinements for verification is that they are sound for the trace inclusion preorder.

THEOREM 3.4 (Soundness of Refinements). *$A \leqslant_{\mathrm{R}} B \Rightarrow A \leqslant_{\mathrm{T}} B$.*

Proof. Suppose $A \leqslant_{\mathrm{R}} B$. Let r be a refinement from A to B, and let e be a function that maps each move (s', β, s) of B to a finite execution fragment of B from s' to s with trace β. Suppose $\beta \in traces(A)$. Then there exists an execution

[2] This notion is due to He Jifeng [13].

[3] See Appendix A for the definition of the composition operator.

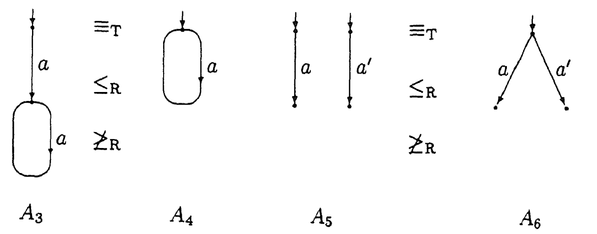

FIG. 2. Refinements.

$\alpha = s_0 a_1 s_1 a_2 s_2 \cdots$ of A with $\beta = trace(\alpha)$. By the first condition in the definition of a refinement, $r(s_0)$ is a start state of B, and by the second condition, $r(s_i) \overset{\widehat{a_{i+1}}}{\Longrightarrow}_B r(s_{i+1})$ for all i. For $i \geqslant 0$, define $\alpha_i = e((r(s_i), \widehat{a_{i+1}}, r(s_{i+1})))$. Next define sequence α' to be the (infinitary) concatenation $\alpha_0\ tail(\alpha_1)\ tail(\alpha_2) \cdots$. By construction, α' is an execution of B with $trace(\alpha') = trace(\alpha) = \beta \in traces(B)$. ▌

Refinements alone are not complete for $\leqslant_{\mathrm{T}}$ or $\leqslant_{*\mathrm{T}}$. We do have a (very) partial completeness result, however, which slightly generalizes a similar result of [25] in that it also allows for τ-steps in the A automaton.

THEOREM 3.5 (Partial Completeness of Refinements). *Suppose A is a forest, B is deterministic, and $A \leqslant_{*\mathrm{T}} B$. Then $A \leqslant_{\mathrm{R}} B$.*

Proof. The relation $r \triangleq after(B) \circ past(A)$ is a refinement from A to B. ▌

3.2. *Forward Simulations*

A *forward simulation* from A to B is a relation f over $states(A)$ and $states(B)$ that satisfies:

1. If $s \in start(A)$ then $f[s] \cap start(B) \neq \varnothing$.
2. If $s' \xrightarrow{a}_A s$ and $u' \in f[s']$, then there exists a state $u \in f[s]$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leqslant_{\mathrm{F}} B$ if there exists a forward simulation from A to B.

EXAMPLE 3.6. Let A_3, A_4, A_5, A_6 be as in Fig. 2. Then $A_4 \leqslant_{\mathrm{F}} A_3$ and $A_6 \not\leqslant_{\mathrm{F}} A_5$.

PROPOSITION 3.7. $A \leqslant_{\mathrm{R}} B \Rightarrow A \leqslant_{\mathrm{F}} B$.

Proof. Any refinement relation is a forward simulation. ▌

The following lemma is the analogue of Lemma 3.2 for forward simulations.

LEMMA 3.8. *Suppose f is a forward simulation from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. If $u' \in f[s']$, then there exists a state $u \in f[s]$ such that $u' \overset{\beta}{\Longrightarrow}_B u$.*

PROPOSITION 3.9. *$\leqslant_{\mathrm{F}}$ is a preorder.*

Proof. For reflexivity, observe that the identity function $id(states(A))$ is a forward simulation from A to itself. For transitivity, use Lemma 3.8 to show that if f and f' are forward simulations from A to B and from B to C, respectively, $f' \circ f$ is a forward simulation from A to C. ▌

THEOREM 3.10 (Soundness of Forward Simulations [35, 20, 47]). $A \leqslant_{\mathrm{F}} B \Rightarrow A \leqslant_{\mathrm{T}} B$.

Proof. Versions of this proof appear in the cited papers. The proof is similar to that of Theorem 3.4. ▌

Also, the following result is well known and variants of it have appeared in many papers (for instance, in [19, 47]).

THEOREM 3.11 (Partial Completeness of Forward Simulations). *Suppose B is deterministic and $A \leqslant_{*\mathrm{T}} B$. Then $A \leqslant_{\mathrm{F}} B$.*

Proof. The relation $f \triangleq after(B) \circ past(A)$ is a forward simulation from A to B. ▌

The following Proposition 3.12 is mainly of technical interest; in particular, it is the only one of our results for which we have not been able to prove an analogue in the timed case. It might also have some implications for verification: if one guesses that a relation f is a forward simulation from a forest A to an automaton B, then one might try to restrict f to a refinement r. Since such a refinement must exist (if f is in fact a forward simulation) and since refinements are usually easier to verify than forward simulations, this may lead to a simpler proof.

PROPOSITION 3.12. *Suppose A is a forest and $A \leqslant_{\mathrm{F}} B$. Then $A \leqslant_{\mathrm{R}} B$.*

Proof. Let f be a forward simulation from A to B. We construct a choice function r for f, and prove that r is a refinement from A to B.

For $n \geqslant 0$, let $Layer_n$ be the set of states s of A for which the (unique) execution leading to it contains n actions. Then the sets $Layer_n$ ($n \geqslant 0$) partition the set $states(A)$ and $Layer_0 = start(A)$. We define functions r_n: $Layer_n \rightarrow states(B)$ induc tively such that $r_n(s) \in f[s]$. By Condition 1 in the definitio of a forward simulation, there exists a function r_0: $Layer_0$ – $start(B)$ satisfying $r_0(s) \in f[s]$. Suppose that r_i has bee. defined for $i \leqslant n$. By Condition 2 in the definition of a forwar(simulation, there exists a function r_{n+1}: $Layer_{n+1} \rightarrow states(B)$ such that if s is in $Layer_{n+1}$ and $s' \xrightarrow{a}_A s$ is th unique incoming step of s, we have $r_n(s') \overset{\hat{a}}{\Longrightarrow}_B r_{n+1}(s)$ an($r_{n+1}(s) \in f[s]$. By construction, the union r of the function r_n is a refinement from A to B with $r(s) \in f[s]$. ▌

Proposition 3.12 allows us to give an alternative proc of the partial completeness result for refinement (Theorem 3.5): if A is a forest, B is deterministic ar $A \leqslant_{*\mathrm{T}} B$, then $A \leqslant_{\mathrm{F}} B$ by Theorem 3.11, and then $A \leqslant_{\mathrm{R}}$ follows using Prop. 3.12.

3.3. *Backward Simulations*

In many respects, backward simulations are the dual of forward simulations. Whereas a forward simulation requires that *some* state in the image of each start state should be a start state, a backward simulation requires that *all* states in the image of a start state be start states. Also, a forward simulation requires that forward steps in the source automaton can be simulated from related states in the target automaton, whereas the corresponding condition for a backward simulation requires that backward steps can be simulated. However, the two notions are not completely dual: the definition of a backward simulation contains a nonemptiness condition, and also, in order to imply soundness in general, backward simulations also require a finite image condition. The mismatch is due to the asymmetry in our automata between the future and the past: from any given state, all the possible histories are finite executions, whereas the possible futures can be infinite.

A *backward simulation* from A to B is a total[4] relation b over $states(A)$ and $states(B)$ that satisfies:

1. If $s \in start(A)$ then $b[s] \subseteq start(B)$.
2. If $s' \xrightarrow{a}_A s$ and $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leqslant_{\mathrm{B}} B$ if there exists a backward simulation from A to B, and $A \leqslant_{\mathrm{iB}} B$ if there exists an image-finite backward simulation from A to B.

EXAMPLE 3.13. Let A_1, A_2 be as in Fig. 1. Then $A_1 \leqslant_{\mathrm{B}} A_2$ but $A_1 \not\leqslant_{\mathrm{iB}} A_2$. If A_3, A_4, A_5, A_6 are as in Fig. 2, then $A_4 \not\leqslant_{\mathrm{B}} A_3$ and $A_6 \leqslant_{\mathrm{iB}} A_5$.

PROPOSITION 3.14. $A \leqslant_{\mathrm{R}} B \Rightarrow A \leqslant_{\mathrm{iB}} B$.

The following lemma is useful in the proofs of the preorder properties and of soundness.

LEMMA 3.15. *Suppose b is a backward simulation from A to B and $s' \overset{\beta}{\Longrightarrow}_A s$. If $u \in b[s]$, then there exists a state $u' \in b[s']$ such that $u' \overset{\beta}{\Longrightarrow}_B u$.*

PROPOSITION 3.16. *$\leqslant_{\mathrm{B}}$ and $\leqslant_{\mathrm{iB}}$ are preorders.*

Proof. The identity function $id(states(A))$ is a backward simulation from A to itself. Using Lemma 3.15 one can easily show that if b is a backward simulation from A to B and b' is a backward simulation from B to C, $b' \circ b$ is a backward simulation from A to C. Moreover, if both b and b' are image-finite, then $b' \circ b$ is image-finite too. ▌

THEOREM 3.17 (Soundness of Backward Simulations).

1. $A \leqslant_{\mathrm{B}} B \Rightarrow A \leqslant_{*\mathrm{T}} B$.
2. $A \leqslant_{\mathrm{iB}} B \Rightarrow A \leqslant_{\mathrm{T}} B$.

Proof. Suppose b is a backward simulation from A to B and suppose $\beta \in traces^*(A)$. Then there is a move $s' \overset{\beta}{\Longrightarrow}_A s$, where s' is a start state of A. Since b is a backward simulation it is a total relation, so there exists a state $u \in b[s]$. By Lemma 3.15, there exists $u' \in b[s']$ with $u' \overset{\beta}{\Longrightarrow}_B u$. By the first condition of the definition of a backward simulation, $u' \in start(B)$. Therefore, $\beta \in traces^*(B)$, which shows the first part of the proposition.

For the second part, suppose that b is image-finite. We have already established $A \leqslant_{*\mathrm{T}} B$, so it is sufficient to show $A \leqslant_{\omega\mathrm{T}} B$. Suppose that $\beta \in traces^{\omega}(A)$, and let $\alpha = s_0 a_1 s_1 a_2 \cdots$ be an infinite execution of A with $trace(\alpha) = \beta$.

