

An Assume-Guarantee Rule for Checking Simulation

THOMAS A. HENZINGER, SHAZ QADEER, SRIRAM K. RAJAMANI,
and SERDAR TASIRAN

University of California, Berkeley

The simulation preorder on state transition systems is widely accepted as a useful notion of refinement, both in its own right and as an efficiently checkable sufficient condition for trace containment. For composite systems, due to the exponential explosion of the state space, there is a need for decomposing a simulation check of the form $P \preceq_s Q$, denoting “ P is simulated by Q ,” into simpler simulation checks on the components of P and Q . We present an assume-guarantee rule that enables such a decomposition. To the best of our knowledge, this is the first assume-guarantee rule that applies to a refinement relation different from trace containment. Our rule is circular, and its soundness proof requires induction on trace trees. The proof is constructive: given simulation relations that witness the simulation preorder between corresponding components of P and Q , we provide a procedure for constructing a witness relation for $P \preceq_s Q$. We also extend our assume-guarantee rule to account for fairness constraints on transition systems.

Categories and Subject Descriptors: D.2.4 [Software Engineering]: Software/Program Verification—*formal methods*; F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs

General Terms: Verification

Additional Key Words and Phrases: Simulation relations, verification rules

1. INTRODUCTION

As a natural consequence of the design and verification process, systems are typically described at several different levels of abstraction. While performing verification in a hierarchical manner, one needs to check proof obligations of the form $P \preceq Q$, where P and Q are system descriptions and \preceq is a preorder on system descriptions. The assertion $P \preceq Q$ holds if P describes the same system as Q , but possibly on a finer level of detail (or equivalently, Q describes the same

A preliminary version of this work appeared in the *Proceedings of the 2nd International Conference on Formal Methods in Computer-Aided Design*, Lecture Notes in Computer Science, vol. 1522. Springer-Verlag, New York, 1998, pp. 421–432.

Authors' address: Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770; email: tah@eecs.berkeley.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dep., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2002 ACM 0164-0925/02/0100–0051 \$5.00

system as P but possibly on a coarser level of abstraction). For example, P may be an RTL (Register Transfer Level) description of a pipelined processor, and Q may be an ISA (Instruction Set Architecture) description of the same processor. The assertion $P \preceq Q$ is therefore variously pronounced as “ P implements Q ,” or “ P refines Q ,” or “ Q specifies P ,” or “ Q abstracts P .”

From a mathematical point of view, a popular choice for the preorder \preceq is *trace containment*. In this case, $P \preceq Q$ asserts that every sequence of inputs and outputs that is possible for P is also possible for Q (at the same time, the more abstract specification Q may allow some traces that are not realized by the more concrete implementation P). While simple and intuitive, trace containment has several shortcomings. First, it is a practical impossibility to check trace containment automatically for all but the smallest examples, because the check is exponential in the number of states of Q if Q is nondeterministic (as specifications often are). Second, if the design is being carried out in a top-down fashion, the system description P fleshes out detail that is left open in system description Q and there is a much tighter relation between P and Q than trace containment would indicate; namely, each implementation state of P corresponds to a specification state of Q . This tighter relation is captured mathematically by the notion of a *simulation relation*. Intuitively, Q simulates P if, starting from the initial states and continuing ad infinitum, every input–output pair of P can be matched by the same input–output pair in Q [Milner 1971]. Clearly, if Q simulates P , then every trace of P is also a trace of Q . The converse is not true; that is, simulation is a stronger requirement than trace containment. However, it is generally believed that trace containment without simulation is more often than not due to coincidence rather than systematic design [Kurshan 1994].

While trace containment is defined *globally*, for input–output sequences of arbitrary length, simulation is defined *locally*, by considering individual input–output pairs for all states. It is this locality in the definition of simulation that leads to significant advantages. First, if Q is claimed to simulate P , then a witness to this claim can be produced in the form of a relation between states of P and states of Q , and the witness can be efficiently checked for correctness (the check is linear in the number of states of P and Q). Such witness relations are widely used in verification methods and tools, under various names like homomorphisms [Kurshan 1994] and refinement mappings [Abadi and Lamport 1991; Lynch 1996]. Second, even if no witness is given, the existence of a simulation can be checked in polynomial time (the check is quadratic in the number of states of P and Q). The number of states of a system, however, depends exponentially on the size of the system description (note that n Boolean variables give rise to 2^n states—this is the *state-explosion problem*). Thus, even algorithms that are linear in the number of states are often infeasible in practice, and techniques have been studied for dividing a given verification task into simpler subtasks.