Consider the digraph G whose nodes are pairs (u, i) such that $(s_i, u) \in b$ and in which there is an edge from (u', i') to (u, i) exactly if $i = i' + 1$ and $u' \overset{\hat{a_i}}{\Longrightarrow}_B u$. Then G satisfies the hypotheses of Lemma A.1, which implies that there is an infinite path in G starting at a root. This corresponds directly to an execution α' of B having $trace(\alpha') = trace(\alpha) = \beta$. Hence, $\beta \in traces(B)$. ▌

Jonsson [22] considers a weaker image-finiteness condition for backward simulations. Translated into our setting, the key observation of Jonsson is that in order to prove $A \leqslant_{\mathrm{T}} B$, it is enough to give a backward simulation b from A to B with the property that each infinite execution of A contains infinitely many states s with $b[s]$ finite. We do not explore this extension in this paper, primarily because it lacks a key feature of simulation techniques. Namely, it fails to reduce reasoning about executions to reasoning about individual states and steps.

The following partial completeness result slightly generalizes a similar result of Jonsson [21] in that it also allows for τ-steps in the B automaton.

THEOREM 3.18 (Partial Completeness of Backward Simulations). *Suppose A is a forest and $A \leqslant_{*\mathrm{T}} B$. Then*

1. *$A \leqslant_{\mathrm{B}} B$, and*
2. *if B has fin then $A \leqslant_{\mathrm{iB}} B$.*

Proof. We define a relation b over $states(A)$ and $states(B)$. Suppose s is a state of A. Since A is a forest there is a unique trace leading up to s, say β. Now define

$$b[s] = \{u \mid \exists \alpha \in execs^*(B) : trace(\alpha) = \beta \wedge last(\alpha) = u \wedge \forall \alpha' \in execs^*(B) : [\alpha' < \alpha \Rightarrow trace(\alpha') \neq \beta]\}.$$

By letting $b[s]$ consist only of those states of B which can be reached via a *minimal* execution with trace β, we achieve that, if s is a start state, all the states in $b[s]$ are start states of B. It is also the case that b satisfies the other conditions in the definition of a backward simulation.

Lemma 2.2 implies that b is image-finite if B has fin. ▌

[4] See Appendix A for the definition of a total relation.

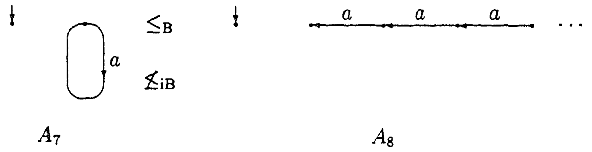

FIG. 3. $\leqslant_{\mathrm{B}}$ and $\leqslant_{\mathrm{iB}}$ are different, even for automata with *fin*.

The next proposition is the dual of Prop. 3.12, and provides us with yet another proof of the partial completeness result for refinements (Theorem 3.5), now using Theorem 3.18. Unlike Prop. 3.12, Prop. 3.19 does have an analogue in the timed case.

PROPOSITION 3.19. *Suppose all states of A are reachable, B is deterministic and $A \leqslant_{\mathrm{B}} B$. Then $A \leqslant_{\mathrm{R}} B$.*

Proof. Let b be a backward simulation from A to B and let s be a reachable state of A. We will prove that $b[s]$ contains exactly one element. Because all states of A are reachable, it follows that b is functional. But any functional backward simulation trivially is a refinement, and so we obtain $A \leqslant_{\mathrm{R}} B$.

Since b is a backward simulation, it is a total relation, so we know $b[s]$ contains at least one element. Suppose that both $u_1 \in b[s]$ and $u_2 \in b[s]$; we prove $u_1 = u_2$. Since s is reachable, there exists a start state s' and a trace β such that $s' \overset{\beta}{\Longrightarrow}_A s$. By Lemma 3.15, there exist states $u'_1, u'_2 \in b[s']$ such that $u'_1 \overset{\beta}{\Longrightarrow}_B u_1$ and $u'_2 \overset{\beta}{\Longrightarrow}_B u_2$. Since b is a backward simulation and s' is a start state of A, u'_1 and u'_2 are start states of B. But B is deterministic and deterministic automata have only a single start state so $u'_1 = u'_2$. Now the fact that B is deterministic also implies $u_1 = u_2$. ▮

The following proposition is mainly of technical interest. It is used as a lemma in the technical report version of this paper [37] to complete the classification of *weak* simulations (see Section 6).

PROPOSITION 3.20. *Suppose all states of A are reachable, B has fin and $A \leqslant_{\mathrm{B}} B$. Then $A \leqslant_{\mathrm{iB}} B$.*

Proof. Let b be a backward simulation from A to B and let s be a state of A. Since s is reachable we can find a trace $\beta \in past(A)[s]$. From the fact that b is a backward simulation it follows that $b[s] \subseteq after(B)[\beta]$. But since B has fin, $after(B)[\beta]$ is finite by Lemma 2.2. This implies that b is image-finite. ▮

EXAMPLE 3.21. Figure 3 shows that the reachability assumptions in Propositions 3.19 and 3.20 are essential. There is a backward simulation from A_7 to A_8, but even though A_8 is deterministic there is no image-finite backward simulation.

3.4. *Combined Forward and Backward Simulations*

Several authors have observed that forward and backward simulations together give a complete proof method for $\leqslant_{*\mathrm{T}}$ (see [17, 13, 21, 22, 23, 25]): if $A \leqslant_{*\mathrm{T}} B$ then there exists an intermediate automaton C with a forward simulation from A to C and a backward simulation from C to B. We prove this below by taking C to be the canonical automaton of A, as defined in Section 2. Alternative proofs can be given using different intermediate automata, for example the automaton obtained by applying the classical subset construction on B (see [22, 25]) or the unfolding construction of Section 5.1 on A.

THEOREM 3.22 (Completeness of Forward and Backward Simulations). *If $A \leqslant_{*\mathrm{T}} B$ then the following are true.*

1. $\exists C\colon A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{B}} B$.
2. *If B has fin then* $\exists C\colon A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{iB}} B$.

Proof. Take $C = can(beh(A))$. By Lemma 2.9, C is a deterministic forest and $A \equiv_{*\mathrm{T}} C$. Since C is deterministic, $A \leqslant_{\mathrm{F}} C$ by Theorem 3.11, and because C is a forest, $C \leqslant_{\mathrm{B}} B$ follows by Theorem 3.18(1). If B has fin then $C \leqslant_{\mathrm{iB}} B$ follows by Theorem 3.18(2). ▮

4. HYBRID SIMULATIONS

4.1. *Forward–Backward Simulations*

Forward–backward simulations were introduced by Klarlund and Schneider, who call them *invariants* in [24] and *ND measures* in [25]. They also occur in the work of Jonsson [22] under the name *subset simulations*, and are related to the *failure simulations* of Gerth [9]. Forward–backward simulations combine in a single relation both a forward and a backward simulation. Below we present simple proofs of their soundness and completeness by making this connection explicit.

Formally, a *forward–backward simulation* from A to B is a relation g over $states(A)$ and $\mathrm{N}(states(B))$ that satisfies:[5]

1. If $s \in start(A)$ then there exists $S \in g[s]$ such that $S \subseteq start(B)$.
2. If $s' \xrightarrow{a}_A s$ and $S' \in g[s']$, then there exists a set $S \in g[s]$ such that for every $u \in S$ there exists $u' \in S'$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leqslant_{\mathrm{FB}} B$ if there exists a forward–backward simulation from A to B, and $A \leqslant_{\mathrm{iFB}} B$ if there exists an image-set-finite forward–backward simulation from A to B.

The following theorem, which is similar to a result of [22], says that a forward–backward simulation is essentially just a combination of a forward and a backward simulation.

THEOREM 4.1.

1. $A \leqslant_{\mathrm{FB}} B \Leftrightarrow (\exists C\colon A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{B}} B)$.
2. $A \leqslant_{\mathrm{iFB}} B \Leftrightarrow (\exists C\colon A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{iB}} B)$.

[5] The N( ) notation is defined in Appendix A.

Proof. "$\Rightarrow$" Let g be a forward–backward simulation from A to B, which is image-set-finite if $A \leqslant_{\mathrm{iFB}} B$. Define C to be the automaton given by:

- $states(C) = range(g)$,
- $start(C) = range(g) \cap \mathbf{P}(start(B))$,
- $acts(C) = acts(B)$, and
- for $S', S \in states(C)$ and $a \in acts(C)$, $S' \xrightarrow{a}_C S \Leftrightarrow \forall u \in S$: $\exists u' \in S'$: $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Then g is a forward simulation from A to C. Also, $\{(S, u) \mid S \in states(C) \text{ and } u \in S\}$ is a backward simulation from C to B, which is image finite if g is image-set-finite.

"$\Leftarrow$" Suppose f is a forward simulation from A to C, and b is a backward simulation from C to B. Then the relation g over $states(A)$ and $\mathrm{N}(states(B))$ defined by $g = \{(s, b[u]) \mid (s, u) \in f\}$ is a forward–backward simulation from A to B. If b is image-finite then g is image-set-finite. ▮

PROPOSITION 4.2.

1. $A \leqslant_{\mathrm{F}} B \Rightarrow A \leqslant_{\mathrm{iFB}} B$.
2. $A \leqslant_{\mathrm{B}} B \Rightarrow A \leqslant_{\mathrm{FB}} B$.
3. $A \leqslant_{\mathrm{iB}} B \Rightarrow A \leqslant_{\mathrm{iFB}} B$.

Proof. Immediate from Theorem 4.1, using that $\leqslant_{\mathrm{iB}}$ and $\leqslant_{\mathrm{F}}$ are reflexive. ▮

In order to show that $\leqslant_{\mathrm{FB}}$ and $\leqslant_{\mathrm{iFB}}$ are preorders, we require a definition of composition for forward–backward simulations, and a transitivity lemma.

If g is a relation over X and $\mathrm{N}(Y)$ and g' is a relation over Y and $\mathrm{N}(Z)$ then the composition $g' \bullet g$ is a relation over X and $\mathrm{N}(Z)$ defined as follows.

$$(x, S') \in g' \bullet g \Leftrightarrow \exists S \in g[x] \colon \exists c \in S \to \mathrm{N}(Z) \colon$$
$$\left(c \subseteq g' \wedge S' = \bigcup \{c(y) \mid y \in S\} \right).$$

Note that in the above definition c is a choice function for $g' \lceil S$. The nonemptiness assumptions for g and g' immediately imply the nonemptiness assumption for $g' \bullet g$.

LEMMA 4.3. *Suppose g is a forward–backward simulation from A to B, and g' is a forward–backward simulation from B to C. Then $g' \bullet g$ is a forward–backward simulation from A to C. Moreover, if g and g' are image-set-finite then $g' \bullet g$ is also image-set-finite.*

Proof. For Condition 1 of the definition of a forward–backward simulation, suppose $s \in start(A)$. Because g is a forward–backward simulation, there is a set $S \in g[s]$ with $S \subseteq start(B)$. Since g' is a forward–backward simulation, it is possible to find, for each $u \in S$, a set $S_u \in g'[u]$ with $S_u \subseteq start(C)$. Hence all states in the set $S' = \bigcup \{S_u \mid u \in S\}$ are start states of C. Now let c be the function with domain S given by $c(u) = S_u$. Then c is a choice function for $g' \lceil S$. From the definition of $\bullet$ it now follows that $(s, S') \in g' \bullet g$. This shows that $g' \bullet g$ satisfies Condition 1.

Now we show Condition 2 of the definition of a forward–backward simulation. Suppose $s' \xrightarrow{a}_A s$ and $(s', S') \in g' \bullet g$. By definition of $g' \bullet g$, there exist $U' \in g[s']$ and a choice function c' for $g' \lceil U'$ such that $S' = \bigcup \{c'(u') \mid u' \in U'\}$. Because g is a forward–backward simulation from A to B, there is a set $U \in g[s]$ such that for each $u \in U$ there exists $u' \in U'$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$. Consider any particular $u \in U$. Choose $u' \in U'$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$. Because g' is a forward–backward simulation, there exists a set $S_u \in g'[u]$ such that for every $v \in S_u$ there exists a $v' \in c'(u')$ with $v' \overset{\hat{a}}{\Longrightarrow}_C v$. Define a choice function c for $g' \lceil U$ by taking $c(u)$ to be the set S_u.

Now consider the set $S = \bigcup \{c(u) \mid u \in U\}$. Then $(s, S) \in g' \bullet g$ by definition. By construction, we can find, for each $v \in S$, a state $v' \in S'$ with $v' \overset{\hat{a}}{\Longrightarrow}_C v$. Thus S has the required property to show Condition 2.

Finally, it is immediate from the definitions that, if g and g' are image-set-finite, $g' \bullet g$ is also image-set-finite. ▮

PROPOSITION 4.4. *$\leqslant_{\mathrm{FB}}$ and $\leqslant_{\mathrm{iFB}}$ are preorders.*

Proof. By Lemma 4.3. ▮

THEOREM 4.5 (Soundness of Forward–Backward Simulations, [24]).

1. $A \leqslant_{\mathrm{FB}} B \Rightarrow A \leqslant_{*\mathrm{T}} B$.
2. $A \leqslant_{\mathrm{iFB}} B \Rightarrow A \leqslant_{\mathrm{T}} B$.

Proof. For part 1, suppose $A \leqslant_{\mathrm{FB}} B$. By Theorem 4.1, there exists an automaton C with $A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{B}} B$. By soundness of forward simulations, Theorem 3.10, $A \leqslant_{\mathrm{T}} C$, and by soundness of backward simulations, Theorem 3.17, $C \leqslant_{*\mathrm{T}} B$. This implies $A \leqslant_{*\mathrm{T}} B$. Part 2 is similar. ▮

THEOREM 4.6 (Completeness of Forward–Backward Simulations [24]). *Suppose $A \leqslant_{*\mathrm{T}} B$. Then*

1. *$A \leqslant_{\mathrm{FB}} B$, and*
2. *if B has fin then $A \leqslant_{\mathrm{iFB}} B$.*

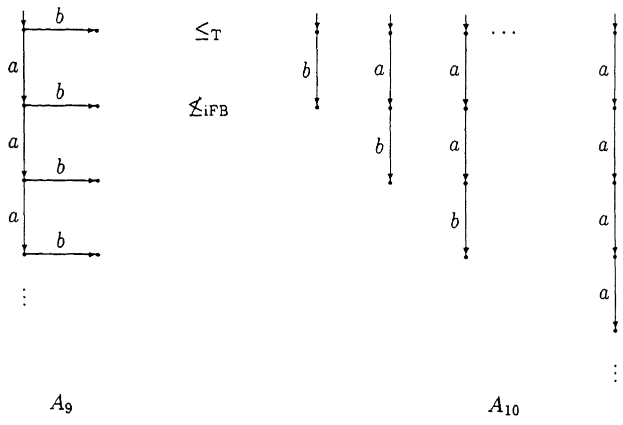

FIG. 4. The difference between $\leqslant_{\mathrm{T}}$ and $\leqslant_{\mathrm{iFB}}$.

Proof. By Theorem 3.22, there exists an automaton C with $A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{B}} B$. Moreover, if B has fin then $A \leqslant_{\mathrm{F}} C \leqslant_{\mathrm{iB}} B$. Then Theorem 4.1 implies the needed conclusions. ▮

EXAMPLE 4.7. The automata A_9 and A_{10} of Fig. 4 illustrate the difference between $\leqslant_{\mathrm{T}}$ and $\leqslant_{\mathrm{iFB}}$, and also show that the assumption that B has *fin* in Theorem 4.6(2) is essential.

4.2. *Backward–Forward Simulations*

Having studied forward–backward simulations, we find it natural to define and study a dual notion of backward–forward simulation.

A *backward–forward simulation* from A to B is a total relation g over *states*(A) and $\mathbf{P}($*states*$(B))$ that satisfies:

1. If $s \in start(A)$ then, for all $S \in g[s]$, $S \cap start(B) \neq \varnothing$.
2. If $s' \xrightarrow{a}_A s$ and $S \in g[s]$, then there exists a set $S' \in g[s']$ such that for every $u' \in S'$ there exists a $u \in S$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

We write $A \leqslant_{\mathrm{BF}} B$ if there exists a backward–forward simulation from A to B, and $A \leqslant_{\mathrm{iBF}} B$ if there exists an image-finite backward–forward simulation from A to B.

As for forward–backward simulations, backward–forward simulations can be characterized as combinations of forward and backward simulations.

THEOREM 4.8.

1. $A \leqslant_{\mathrm{BF}} B \Leftrightarrow (\exists C\colon A \leqslant_{\mathrm{B}} C \leqslant_{\mathrm{F}} B)$.
2. $A \leqslant_{\mathrm{iBF}} B \Leftrightarrow (\exists C\colon A \leqslant_{\mathrm{iB}} C \leqslant_{\mathrm{F}} B)$.

Proof. "$\Rightarrow$" Let g be a backward–forward simulation from A to B, which is image-finite if $A \leqslant_{\mathrm{iBF}} B$. Define C to be the automaton given by

- *states*$(C) = range(g)$,
- *start*$(C) = range(g \lceil start(A))$,
- *acts*$(C) =$ *acts*(B), and
- for S', $S \in states(C)$ and $a \in acts(C)$, $S' \xrightarrow{a}_C S \Leftrightarrow \forall u' \in S'\colon \exists u \in S\colon u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Then g is a backward simulation from A to C (and image-finiteness carries over). Also, the relation $\{(S, u) \mid S \in states(C)$ and $u \in S\}$ is a forward simulation from C to B.

"$\Leftarrow$" Easy. ▮

PROPOSITION 4.9.

1. $A \leqslant_{\mathrm{F}} B \Rightarrow A \leqslant_{\mathrm{iBF}} B$.
2. $A \leqslant_{\mathrm{B}} B \Rightarrow A \leqslant_{\mathrm{BF}} B$.
3. $A \leqslant_{\mathrm{iB}} B \Rightarrow A \leqslant_{\mathrm{iBF}} B$.

Proof. Immediate from Theorem 4.8, using the fact that $\leqslant_{\mathrm{iB}}$ and $\leqslant_{\mathrm{F}}$ are reflexive. ▮

In order to show the properties of backward–forward simulations, it is useful to relate them to forward–backward simulations.

THEOREM 4.10.

1. $A \leqslant_{\mathrm{BF}} B \Leftrightarrow A \leqslant_{\mathrm{FB}} B$.
2. $A \leqslant_{\mathrm{iBF}} B \Rightarrow A \leqslant_{\mathrm{iFB}} B$.

Proof. For one direction of 1, suppose that $A \leqslant_{\mathrm{BF}} B$. Then by Theorem 4.8, there exists an automaton C with $A \leqslant_{\mathrm{B}} C \leqslant_{\mathrm{F}} B$. By Prop. 4.2, $A \leqslant_{\mathrm{FB}} C$ and $C \leqslant_{\mathrm{FB}} B$. Now $A \leqslant_{\mathrm{FB}} B$ follows by Prop. 4.4. The proof of 2 is similar.

For the other direction of 1, suppose that f is a forward–backward simulation from A to B. Given a state s of A, we define $g[s]$ to be exactly the set of subsets S of *states*(B) such that S intersects each set in $f[s]$ in at least one element. We claim that g is a backward–forward simulation from A to B.

1. g is total.

Proof. Suppose $s \in states(A)$. By assumption f is a forward–backward simulation, so all elements of $f[s]$ are nonempty. Hence the set $S = \bigcup f[s]$ intersects each element of $f[s]$ in at least one element. Thus, by definition S is in $g[s]$.

2. If $s \in start(A)$ then, for all $S \in g[s]$, $S \cap start(B) \neq \varnothing$

Proof. Suppose $s \in start(A)$ and $S \in g[s]$. By assumption f is a forward–backward simulation, so there exists a set S' in $f[s]$ such that $S' \subseteq start(B)$. By definition of g, S intersects S' in at least one element. Hence S intersects *start*(B) in at least one element.

3. If $s' \xrightarrow{a}_A s$ and $S \in g[s]$, then there exists a set $S' \in g[s']$ such that for every $u' \in S'$ there exists a $u \in S$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Proof. Suppose $s' \xrightarrow{a}_A s$ and $S \in g[s]$. Let $f[s'] = \{S'_i \mid i \in I\}$. By assumption f is a forward–backward simulation, so there exists, for each $i \in I$, a set $S_i \in f[s]$ such that for every $u \in S_i$ there exists $u' \in S'_i$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$. By definition of g, S intersects each of the sets S_i in at least one element. So choose, for each i, an element u_i in the intersection of S and S_i. Then, for each i, there exists $u'_i \in S'_i$ such that $u'_i \overset{\hat{a}}{\Longrightarrow}_B u_i$. Let $S' = \{u'_i \mid i \in I\}$. Then S' intersects each element of $f[s']$ in at least one element, so $S' \in f[s']$. By construction, for every $u' \in S'$ there exists a $u \in S$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Hence $A \leqslant_{\mathrm{BF}} B$. ▮

EXAMPLE 4.11. In general it is not the case that $A \leqslant_{\mathrm{iFB}} B$ implies $A \leqslant_{\mathrm{iBF}} B$. A counterexample is presented in Fig. 5. The diagram shows two automata A_{11} and A_{12}.

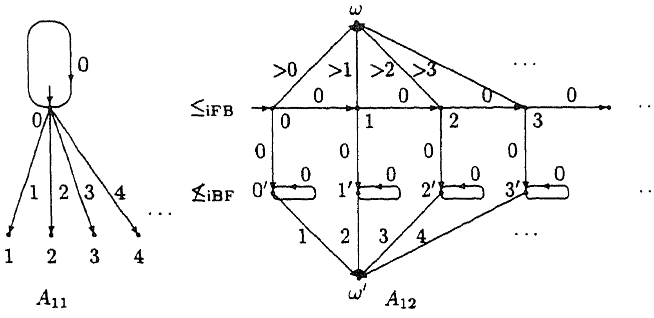

FIG. 5. The difference between $\leqslant_{\text{iFB}}$ and $\leqslant_{\text{iBF}}$.