Compositional techniques for dividing the verification task $P \preceq Q$ into simpler subtasks are guided by the structures of P and Q . If the refinement relation \preceq is interpreted as trace containment, a number of compositional techniques are known. Specifically, if $P = P_1 \parallel P_2$ (i.e., P consists of components P_1

and P_2) and $Q = Q_1 \parallel Q_2$, then in order to check $P \preceq Q$, it suffices to check that $P_1 \preceq Q_1$ and $P_2 \preceq Q_2$. This *compositional principle* for trace containment is propositionally valid whenever parallel composition corresponds to trace intersection (replace \parallel by conjunction, and \preceq by implication). Unfortunately, the compositional principle is often not helpful, because P_1 typically refines Q_1 only when constrained by an environment that behaves like P_2 , and similarly, P_2 may refine Q_2 only when constrained by an environment that behaves like P_1 .

Under certain modeling assumptions (nonblocking and finite nondeterminism), the compositional principle can be strengthened to an *assume-guarantee principle* [Abadi and Lamport 1995; Alur and Henzinger 1996; McMillan 1997]: in order to check $P \preceq Q$, it suffices to check that $P_1 \parallel Q_2 \preceq Q_1$ and $Q_1 \parallel P_2 \preceq Q_2$. Three observations about this proof rule are important. First, the rule addresses the issue that the environment of P_1 may have to be suitably constrained in order to implement Q_1 , and similarly for P_2 . Second, the rule avoids reasoning about the compound implementation $P_1 \parallel P_2$, which typically has the largest of the involved state spaces (an implementation P_i usually has many more state variables than the corresponding specification Q_i).¹ Third, unlike the compositional principle, the assume-guarantee principle is circular: when establishing the specification Q_1 , it is assumed that the specification Q_2 is satisfied, and vice-versa. Indeed, the assume-guarantee principle is not propositionally valid (its proof requires induction on the length of traces).

By contrast to the case of trace containment, if the refinement relation \preceq is interpreted as “is simulated by,” denoted by \preceq_s , then little is known about compositional techniques other than the fact that the compositional principle remains valid whenever parallel composition corresponds to the intersection of trees whose branches are traces (this is because Q simulates P iff every trace tree of P is also a trace tree of Q). In particular, it would be useful to have an assume-guarantee principle for simulation, which, given witnesses for the two subtasks $P_1 \parallel Q_2 \preceq_s Q_1$ and $Q_1 \parallel P_2 \preceq_s Q_2$, lets us construct a witness for $P \preceq_s Q$. In this article, we show that the assume-guarantee principle is sound for simulation under the same assumptions under which it is sound for trace containment. Second, we show how the compound witness can be constructed from the witnesses for the subtasks. Third, we show that in analogy to the case of trace containment, the assume-guarantee principle for simulation can be extended to account for fairness constraints in system descriptions [Abadi and Lamport 1995; Alur and Henzinger 1996].

We illustrate the assume-guarantee rule for simulation using an example. Figures 1 and 2 show Moore-machine specifications and implementations for a sender and receiver in a communication protocol. We use a synchronous notion of composition: the composition of two Moore machines has both components execute simultaneously in each step, each providing inputs to the other. Let us first consider the specifications of the sender (S') and receiver (R'). Each state

¹Moreover, when performing the check $P_1 \parallel Q_2 \preceq Q_1$, the specification Q_2 can often be simplified to Q'_2 , such that (1) Q'_2 only contains the features of Q_2 that constrain P_1 and (2) it is straightforward to show $Q_2 \preceq Q'_2$ (for example, Q'_2 may be obtained from Q_2 by removing some state variables).

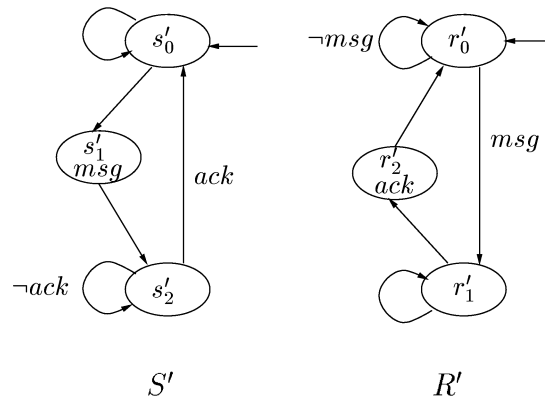


Fig. 1. Specifications of the sender and the receiver.