In the diagram a label $>i$ next to an arc means that in fact there are infinitely many steps, labeled $i+1$, $i+2$, $i+3$, etc.

We claim that the relation g given by

$$g[0] = \{\{0\}, \{0', 1\}, \{0', 1', 2\}, ...\}$$

$$g[n] = \{\{\omega\}, \{\omega'\}\} \quad \text{for } n > 0$$

is an image-set-finite forward–backward simulation from A_{11} to A_{12}.

However, there is no image-finite backward–forward simulation from A_{11} to A_{12}. We see this as follows. Suppose g is an image-finite backward–forward simulation from A_{11} to A_{12}. In order to prove that this assumption leads to a contradiction, we first establish that $g[0]$ does not contain a finite subset X of N. First note that by the first condition in the definition of a backward–forward simulation, all sets in $g[0]$ are nonempty. The proof proceeds by induction on the maximal element of X. For the induction base, observe that $\{0\} \notin g[0]$, since 0 has an incoming 0-step in A_{11} but not in A_{12}. For the induction step, suppose that we have established that $g[0]$ contains no finite subset of N with a maximum less than n, and suppose $X \in g[0]$ with X a finite subset of N with maximum n. Using the fact that 0 has an incoming 0-step in A_{11}, the second condition in the definition of a backward–forward simulation gives that $g[0]$ contains an element of $g[0]$ which is a subset of N with a maximum less than n. This contradicts the induction hypothesis.

Pick some state $n > 0$ of A_{11} and a set $S' \in g[n]$. Since A_{11} has a step $0 \xrightarrow{n} n$, there exists a set $S \in g[0]$ such that every state in S has an outgoing n-step. Then S must be a subset of $\{0, ..., n-1, (n-1)'\}$. Since $g[0]$ does not contain the empty set or a finite subset of N, it follows that $(n-1)' \in S$. But since n was chosen arbitrarily (besides being positive) it follows that $g[0]$ has an infinite number of elements. This gives a contradiction with the assumption that g is image-finite.

Proposition 4.12. *$\leqslant_{\text{BF}}$ is a preorder. (However, $\leqslant_{\text{iBF}}$ is not a preorder.)*

Proof. The fact that $\leqslant_{\text{BF}}$ is a preorder, is trivially implied by Theorem 4.10 and Prop. 4.4.

The counterexample of Fig. 5 tells us that $\leqslant_{\text{iBF}}$ is not a preorder in general. If we take the two automata A_{11} and A_{12} from the example, then we can find an automaton C with $A_{11} \leqslant_{\text{F}} C \leqslant_{\text{iB}} A_{12}$, using Theorem 3.22. By Prop. 4.9, $A_{11} \leqslant_{\text{iBF}} C$ and $C \leqslant_{\text{iBF}} A_{12}$. Hence it cannot be that $\leqslant_{\text{iBF}}$ is transitive, because this would imply $A_{11} \leqslant_{\text{iBF}} A_{12}$. ▌

Soundness and completeness results for backward–forward simulations now follow from those for forward–backward simulations.

Theorem 4.13 (Soundness of Backward–Forward Simulations).

1. $A \leqslant_{\text{BF}} B \Rightarrow A \leqslant_{*\text{T}} B$.
2. $A \leqslant_{\text{iBF}} B \Rightarrow A \leqslant_{\text{T}} B$.

Proof. By Theorem 4.10 and Theorem 4.5. ▌

Theorem 4.14 (Completeness of Backward–Forward Simulations). $A \leqslant_{*\text{T}} B \Rightarrow A \leqslant_{\text{BF}} B$.

Proof. By Theorem 4.6 and Theorem 4.10. ▌

Example 4.11 falsifies the completeness result that one might expect here. That is, Theorem 4.14 does not have a second case saying that if B has *fin* and $A \leqslant_{*\text{T}} B$, then $A \leqslant_{\text{iBF}} B$.

5. AUXILIARY VARIABLE CONSTRUCTIONS

In this section, we present two new types of relations, history relations and prophecy relations, which correspond to the notions of history and prophecy variable of Abadi and Lamport [1]. We show that there is a close connection between history relations and forward simulations, and also between prophecy relations and backward simulations. Using these connections together with the earlier results of this paper, we can easily derive a completeness theorem for refinements similar to the one of Abadi and Lamport [1]. In fact, in the setting of this paper, the combination of history and prophecy relations and refinements gives exactly the same verification power as the combination of forward and backward simulations.

5.1. *History Relations*

A relation h over *states*(A) and *states*(B) is a *history relation* from A to B if h is a forward simulation from A to B and h^{-1} is a refinement from B to A. We write $A \leqslant_{\text{H}} B$ if there exists a history relation from A to B. Thus $A \leqslant_{\text{H}} B$ implies $A \leqslant_{\text{F}} B$ and $B \leqslant_{\text{R}} A$.

We give an example of a history relation, using the construction of the *unfolding* of an automaton; the unfolding of an automaton augments the automaton by remembering information about the past.

The *unfolding* of an automaton A, notation $unfold(A)$, is the automaton B defined by

- $states(B) = execs^*(A)$,
- $start(B) =$ the set of finite executions of A that consist of a single start state,
- $acts(B) = acts(A)$, and
- for $\alpha', \alpha \in states(B)$ and $a \in acts(B)$, $\alpha' \xrightarrow{a}_B \alpha \Leftrightarrow \alpha = \alpha' a\, last(\alpha)$.

PROPOSITION 5.1. *$unfold(A)$ is a forest and $A \leqslant_H unfold(A)$.*

Proof. Clearly, $unfold(A)$ is a forest. The function *last* which maps each finite execution of A to its last state is a refinement from $unfold(A)$ to A, and the relation $last^{-1}$ is a forward simulation from A to $unfold(A)$. ▮

EXAMPLE 5.2. For the automata of Fig. 2, $A_3 \nleqslant_H A_4$, $A_4 \leqslant_H A_3$, $A_5 \nleqslant A_6$ and $A_6 \nleqslant A_5$.

PROPOSITION 5.3. *$\leqslant_H$ is a preorder.*

Proof. Reflexivity is trivial. For transitivity, suppose h is a history relation from A to B and h' is a history relation from B to C. Then h is a forward simulation from A to B and h' is a forward simulation from B to C, so $h' \circ h$ is a forward simulation from A to C, by Proposition 3.9. Also, since h'^{-1} is a refinement from C to B and h^{-1} is a refinement from B to A, $(h' \circ h)^{-1} = h^{-1} \circ h'^{-1}$ is a refinement from C to A by Proposition 3.3. It now follows that $h' \circ h$ is a history relation from A to C. ▮

Although inspired by [39, 22, 25], the notion of a history relation is a new contribution of this paper. It provides a simple and abstract view of the *history variables* of Abadi and Lamport [1] (which in turn are abstractions of the *auxiliary variables* of Owicki and Gries [43]). Translated into the setting of this paper, history variables can be simply defined in terms of history relations, as follows.

An automaton B is obtained from an automaton A by *adding a history variable* if there exists a set V such that

- $states(B) \subseteq states(A) \times V$, and
- the relation $\{(s, (s, v)) \mid (s, v) \in states(B)\}$ is a history relation from A to B.

Whenever B is obtained from A by adding a history variable, then $A \leqslant_H B$ by definition. The following proposition states that the converse is also true if one is willing to consider automata up to isomorphism.

Two automata A and B are *isomorphic*, notation $A \cong B$, iff they have the same sets of actions and there exists an isomorphism between them, i.e., a bijective function φ from $states(A)$ to $states(B)$ satisfying

1. $s \in start(A)$ iff $\varphi(s) \in start(B)$.
2. $s' \xrightarrow{a}_A s$ iff $\varphi(s') \xrightarrow{a}_B \varphi(s)$.

PROPOSITION 5.4. *Suppose $A \leqslant_H B$. Then there exists an automaton C that is isomorphic to B and obtained from A by adding a history variable.*

Proof. Let h be a history relation from A to B. Define automaton C by

- $states(C) = h$,
- $(s, u) \in start(C) \Leftrightarrow u \in start(B)$,
- $acts(C) = acts(B)$, and
- for (s', u'), $(s, u) \in states(C)$ and $a \in acts(C)$, $(s', u') \xrightarrow{a}_C (s, u) \Leftrightarrow u' \xrightarrow{a}_B u$.

The function φ that maps a state (s, u) of C to the state u of B is an isomorphism between C and B: φ is bijective because h^{-1} is a function from states of B to states of A, and from the definition of C it is immediate that φ preserves initial states and steps. In order to see that C is obtained from A by adding a history variable, let $states(B)$ play the role of the set V required in the definition of a history variable. We check that $h' = \{(s, (s, v)) \mid (s, v) \in h\}$ is a history relation from A to C.

1. h' is a forward simulation from A to C.

Proof. Suppose $s \in start(A)$. Since h is a history relation from A to B, it is in particular a forward simulation from A to B. Thus there exists a state $v \in start(B) \cap h[s]$. By definition of C, $(s, v) \in start(C)$, and by definition of h', $(s, v) \in h'[s]$.

Next suppose $s' \xrightarrow{a}_A s$ and $(s', v') \in h'[s']$. Then $v' \in h[s']$ and so there exists a $v \in h[s]$ such that $v' \overset{\hat{a}}{\Longrightarrow}_B v$. This implies $(s, v) \in h'[s]$ and $(s', v') \overset{\hat{a}}{\Longrightarrow}_C (s, v)$.

2. h'^{-1} is a refinement from C to A.

Proof. Suppose $(s, v) \in start(C)$. Then $v \in start(B)$. Since h is a history relation from A to B, h^{-1} is a refinement from B to A. This implies

$$h'^{-1}(s, v) = s = h^{-1}(v) \in start(A).$$

Next suppose $(s', v') \xrightarrow{a}_C (s, v)$. Then by definition of C, $v' \xrightarrow{a}_B v$. Hence, since h^{-1} is a refinement from B to A,

$$h'^{-1}(s', v') = s' = h^{-1}(v') \overset{\hat{a}}{\Longrightarrow}_A h^{-1}(v) = s = h'^{-1}(s, v). \quad ▮$$

At first sight, Prop. 5.4 may look tautological, since history variables are defined in terms of history relations. However, note that the analogue of Prop. 5.4 does not hold in the setting of Klarlund and Schneider [25], who define their notion of a history variable in terms of forward simulations rather than history relations. Klarlund and Schneider [25] say that an automaton B is obtained from an automaton A by *adding history information* if there exists a set V such that

- $states(B) \subseteq states(A) \times V$, and
- the relation $\{(s, (s, v)) \mid (s, v) \in states(B)\}$ is a forward simulation from A to B.

It is easy to see that even though there is a forward simulation from automaton A_5 to automaton A_6 in Fig. 2, A_6 is not isomorphic to any automaton C obtained from A_5 by adding history information. This follows because each such C must have at least two start states.

Proposition 5.4 shows that in our setting history relations *do* capture the essence of history variables. For this reason and also because history relations have nicer theoretical properties, we will state all our results in this subsection in terms of relations, and will not mention the auxiliary variables any further.

THEOREM 5.5 (Soundness of History Relations). $A \leqslant_{\mathrm{H}} B \Rightarrow A \equiv_{\mathrm{T}} B$.

Proof. Immediate from the soundness of refinements and forward simulations. ∎

In fact, a history relation from A to B is just a functional (*weak*) *bisimulation* between A and B in the sense of Park [44] and Milner [41]. This implies that if there exists a history relation from A to B, both automata are *observation-equivalent*. Hence, history relations preserve the behavior of automata in a very strong sense.

We can now state and prove the completeness results of Sistla [46].

THEOREM 5.6 (Completeness of History Relations and Backward Simulations, [46]). *Suppose $A \leqslant_{*\mathrm{T}} B$. Then*

1. $\exists C$: $A \leqslant_{\mathrm{H}} C \leqslant_{\mathrm{B}} B$, *and*
2. *if B has fin then* $\exists C$: $A \leqslant_{\mathrm{H}} C \leqslant_{\mathrm{iB}} B$.

Proof. Take $C = unfold(A)$. By Prop. 5.1, C is a forest and $A \leqslant_{\mathrm{H}} C$. Since $A \leqslant_{*\mathrm{T}} B$, also $C \leqslant_{*\mathrm{T}} B$ by the soundness of history relations (Theorem 5.5). Next we can apply the partial completeness result for backward simulations (Theorem 3.18) to conclude (1) $C \leqslant_{\mathrm{B}} B$, and (2) if B has fin then $C \leqslant_{\mathrm{iB}} B$. ∎

Suppose R is a relation over $states(A)$ and $states(B)$ with $R \cap (start(A) \times start(B)) \neq \varnothing$. (Typically, R will be a forward or a backward simulation.) The *superposition* $sup(A, B, R)$ of B onto A via R is the automaton C defined by

- $states(C) = R$,
- $start(C) = R \cap (start(A) \times start(B))$,
- $acts(C) = acts(A) \cap acts(B)$, and
- for (s', u'), $(s, u) \in states(C)$ and $a \in acts(C)$,

$$(s', u') \xrightarrow{a}_{C} (s, u) \Leftrightarrow s' \overset{\hat{a}}{\Longrightarrow}_{A} s \wedge u' \overset{\hat{a}}{\Longrightarrow}_{B} u.$$

LEMMA 5.7. *Suppose f is a forward simulation from A to B. Let $C = sup(A, B, f)$ and let π_1 and π_2 be the projection functions that map states of C to their first and second components, respectively. Then π_1^{-1} is a history relation from A to C and π_2 is a refinement from C to B.*

The following theorem gives a precise and compact formulation of the folklore result that forward simulations are the same as history variables combined with refinements.

THEOREM 5.8. $A \leqslant_{\mathrm{F}} B \Leftrightarrow (\exists C\colon A \leqslant_{\mathrm{H}} C \leqslant_{\mathrm{R}} B)$.

Proof. For the implication "⇒", suppose $A \leqslant_{\mathrm{F}} B$. Let f be a forward simulation from A to B. Take $C = sup(A, B, f)$. The result follows by Lemma 5.7. For the implication "⇐", suppose that $A \leqslant_{\mathrm{H}} C \leqslant_{\mathrm{R}} B$. Then $A \leqslant_{\mathrm{F}} C$ by the definition of history relations, and $C \leqslant_{\mathrm{F}} B$ because any refinement is a forward simulation. Now $A \leqslant_{\mathrm{F}} B$ follows by the fact that $\leqslant_{\mathrm{F}}$ is a preorder. ∎

5.2. *Prophecy Relations*

Now we will present prophecy relations and show that they correspond to backward simulations, very similarly to the way in which history relations correspond to forward simulations.

A relation p over $states(A)$ and $states(B)$ is a *prophecy relation* from A to B if p is a backward simulation from A to B and p^{-1} is a refinement from B to A. We write $A \leqslant_{\mathrm{P}} B$ if there exists a prophecy relation from A to B, and $A \leqslant_{\mathrm{iP}} B$ if there is an image-finite prophecy relation from A to B. Thus $A \leqslant_{\mathrm{iP}} B$ implies $A \leqslant_{\mathrm{iB}} B$ and $A \leqslant_{\mathrm{P}} B$, and $A \leqslant_{\mathrm{P}} B$ implies $A \leqslant_{\mathrm{B}} B$ and $B \leqslant_{\mathrm{R}} A$. We give an example of a prophecy relation, using the construction of the *guess* of an automaton. This new construction is a kind of dual to the unfolding construction of the previous subsection in that the states contain information about the future rather than about the past.[6]

The *guess* of an automaton A, notation $guess(A)$, is the automaton B defined by

- $states(B) = frag^*(A)$,
- $start(B) = execs^*(A)$,

[6] Just as the unfolding operation gives rise to a forest, the guess construction leads to the dual notion of a *backward forest*, i.e., an automaton with the property that for each state there is a unique maximal execution that starts in it. Also, similar to the partial completeness result for backward simulations that requires one of the automata to be a forest, there is a partial completeness result for forward simulations that involves backward forests. Since the guess construction appears to be useful only in proving finite trace inclusion, we decided not to work out the forward/backward duality completely at this point.

- $acts(B) = acts(A)$, and
- for α', $\alpha \in states(B)$ and $a \in acts(B)$, $\alpha' \xrightarrow{a}_B \alpha \Leftrightarrow first(\alpha')\, a\alpha = \alpha'$.

PROPOSITION 5.9. *$A \leqslant_P guess(A)$.*

Proof. The function *first* which maps each execution fragment of A to its first state is a refinement from $guess(A)$ to A, and the relation $first^{-1}$ is a backward simulation from A to $guess(A)$. ▮

EXAMPLE 5.10. For the automata of Fig. 2 we have $A_3 \not\leqslant_P A_4$, $A_4 \not\leqslant_P A_3$, $A_5 \not\leqslant_P A_6$ and $A_6 \leqslant_{iP} A_5$. The difference between $\leqslant_P$ and $\leqslant_{iP}$ is illustrated by the automata of Fig. 3: $A_7 \leqslant_P A_8$ but $A_7 \not\leqslant_{iP} A_8$. The automata A_1 and A_2 of Fig. 1 cannot be used directly to show the difference between $\leqslant_P$ and $\leqslant_{iP}$ since neither $A_1 \leqslant_P A_2$ nor $A_2 \leqslant_P A_1$. However, we obtain a counterexample by unfolding the A_2 automaton: $A_1 \leqslant_P unfold(A_2)$ but $A_1 \not\leqslant_{iP} unfold(A_2)$.

PROPOSITION 5.11. *$\leqslant_P$ and $\leqslant_{iP}$ are preorders.*

The following proposition sheds some more light on the relationship between $\leqslant_P$ and $\leqslant_{iP}$.

PROPOSITION 5.12. *Suppose all states of A are reachable, B has fin, and $A \leqslant_P B$. Then $A \leqslant_{iP} B$.*

Proof. Let p be a prophecy relation from A to B. Then p is a backward simulation. Now the proof of Proposition 3.20 implies that p is image-finite. Thus $A \leqslant_{iP} B$. ▮

We will now show that prophecy relations capture the essence of prophecy variables, just as history relations capture the essence of history variables.

An automaton B is obtained from an automaton A by *adding a prophecy variable* if there exists a set V such that

- $states(B) \subseteq states(A) \times V$, and
- the relation $\{(s, (s, v)) \mid (s, v) \in states(B)\}$ is a prophecy relation from A to B.

A prophecy variable is *bounded* if the underlying prophecy relation is image-finite.

PROPOSITION 5.13. *Suppose $A \leqslant_P B$. Then there exists an automaton C that is isomorphic to B and obtained from A by adding a prophecy variable, which is bounded if $A \leqslant_{iP} B$.*

Again, we will state all further results in this subsection in terms of relations, and not mention the auxiliary variables any further.

THEOREM 5.14 (Soundness of Prophecy Relations).

1. $A \leqslant_P B \Rightarrow A \equiv_{*T} B$.
2. $A \leqslant_{iP} B \Rightarrow A \equiv_T B$.

Proof. Immediate from the soundness of refinements and backward simulations. ▮

LEMMA 5.15. *Suppose b is a backward simulation from A to B. Let $C = sup(A, B, b)$ and let π_1 and π_2 be the projection functions that map states of C to their first and second components, respectively. Then π_1^{-1} is a prophecy relation from A to C and π_2 is a refinement from C to B. If b is image-finite then so is π_1^{-1}.*

THEOREM 5.16.

1. $A \leqslant_B B \Leftrightarrow (\exists C\colon A \leqslant_P C \leqslant_R B)$.
2. $A \leqslant_{iB} B \Leftrightarrow (\exists C\colon A \leqslant_{iP} C \leqslant_R B)$.

Proof. The proof of 1 is analogous to that of Theorem 5.8, using Lemma 5.15. Statement 2 can be proved similarly. ▮

The following result is dual to Sistla's completeness result.

THEOREM 5.17 (Completeness of Prophecy Relations and Forward Simulations). *$A \leqslant_{*T} B \Rightarrow \exists C\colon A \leqslant_P C \leqslant_F B$.*

Proof.

$$
\begin{aligned}
A \leqslant_{*T} B &\Rightarrow A \leqslant_{BF} B \\
&\qquad \text{(by Theorem 4.14)} \\
&\Rightarrow \exists D\colon A \leqslant_B D \leqslant_F B \\
&\qquad \text{(by Theorem 4.8)} \\
&\Rightarrow \exists C, D\colon A \leqslant_P C \leqslant_R D \leqslant_F B \\
&\qquad \text{(by Theorem 5.16)} \\
&\Rightarrow \exists C\colon A \leqslant_P C \leqslant_F B \\
&\qquad \text{(by Propositions 3.7 and 3.9).}
\end{aligned}
$$
▮

5.3. *Completeness of History and Prophecy Relations*

We finish this section with versions of the completeness results of Abadi and Lamport [1].