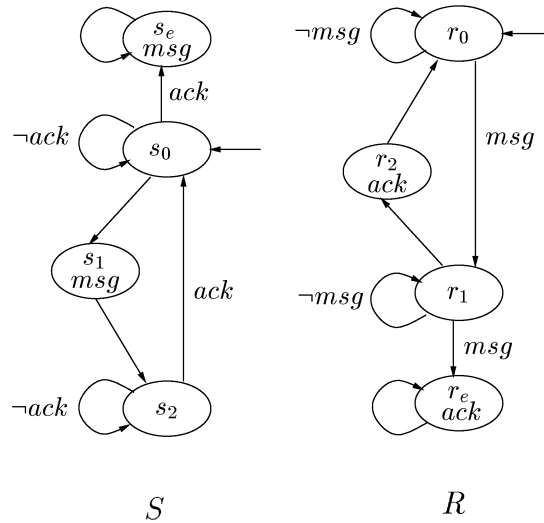


Fig. 2. Implementations of the sender and the receiver.

is labeled with output propositions that are true in the state, and each arc is labeled with conditions on inputs that need to be satisfied for the arc to be taken. A state without label means that no output propositions are true, and an arc without label means that there is no condition on the inputs. The initial states s'_0 of the sender and r'_0 of the receiver are marked using arrows. The sender has one output proposition, msg , which is true whenever a message is produced, and one input proposition, ack , which is used to acknowledge the receipt of a message by the receiver. The receiver has ack as its only output proposition and msg as its only input proposition. The sender starts at s'_0 and can stay there for an arbitrary number of steps. It can then nondeterministically produce a message by moving to s'_1 . Once at s'_1 , it moves to s'_2 in the next step. Then, the sender waits at s'_2 until it receives an ack . On receiving the ack , it goes back to s'_0 . The receiver starts at r'_0 and moves to r'_1 on seeing a msg . It can stay at

r'_1 for an arbitrary number of steps, and then nondeterministically moves to r'_2 . After acknowledging at r'_2 , the receiver moves back to r'_0 in the next step.

Let us now consider the implementations of the sender (S) and receiver (R). If S receives an *ack* while at s_0 , it goes to an “error state” s_e , from which it keeps sending messages in every step. Such a behavior is not allowed by the specification S' . Thus, S' does not simulate S . However, after composing with the specification of the receiver, it can be seen that $S \parallel R' \preceq_s S'$, because no acknowledgments are received by S while at s_0 . The relation $\Theta_S = \{(\langle s_0, r'_0 \rangle, s'_0), (\langle s_1, r'_0 \rangle, s'_1), (\langle s_2, r'_1 \rangle, s'_2), (\langle s_2, r'_2 \rangle, s'_2)\}$ is a witness to the simulation. Similarly, R' does not simulate R , but $S' \parallel R \preceq_s R'$ with the relation $\Theta_R = \{(\langle s'_0, r_0 \rangle, r'_0), (\langle s'_1, r_0 \rangle, r'_0), (\langle s'_2, r_1 \rangle, r'_1), (\langle s'_2, r_2 \rangle, r'_2)\}$ as witness. In Section 3, we prove that the existence of simulation relations from $S \parallel R'$ to S' and from $S' \parallel R$ to R' is sufficient to conclude the existence of a simulation relation Ω from $S \parallel R$ to $S' \parallel R'$, and we give a procedure for constructing Ω from Θ_S and Θ_R . For our example, the witness simulation relation Ω is $\{(\langle s_0, r_0 \rangle, \langle s'_0, r'_0 \rangle), (\langle s_1, r_0 \rangle, \langle s'_1, r'_0 \rangle), (\langle s_2, r_1 \rangle, \langle s'_2, r'_1 \rangle), (\langle s_2, r_2 \rangle, \langle s'_2, r'_2 \rangle)\}$.

We use Moore machines to model systems for simplicity. Our results apply to other nonblocking and finitely nondeterministic models such as (Fair) Reactive Modules [Alur and Henzinger 1996]. Section 2 defines Moore machines and establishes the connection between trace-tree containment and simulation. In Section 3, we prove the validity of the assume-guarantee rule for the simulation preorder and describe how the compound witness simulation relation is constructed from the witnesses for the components. Section 4 defines simulation on fair Moore machines. The assume-guarantee rule for fair simulation is proved in Section 5.