THEOREM 5.18 (Completeness of History Relations, Prophecy Relations, and Refinements [1]). *Suppose $A \leqslant_{*T} B$. Then*

1. *$\exists C, D\colon A \leqslant_H C \leqslant_P D \leqslant_R B$, and*
2. *if B has fin then $\exists C, D\colon A \leqslant_H C \leqslant_{iP} D \leqslant_R B$.*

Proof. By Sistla's result (Theorem 5.6), there exists an automaton C with $A \leqslant_H C \leqslant_B B$. Next, Theorem 5.16 yields the required automaton D with $C \leqslant_P D \leqslant_R B$, which proves 1. Now statement 2 is routine. ▮

Similarly, we obtain the dual result:

THEOREM 5.19. *$A \leqslant_{*T} B \Rightarrow \exists C, D\colon A \leqslant_P C \leqslant_H D \leqslant_R B$.*

6. INCLUDING INVARIANTS

For the sake of simplicity, our entire development so far has been carried out without any mention of invariants;

in fact, all considerations involving reachability of the various states have been ignored. However, in actual verification examples using simulations, it is almost always the case that a preliminary collection of invariants is proved, then used in proving the step correspondence. In this section, we show how to integrate invariants into simulation proofs.

We define an *invariant* of an automaton A to be any superset of the set of reachable states of A, i.e., a property that is true of all the reachable states of A.[7] One way to prove that a property is an invariant is by induction on the length of a finite execution that leads to the state in question. More usually, a batch of invariants is proved together, by induction. In fact, invariants are most often proved in several batches, where each batch is proved by induction, assuming that those in the previous batches are true.

We now define versions of all our simulations that use invariants. We call these simulations "weak," although that is is bit of a misnomer in the case of some of the simulations.[8] Let A and B be automata with invariants I_A and I_B, respectively.

A *weak refinement* from A to B, with respect to I_A and I_B, is a function r from $states(A)$ to $states(B)$ that satisfies the following two conditions:

1. If $s \in start(A)$ then $r(s) \in start(B)$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$, and $r(s') \in I_B$, then $r(s') \overset{\hat{a}}{\Longrightarrow}_B r(s)$.

A *weak forward simulation* from A to B, with respect to I_A and I_B, is a relation f over $states(A)$ and $states(B)$ that satisfies the following two conditions:

1. If $s \in start(A)$ then $f[s] \cap start(B) \neq \emptyset$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$, and $u' \in f[s'] \cap I_B$, then there exists a state $u \in f[s]$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.

Thus, weak refinements and weak forward simulations are weaker than ordinary refinements and forward simulations in that they allow use of invariants for all the hypothesized states.

A *weak backward simulation* from A to B, with respect to I_A and I_B, is a relation b over $states(A)$ and $states(B)$ that satisfies:

1. If $s \in start(A)$ then $b[s] \cap I_B \subseteq start(B)$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$, and $u \in b[s] \cap I_B$, then there exists a state $u' \in b[s'] \cap I_B$ such that $u' \overset{\hat{a}}{\Longrightarrow}_B u$.
3. If $s \in I_A$ then $b[s] \cap I_B \neq \emptyset$.

Thus, weak backward simulations allow use of invariants in all the hypothesized states. However, they also require additional proof obligations: in the second and third properties, it is necessary to show that the state produced satisfies I_B. So, strictly speaking, they are not weaker than ordinary backward simulations.

A *weak forward–backward simulation* from A to B, with respect to I_A and I_B, is a relation g over $states(A)$ and $\mathbf{P}(states(B))$ that satisfies:

1. If $s \in start(A)$ then there exists $S \in g[s]$ such that $S \cap I_B \subseteq start(B)$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$ and $S' \in g[s']$, then there exists a set $S \in g[s]$ •such that for every $u \in S \cap I_B$ there exists $u' \in S' \cap I_B$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.
3. If $s \in I_A$ and $S \in g[s]$ then $S \cap I_B \neq \emptyset$.

A *weak backward–forward simulation* from A to B, with respect to I_A and I_B, is a relation g over $states(A)$ and $\mathbf{P}(states(B))$ that satisfies:

1. If $s \in start(A)$ then, for all $S \in g[s]$, $S \cap start(B) \neq \emptyset$.
2. If $s' \xrightarrow{a}_A s$, $s', s \in I_A$ and $S \in g[s]$, then there exists a set $S' \in g[s']$ such that for every $u' \in S' \cap I_B$ there exists a $u \in S \cap I_B$ with $u' \overset{\hat{a}}{\Longrightarrow}_B u$.
3. If $s \in I_A$ then $g[s] \neq \emptyset$.

A relation h over $states(A)$ and $states(B)$ is a *weak history relation* from A to B, with respect to I_A and I_B, provided that h is a weak forward simulation from A to B, with respect to I_A and I_B, and h^{-1} is a weak refinement from B to A, with respect to I_B and I_A.

A relation p over $states(A)$ and $states(B)$ is a *weak prophecy relation* from A to B, with respect to I_A and I_B, provided that p is a weak backward simulation from A to B, with respect to I_A and I_B, and p^{-1} is a weak refinement from B to A, with respect to I_B and I_A.

We write $A \leqslant_{\mathrm{wR}} B$, $A \leqslant_{\mathrm{wF}} B$, $A \leqslant_{\mathrm{wB}} B$, $A \leqslant_{\mathrm{wiB}} B$, $A \leqslant_{\mathrm{wFB}} B$, $A \leqslant_{\mathrm{wiFB}} B$, $A \leqslant_{\mathrm{wBF}} B$, $A \leqslant_{\mathrm{wiBF}} B$, $A \leqslant_{\mathrm{wH}} B$, $A \leqslant_{\mathrm{wP}} B$ and $A \leqslant_{\mathrm{wiP}} B$ to denote the existence of a weak refinement, weak forward simulation, weak backward simulation, weak image-finite backward simulation, etc., from A to B, with respect to some invariants I_A and I_B.

Proposition 6.1. *The relations* $\leqslant_{\mathrm{wR}}$, $\leqslant_{\mathrm{wF}}$, $\leqslant_{\mathrm{wB}}$, $\leqslant_{\mathrm{wiB}}$, $\leqslant_{\mathrm{wFB}}$, $\leqslant_{\mathrm{wiFB}}$, $\leqslant_{\mathrm{wBF}}$, $\leqslant_{\mathrm{wH}}$, $\leqslant_{\mathrm{wP}}$, *and* $\leqslant_{\mathrm{wiP}}$ *are all preorders. (However,* $\leqslant_{\mathrm{wiBF}}$ *is not a preorder.)*

Theorem 6.2 (Soundness of Weak Simulations),

1. *If* $A \leqslant_{\mathrm{wR}} B$, $A \leqslant_{\mathrm{wF}} B$, $A \leqslant_{\mathrm{wiB}} B$, $A \leqslant_{\mathrm{wiFB}} B$, $A \leqslant_{\mathrm{wiBF}} B$, $A \leqslant_{\mathrm{wH}} B$, *or* $A \leqslant_{\mathrm{wiP}} B$, *then* $A \leqslant_{\mathrm{T}} B$.
2. *If* $A \leqslant_{\mathrm{wB}} B$, $A \leqslant_{\mathrm{wFB}} B$, $A \leqslant_{\mathrm{wBF}} B$, *or* $A \leqslant_{\mathrm{wP}} B$, *then* $A \leqslant_{*\mathrm{T}} B$.

[7] Sometimes the term "invariant" is used with a slightly different meaning, to denote a property that holds initially and is preserved by all transitions.

[8] This usage of the term "weak" has nothing to do with Milner's usage [41]; he uses it to indicate whether or not internal steps are abstracted away.

Proposition 6.1 and Theorem 6.2 can be proved analogously to the way we proved the corresponding results for the nonweak case. Alternatively, it is possible to derive these results as consequences of the corresponding results for the nonweak case. We do this in a technical report version of this paper, [37].

7. CONCLUSIONS AND RELATED WORK

In this paper, we have given a unified, comprehensive presentation of simulation proof methods for untimed automata, including refinements, forward and backward simulations and combinations thereof, and history and prophecy relations. We have given relationships between all of these kinds of simulations, plus soundness and completeness results.

We summarize the basic implications between the various simulation techniques of this paper in a diagram. Suppose $M, N \in \{\mathrm{T}, *\mathrm{T}, \mathrm{R}, \mathrm{F}, (\mathrm{i})\mathrm{B}, (\mathrm{i})\mathrm{FB}, (\mathrm{i})\mathrm{BF}, \mathrm{H}, (\mathrm{i})\mathrm{P}\}$, where the (i) indicates that i is optional. Then $A \leqslant_{\mathrm{M}} B \Rightarrow A \leqslant_{\mathrm{N}} B$ for all automata A and B if and only if there is a path from $\leqslant_{\mathrm{M}}$ to $\leqslant_{\mathrm{N}}$ in Fig. 6 consisting of thin lines only. If B has *fin*, then $A \leqslant_{\mathrm{M}} B \Rightarrow A \leqslant_{\mathrm{N}} B$ for all automata A and B if and only if there is a path from $\leqslant_{\mathrm{M}}$ to $\leqslant_{\mathrm{N}}$ consisting of thin lines and thick lines. In the technical report version of this paper, [37], this classification is extended to include the various weak simulations as well.

The classification of Fig. 6 has been established for a specific automaton model (labeled transition systems with multiple start states but without final states) and a specific behavioral preorder (inclusion of finite and infinite traces with hiding of internal actions). We have chosen this model because of its simplicity and because it is used both in the theory of I/O automata [35, 20] and in the theory of process algebras [4, 16, 41], two important approaches toward specification and verification of concurrent systems. Simulations techniques play an important role in many other models of computation, and variants of most of the simulations that we discuss here have been proposed in the literature for other models and other notions of behavior. If one attempts to classify all the simulations that have been defined for any given approach, then typically one will get a picture very similar to our Fig. 6. Still, it is in most cases difficult, if not impossible, to formally derive results about simulations in one approach from the corresponding results in another approach: although the general picture is the same, the details are frequently different, and one should always be careful with claims that simulation proof methods carry over from one setting to another. We give some examples.

$$\begin{tikzcd}
 & & \leq_{\mathrm{iP}} \arrow[r] \arrow[d] & \leq_{\mathrm{P}} \arrow[d] \\
 & \leq_{\mathrm{R}} \arrow[r] \arrow[d] & \leq_{\mathrm{iB}} \arrow[r] \arrow[d] & \leq_{\mathrm{B}} \arrow[d] \\
\leq_{\mathrm{H}} \arrow[r] & \leq_{\mathrm{F}} \arrow[r] & \leq_{\mathrm{iBF}} \arrow[r] \arrow[d] & \leq_{\mathrm{BF}} \arrow[d, shift right] \\
 & & \leq_{\mathrm{iFB}} \arrow[r] \arrow[d, shift right] & \leq_{\mathrm{FB}} \arrow[u, shift right] \arrow[d, shift right] \\
 & & \leq_{\mathrm{T}} \arrow[u, shift right, line width=1pt] \arrow[r, shift right] & \leq_{*\mathrm{T}} \arrow[l, shift right, line width=1pt] \arrow[u, shift right]
\end{tikzcd}$$

FIG. 6. Classification of basic relations between automata.

1. In this paper we follow an action-based approach, in which the behavior of a system is a sequence of (visible) *actions*. Another popular approach, followed for instance in [1], is based on states: the behavior of a system is a sequence of *states* (up to stuttering). These different approaches are, in some sense, equivalent. In [5], for instance, translations are presented between an action based model of labeled transition systems (LTSs) and a state base model of Kripke structures (KSs). These translations show that the concept of internal actions in action-based approaches is the same, in some sense, as the concept of stuttering in state-based approaches. However, if one takes our history variables and maps them to the state based world via the translation of [5], one gets something which is slightly different from the history variables of [1], due to a subtle difference in the treatment of internal actions/stuttering. By slightly restricting our history and prophecy relations one can obtain history and prophecy variables that do correspond exactly to those of [1]. However, doing this either destroys the classification of Fig. 6, or forces us to change the definitions of all the other simulations as well, with the result that the correspondence with previous work on simulations in action-based approaches (for instance, [19, 21, 22, 34, 35, 39]) gets lost.

2. In classical automata theory, there is a complete duality between past and future since besides start states there are also final states, and traces correspond to finite executions from a start state to a final state. In our automata there are no final states, and traces correspond to possibly infinite executions from start states. As a result forward and backward simulations are not completely dual in our setting, unlike in classical automata theory.

3. There are a few results that depend on whether invariants are included in the definitions of the simulations. For example, the implication

$$A \leqslant_{\mathrm{B}} B \wedge B \quad \text{has } \mathit{fin} \Rightarrow A \leqslant_{\mathrm{iB}} B$$

is not valid in our setting (Example 3.21), but does hold in the context of [39] because there reachability conditions are included in the definition of backward simulations (cf. Proposition 3.20).

4. Simulation techniques have been used not only to prove trace inclusion, but also to prove several other preorders from Van Glabbeek's [11, 12] linear time—branching time spectrum. In [23, 13] for instance, proof methods based on forward and backward simulations are presented and proved to be sound and complete for the failure preorder of CSP [16]. In the definitions of these simulations additional clauses are present which for instance require that related states have the same initial actions.

5. All the automata studied in this paper have been untimed. In Part II [38], the simulation definitions and the results of this paper are extended to timed systems. In fact, many of the results for the timed case are obtained as consequences of the analogous results for the untimed case. However, there are several results that do carry over, but cannot be proved from the untimed results. Furthermore, the implication

$$A \leqslant_{\mathrm{F}} B \wedge A \quad \text{is a forest} \Rightarrow A \leqslant_{\mathrm{R}} B$$

of Proposition 3.12 does not carry over to the timed setting.

6. As far as the classification of simulations is concerned, our work is closely related to and extends that of Jonsson [22]. Jonsson, however, has a more powerful notion of backward simulation that can also handle automata with infinite invisible nondeterminism. We preferred not to use this notion since it fails to reduce reasoning about entire executions to reasoning about individual states and transitions.

7. This paper is related to the work of [17, 18, 3, 7, 48] on data refinement. In [17], an *operation* is a binary relation over some universal set Σ. A *data type* is a triple (AI, AO, AF), where AI and AF are the initialization and finalization operation, respectively, and $AO = \{AO_j \mid j \in J\}$ is an indexed set of operations. An automaton A can be encoded as a data type by defining

$$AI \triangleq \Sigma \times start(A)$$
$$AO_a \triangleq \{(s', s) \mid s' \xrightarrow{a}_A s\}, \quad \text{for all } a \in acts(A)$$
$$AF \triangleq states(A) \times \Sigma.$$

Here $acts(A)$ plays the role of the index set J. In [17], a *downward simulation* from $(AI, \{AO_j \mid j \in J\}, AF)$ to $(BI, \{BO_j \mid j \in J\}, BF)$ is defined to be any relation R for which the following inequations hold:

$$BI \subseteq R \circ AI$$
$$BO_j \circ R \subseteq R \circ AO_j, \quad \text{for all } j \in J$$
$$BF \circ R \subseteq AF.$$

It is easy to verify that in a setting without τ-steps, a relation $f \subseteq states(A) \times states(B)$ is a forward simulation from A to B iff f^{-1} is a downward simulation from the data type encoding B to the data type encoding A. A similar correspondence can be established between our backward simulations and the upward simulations of [17]. Just as forward and backward simulations provide a sound and complete proof method for trace inclusion between automata, downward and upward simulations offer a sound and complete proof method for *refinement* between data types. Surprisingly however, the definition of refinement between data types is completely different from the definition of trace inclusion between automata: informally, one data type is refined by another if any program that uses the former would function at least as well using the latter. Even more surprising (at least for us) is the fact that the requirements of totality and finite invisible nondeterminism that we used to prove soundness of backward simulations, also play a role in the soundness result of upward simulations, in case iteration and recursion can be used in the formation of programs [17]. Clearly, an important topic for future research is the connection between automata based simulation techniques and methods for data refinement. A specific question here concerns the relationship between forward–backward simulations and the single complete rule for data refinement of [7].

The present paper provides complete proof methods for trace inclusion between automata with finite invisible nondeterminism. Such automata express the class of safety properties [2]. For simplicity, we have not considered liveness properties here. Simulation techniques that deal with liveness are for instance described in [21, 22, 1, 8].The results of [1, 8] are more general than ours because safety and liveness issues are separated through the use of automata that are equipped with additional liveness properties.

History variables were first defined at the syntactic level for specific (parallel) programming languages. Owicki and Gries [43], for instance, define an *auxiliary variable set* for a statement S to be a set of variables AV that appears in S only in assignments $x := E$, where x is in AV. One of the contributions of Abadi and Lamport [1] is a language independent, semantic definition of this important concept. In this paper we have simplified their definition and the proof of their completeness theorem by observing that history variables are equivalent to history relations, and the dual prophecy variables are equivalent to prohecy relations. Several authors have observed that forward and backward simulations are closely related to history and prophecy variables, respectively, [39, 22, 25].[9] Still we believe that,

[9] Note, however, that [39] contains some minor flaws (Propositions 7.1, 7.6, and 8.1 are incorrect), and that the auxiliary variables of [25] have the peculiar property that adding them may change the visible behavior of an automaton.

through Theorems 5.8 and 5.16, our paper is the first to establish an exact correspondence in a general setting of transition systems.

In this paper we have only discussed simulation techniques at the semantic level of automata. We have not paid any attention to the syntax that is used to define these automata. Since some of our methods require the introduction of intermediate automata, this means that if one wants to use these methods for any given language, one has to check whether this language is suffiently expressive to describe the intermediate automata. Also, one has to check whether the language used for specifying relations is sufficiently expressive to define the various simulation relations that are required in a correctness proof. We leave it is a topic for future research to find syntactic formulations of our results.

Refinements, history variables and forward simulations have been used extensively and successfully for verifying concurrent algorithms. Backward simulations and prophecy variables have also been shown to be of practical value in a few cases. Additional work remains to determine the practical utility of backward simulations, prophecy variables and relations, and the hybrid methods of this paper. This will involve applying these techniques to a wide range of examples.

APPENDIX A. MATHEMATICAL PRELIMINARIES

This appendix contains some basic mathematical notation, plus a compactness lemma.

A.1. *Sequences*

Let K be any set. The sets of finite and infinite sequences of elements of K are denoted by K^* and K^ω, respectively. The symbol λ denotes the empty sequence and the sequence containing one element $k \in K$ is denoted by k. Concatenation of a finite sequence with a finite or infinite sequence is denoted by juxtaposition. We say that a sequence σ is a *prefix* of a sequence ρ, denoted by $\sigma \leqslant \rho$, if either $\sigma = \rho$, or σ is finite and $\rho = \sigma\sigma'$ for some sequence σ'. A set L of sequences is *prefix closed* if, whenever some sequence is in L, all its prefixes are also.

If σ is a nonempty sequence then *first*(σ) denotes the first element of σ, and *tail*(σ) denotes the sequence obtained from σ by removing *first*(σ). Moreover, if σ is finite, then *last*(σ) denotes the last element of σ.

If σ is a sequence over K and $K' \subseteq K$, then $\sigma \lceil K'$ denotes the projection of σ on K', i.e., the subsequence of σ consisting of the elements of K'. If L is a set of sequences, $L \lceil K'$ is defined as $\{\sigma \lceil K' \mid \sigma \in L\}$.

A.2. *Sets, Relations, and Functions*

A *relation* over sets X and Y is defined to be any subset of $X \times Y$. If R is a relation over X and Y, then we define the *domain* of R to be *domain*$(R) \triangleq \{x \in X \mid (x, y) \in R$ for some $y \in Y\}$, and the *range* of R to be *range*$(R) \triangleq \{y \in Y \mid (x, y) \in R$ for some $x \in X\}$. A relation R over X and Y is *total* over X if *domain*$(R) = X$. If X is any set, we let *id*(X) denote the identity relation over X and X, i.e., $\{(x, x) \mid x \in X\}$.

Suppose that R and R' are relations over X and Y and over Y and Z, respectively. Then the *composition* of R and R', denoted by $R' \circ R$ (pronounced R' *after* R) is the relation over X and Z defined by

$$(x, z) \in R' \circ R \Leftrightarrow \exists y \in Y: ((x, y) \in R \wedge (y, z) \in R').$$

For all relations R, R', and R'', $R \circ (R' \circ R'') = (R \circ R') \circ R''$. Also, for any relation R over X and Y, $id(Y) \circ R = R \circ id(X) = R$.

If R is a relation over X and Y, then the *inverse* of R, written R^{-1}, is defined to be the relation over Y and X consisting of those pairs (y, x) such that $(x, y) \in R$. For any pair of relations R and R', $(R' \circ R)^{-1} = R^{-1} \circ (R')^{-1}$.

If R is a relation over X and Y, and Z is a set, then $R \lceil Z$ is the relation over $X \cap Z$ and Y given by $R \lceil Z \triangleq R \cap (Z \times Y)$. If R is a relation over X and Y and $x \in X$, we define $R[x] = \{y \in Y \mid (x, y) \in R\}$. We say that a relation R over X and Y is a *function from X to Y* if $|R[x]| = 1$ for all $x \in X$ in this case, we write $R(x)$ to denote the unique element o $R[x]$. We write $X \to Y$ for the set of functions from X to Y A function c from X to Y is a *choice function* for a relatior R over X and Y provided that $c \subseteq R$ (i.e., $c(x) \in R[x]$ for all $x \in X$).

If X is a set, $\mathbf{P}(X)$ denotes the powerset of X, i.e., the set of subsets of X, and $\mathbf{N}(X)$ the set of nonempty subsets of X, i.e., the set $\mathbf{P}(X) - \{\varnothing\}$. We say that a relation R over X and Y is *image-finite* if $R[x]$ is finite for all x in X. If R is a relation over X and $\mathbf{P}(Y)$, then we say that R is *image-set-finite* if every set in the range of R is finite.