2. SIMULATION RELATIONS ON MOORE MACHINES

2.1 Moore Machines

A *Moore machine* is a tuple $\langle S, s_0, I, O, L, R \rangle$, where

- S is the set of states,
- $s_0 \in S$ is the initial state,
- I is the set of input propositions,
- O is the set of output propositions, which is disjoint from I ,
- $L: S \rightarrow 2^O$ is a function that labels each state with the set of output propositions true in that state, and
- $R \subseteq S \times 2^I \times S$ is the transition relation. Each transition is labeled with a set of input propositions, which need to be true to take the transition. We write $R(s, i, \tilde{s})$ as shorthand for $(s, i, \tilde{s}) \in R$.

We restrict our attention to Moore machines that satisfy the following two properties:

- (1) *Nonblocking*. For all states $s \in S$ and inputs $i \subseteq I$, there exists a state \tilde{s} such that $R(s, i, \tilde{s})$.

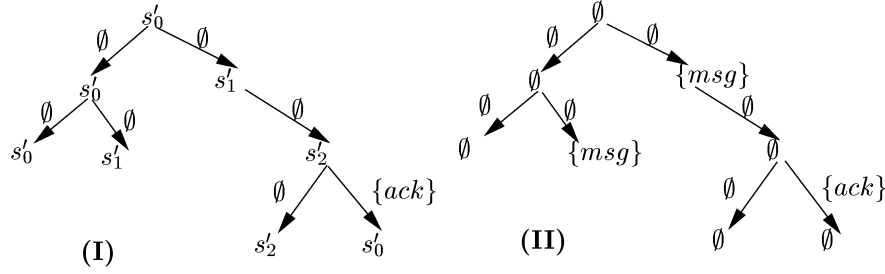


Fig. 3. (I) is a run tree for the Moore machine S' of Figure 1, and (II) is the corresponding trace tree.

- (2) *Finite nondeterminism.* For all states $s \in S$, inputs $i \subseteq I$, and outputs $o \subseteq O$, there are at most a finite number of states \bar{s} such that $R(s, i, \bar{s})$ and $L(\bar{s}) = o$.²

2.2 Run Trees and Trace Trees

Let \mathcal{O} be an ordinal large enough to well-order the transitions of the Moore machines under consideration (e.g., $\mathcal{O} = \mathbb{N}$ for Moore machines with a countable number of transitions), and let \mathcal{O}^* be the set of finite sequences of elements in \mathcal{O} . A *tree* is a nonempty, finite or infinite set $\tau \subseteq \mathcal{O}^*$ such that if $xn \in \tau$, for $x \in \mathcal{O}^*$ and $n \in \mathcal{O}$, then (1) $x \in \tau$ and (2) $xm \in \tau$ for all $0 \leq m < n$. The elements of τ represent nodes: the empty sequence ϵ is the root of τ , and for each node x , the nodes of the form xn , for $n \in \mathcal{O}$, are the children of x . An *edge* of τ is a pair $\langle x, xn \rangle$ of nodes $x, xn \in \tau$. A *path* ρ of τ is a finite or infinite set $\rho \subseteq \tau$ of nodes that satisfies the following three conditions: (1) $\epsilon \in \rho$, (2) for each node $x \in \rho$, there exists at most one $n \in \mathcal{O}$ with $xn \in \rho$, and (3) if $xn \in \rho$, then $x \in \rho$. Given a pair of sets A and B , an $\langle A, B \rangle$ -labeled tree is a triple $\langle \tau, \lambda, \delta \rangle$, where τ is a tree, λ is a node labeling function that maps every node $x \in \tau$ to an element $\lambda(x) \in A$, and δ is an edge labeling function that maps every edge $\langle x, xn \rangle$ of τ to an element $\delta(\langle x, xn \rangle) \in B$. Then, every path $\rho = \{\epsilon, n_0, n_0n_1, \dots\}$ of τ induces a sequence $\Gamma_{\lambda, \delta}(\rho) = \lambda(\epsilon) \cdot \delta(\langle \epsilon, n_0 \rangle) \cdot \lambda(n_0) \cdot \delta(\langle n_0, n_0n_1 \rangle) \cdot \lambda(n_0n_1) \cdot \dots$ of alternating labels from A and B .