A.3. *A Basic Graph Lemma*

We require the following lemma, a generalization of König's Lemma [26]. If G is a digraph, then a *root* of G is defined to be a node with no incoming edges.

LEMMA A.1. *Let G be an infinite digraph that satisfies the following properties.*

1. *G has finitely many roots.*
2. *Each node of G has finite outdegree.*
3. *Each node of G is reachable from some root of G.*

Then there is an infinite path in G starting from some root.

Proof. The usual proof for König's Lemma extends to this case. ▌

APPENDIX B. GLOSSARY OF CONVENTIONS

a	Actions
b	Backward simulations
c	Choice functions
f	Forward simulations
g	Forward–backward and backward–forward simulations
h	History relations
i	Indices
n	Natural numbers
p	Prophecy relations
r	Refinements
s, u	States
A, B, C, D	Automata
G	Digraphs
I	Invariants
K	Sets of symbols
L	Sets of sequences
P, Q	Trace properties
R	Relations
S, U	Sets of states
X, Y, Z	Sets
α	Execution fragments
β	Sequences of external actions (traces)
γ	Sequences of actions
λ	The empty sequence
π	Projections
σ, ρ	Sequences
τ	The internal action

Received March 2, 1993; final manuscript received November 18, 1994

REFERENCES

1. Abadi, M., and Lamport, L. (1991), The existence of refinement mappings, *Theoret. Comput. Sci.* **82**(2), 253–284.
2. Alpern, B., and Schneider, F. B. (1985), Defining liveness, *Inform. Process. Lett.* **21**, 181–185.
3. Back, R. J. R., and von Wright, J. (1990), Refinement calculus. I. Sequential nondeterministic programs, *in* "Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 42–66, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.
4. Baeten, J. C. M., and Weijland, W. P. (1990), "Process Algebra," Cambridge Tracts in Theoretical Computer Science, Vol. 18, Cambridge Univ. Press, London, New York.
5. De Nicola, R., and Vaandrager, F. W. (1990), Three logics for branching bisimulation (extended abstract), *in* "Proceedings, 5th Annual Symposium on Logic in Computer Science, Philadelphia, USA," pp. 118–129, IEEE Comput. Soc. Press, Los Alamitos, CA; Full version to appear in *J. Assoc. Comput. Mach.*; available as Rapporto di Ricerca SI-92/07, Dipartimento di Scienze dell'Informazione, Università degli Studi di Roma "La Sapienza," November 1992.
6. Dijkstra, E. W. (1976), "A Discipline of Programming," Prentice–Hall International, Englewood Cliffs, NJ.
7. Gardiner, P. H. B., and Morgan, C. C. (1993), A single complete rule for data refinement, *J. Formal Aspects Comput. Sci.* **5**, 367–382.
8. Gawlick, R., Segala, R., Søgaard-Andersen, J. F., and Lynch, N. (1994), Liveness in timed and untimed systems, *in* "Proceedings, 21th ICALP, Jerusalem" (S. Abiteboul and E. Shamir, Eds.), Lecture Notes in Computer Science, Vol. 820, Springer-Verlag, Berlin/New York; a full version appears as MIT Technical Report MIT/LCS/TR-587.
9. Gerth, R. (1990), Foundations of compositional program refinement [first version], *in* "Proceedings, REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 777–808, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.
10. Ginzburg, A. (1968), "Algebraic Theory of Automata," Academic Press, New York/London.
11. van Glabbeek, R. J. (1990), The linear time–branching time spectrum, *in* "Proceedings CONCUR 90, Amsterdam" (J. C. M. Baeten and J. W. Klop, Eds.), pp. 278–297, Lecture Notes in Computer Science, Vol. 458, Springer-Verlag, Berlin/New York.
12. van Glabbeek, R. J. (1993), The linear time–branching time spectrum, II (the semantics of sequential systems with silent moves), *in* "Proceedings CONCUR 93, Hildesheim, Germany" (E. Best, Ed.), pp. 66–81, Lecture Notes in Computer Science, Vol. 715, Springer-Verlag, Berlin/New York.
13. He, J. (1989), Process simulation and refinement, *J. Formal Aspects Comput. Sci.* **1**, 229–241.
14. Helmink, L., Sellink, M. P. A., and Vaandrager, F. W. (1994), Proof-checking a data link protocol, *in* "Proceedings, International Workshop TYPES'93, Nijmegen, The Netherlands, May 1993" (H. Barendregt and T. Nipkow, Eds.), pp. 127–165, Lecture Notes in Computer Science, Vol. 806, Springer-Verlag, Berlin/New York; full version available as Report CS-R9420, CWI, Amsterdam, March 1994.
15. Hoare, C. A. R. (1972), Proof of correctness of data representations, *Acta Informat.* **1**, 271–281.
16. Hoare, C. A. R. (1985), "Communicating Sequential Processes," Prentice–Hall International, Englewood Cliffs, NJ.
17. Hoare, C. A. R., He, J., and Sanders, J. W. (1987), Prespecification in data refinement, *Inform. Process. Lett.* **25**, 71–76.
18. Jones, C. B. (1986), "Systematic Software Development Using VDM," Prentice–Hall International, Englewood Cliffs, NJ.
19. Jonsson, B. (1987), "Compositional Verification of Distributed Systems," Ph.D. thesis, Department of Computer Systems, Uppsala University, 1987, [DoCS 87/09].
20. Jonsson, B. Modular verification of asynchronous networks, *in* "Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, Aug. 1987," pp. 152–166.
21. Jonsson, B. (1990), On decomposing and refining specifications of distributed systems, *in* "Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 361–387, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.
22. Jonsson, B. (1991), Simulations between specifications of distributed systems, *in* "Proceedings CONCUR 91, Amsterdam" (J. C. M. Baeten and J. F. Groote, Eds.), pp. 346–360, Lecture Notes in Computer Science, Vol. 527, Springer-Verlag, Berlin/New York.
23. Josephs, M. B. (1988), A state-based approach to communicating processes, *Distrib. Comput.* **3**, 9–18.
24. Klarlund N., and Schneider, F. B. (1989), "Verifying Safety Properties Using Infinite-State Automata," Technical Report 89-1039, Department of Computer Science, Cornell University.
25. Klarlund N., and Schneider, F. B. (1993), Proving nondeterministically specified safety properties using progress measures, *Inform. and Comput.* **107**, 151–170.

26. Knuth, D. E. (1973), "Fundamental Algorithms," The Art of Computer Programming, Vol. 1, 2nd ed., Addison-Wesley, Reading, MA.

27. Lam, S. S., and Shankar, A. U. (1984), Protocol verification via projections, *IEEE Trans. Software Engrg.* **10**(4), 325–342.

28. Lamport L. (1983), Specifying concurrent program modules, *ACM Trans. Programming Languages Systems* **5**(2), 190–222.

29. Lampson, B. W., Lynch, N. A., and Søgaard-Andersen, J. F. Correctness of at-most-once message delivery protocols, *in* "FORTE'93—Sixth International Conference on Formal Description Techniques, Boston, MA, October 1993," pp. 387–402.

30. Liskov, B. L., and Guttag, J. V. (1986), "Abstraction and Specification in Program Development," MIT Press, Cambridge, MA/McGraw-Hill, New York.

31. Loewenstein, P., and Dill, D. L. (1991), Verification of a multiprocessor cache protocol using simulation relations and higher-order logic [summary], *in* "Proceedings of the 2nd International Conference on Computer-Aided Verification, New Brunswick, NJ" (E. M. Clarke and R. P. Kurshan, Eds.), pp. 302–311, Lecture Notes in Computer Science, Vol. 531, Springer-Verlag, Berlin/New York.

32. Lucas, P. (1968), "Two Constructive Realizations of the Block Concept and Their Equivalence," Technical Report 25.085, IBM Laboratory, Vienna.

33. Lynch, N. A. (1993), "Concurrency Control for Resilient Nested Transactions," MIT Technical Report MIT/LCS/TR-285, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

34. Lynch, N. A. (1990), Multivalued possibilities mappings, *in* "Proceedings, REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 519–530, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.

35. Lynch, N. A., and Tuttle, M. R. Hierarchical correctness proofs for distributed algorithms, *in* "Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, August 1987," pp. 137–151; a full version is available as MIT Technical Report MIT/LCS/TR-387.

36. Lynch, N. A., and Vaandrager, F. W. (1992), Forward and backward simulations for timing-based systems, *in* "Proceedings REX Workshop on Real-Time: Theory in Practice, Mook, The Netherlands, June 1991" (J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, Eds.), pp. 397–446, Lecture Notes in Computer Science, Vol. 600, Springer-Verlag, Berlin/New York.

37. Lynch, N. A., and Vaandrager, F. W. (1993), "Forward and Backward Simulations. I. Untimed Systems," Report CS-R9313, CWI, Amsterdam; also (1995) MIT/LCS/TM-486. b, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

38. Lynch, N. A., and Vaandrager, F. W. (1993), "Forward and Backward Simulations. II. Timing-Based Systems," Report CS-R9314, CWI, Amsterdam; also (1995) MIT/LCS/TM-487. c, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

39. Merritt, M. (1990), Completeness theorems for automata, *in* "Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 544–560, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.

40. Milner, R. (1971), An algebraic definition of simulation between programs, *in* "Proceedings, 2nd Joint Conference on Artificial Intelligence," pp. 481–489, BCS; also available as Report No. CS-205, Computer Science Department, Stanford University, February 1971.

41. Milner, R. (1989), "Communication and Concurrency," Prentice-Hall International, Englewood Cliffs, NJ.

42. Nipkow, T. (1990), Formal verification of data type refinement—Theory and practice, *in* "Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook, The Netherlands, May/June 1989" (J. W. de Bakker, W. P. de Roever, and G. Rozenberg, Eds.), pp. 561–591, Lecture Notes in Computer Science, Vol. 430, Springer-Verlag, Berlin/New York.

43. Owicki, S., and Gries, D. (1976), An axiomatic proof technique for parallel programs, *Acta Informat.* **6**(4), 319–340.

44. Park, D. M. R. (1981), Concurrency and automata on infinite sequences, *in* "5th GI Conference, Karlsruhe, Germany, March 1981" (P. Deussen, Ed.), pp. 167–183, Lecture Notes in Computer Science, Vol. 104, Springer-Verlag, Berlin/New York.

45. Reynolds, J. C. (1981), "The Craft of Programming," Prentice-Hall International, Englewood Cliffs, NJ.

46. Sistla, A. P. (1991), Proving correctness with respect to nondeterministic safety specifications, *Inform. Process. Lett.* **39**(1), 45–49.

47. Stark, E. W. (1988), Proving entailment between conceptual state specifications, *Theoret. Comput. Sci.* **56**, 135–154.

48. von Wright, J. (1994), The lattice of data refinement, *Acta Informat.* **31**, 105–135.