A *run tree* of the Moore machine $P = \langle S, s_0, I, O, L, R \rangle$ is a $\langle S, 2^I \rangle$ -labeled tree $T = \langle \tau, \lambda, \delta \rangle$ such that $\lambda(\epsilon) = s_0$, and for all edges $\langle x, xn \rangle$ of τ , we have $R(\lambda(x), \delta(\langle x, xn \rangle), \lambda(xn))$. In other words, the nodes of a run tree are labeled by the states of P , starting with the initial state at the root, and the edges, which are labeled by inputs, correspond to transitions of P . The run tree T is *maximal* if for every node $x \in \tau$, if $R(\lambda(x), i, s)$ for some input $i \subseteq I$ and state $s \in S$, then there exists a node $xn \in \tau$ such that $\delta(\langle x, xn \rangle) = i$ and $\lambda(xn) = s$. A *trace tree* of P is a $\langle 2^O, 2^I \rangle$ -labeled tree $T' = \langle \tau, \lambda', \delta \rangle$ such that there exists a run tree $T = \langle \tau, \lambda, \delta \rangle$ of P , and for every node $x \in \tau$, we have $L(\lambda(x)) = \lambda'(x)$. For brevity, we write $T' = L(T)$. See Figure 3 for an example.

²For simplicity, we consider Moore machines with a single initial state. Our results apply if there are multiple initial states, as long as the initial states satisfy finite nondeterminism; that is, for all outputs $o \subseteq O$, there are at most a finite number of initial states s such that $L(s) = o$.

2.3 Tree Containment

Consider a Moore machine P with input propositions I^P and output propositions O^P , and a Moore machine Q with input propositions I^Q and output propositions O^Q . We say that Q is *refinable* by P if (1) $O^Q \subseteq O^P$ and (2) $I^Q \subseteq I^P \cup O^P$. Note that I^Q can contain elements from O^P ; this is to allow specifications Q that do not constrain some of the implementation (P) outputs to treat them as unconstrained inputs. Suppose that Q is refinable by P . Let $T = \langle \tau, \lambda, \delta \rangle$ be a trace tree of P . The *projection* of T onto Q is the $\langle 2^{O^Q}, 2^{I^Q} \rangle$ -labeled tree $[T]_Q = \langle \tau, \lambda', \delta' \rangle$ such that for every node $x \in \tau$, we have $\lambda'(x) = \lambda(x) \cap O^Q$, and for every edge $\langle x, xn \rangle$ of τ , we have $\delta'(\langle x, xn \rangle) = (\delta(\langle x, xn \rangle) \cup \lambda(x)) \cap I^Q$. We say that Q *tree contains* P if (1) Q is refinable by P , and (2) for every trace tree T of P , the projection $[T]_Q$ is a trace tree of Q .

2.4 Composition

Two Moore machines $P = \langle S^P, s_0^P, I^P, O^P, L^P, R^P \rangle$ and $Q = \langle S^Q, s_0^Q, I^Q, O^Q, L^Q, R^Q \rangle$ are *composable* if $O^P \cap O^Q = \emptyset$. If P and Q are composable, then their *composition* is the Moore machine $P \parallel Q = \langle S, s_0, I, O, L, R \rangle$, where

- $S = S^P \times S^Q$,
- $s_0 = \langle s_0^P, s_0^Q \rangle$,
- $I = (I^P \cup I^Q) \setminus (O^P \cup O^Q)$,
- $O = O^P \cup O^Q$,
- $L(\langle p, q \rangle) = L^P(p) \cup L^Q(q)$ for all states $\langle p, q \rangle \in S$,
- $R(\langle p, q \rangle, i, \langle \tilde{p}, \tilde{q} \rangle)$ iff $R^P(p, (i \cup L^Q(q)) \cap I^P, \tilde{p})$ and $R^Q(q, (i \cup L^P(p)) \cap I^Q, \tilde{q})$ for all states $\langle p, q \rangle, \langle \tilde{p}, \tilde{q} \rangle \in S$ and inputs $i \subseteq I$.

Note that if both P and Q are nonblocking and finitely nondeterministic, then so is $P \parallel Q$.

The observable branching behavior of a Moore machine is characterized by its set of trace trees. The composition of two Moore machines results in the intersection of the corresponding sets of trace trees, after appropriate projections on the inputs and outputs. Note that if the Moore machines P and Q are composable, then P and Q are both refinable by $P \parallel Q$, so for each trace tree T of $P \parallel Q$, the projections $[T]_P$ and $[T]_Q$ are defined.

PROPOSITION 1. *Consider two composable Moore machines P and Q . Then T is a trace tree of $P \parallel Q$ iff (1) $[T]_P$ is a trace tree of P , and (2) $[T]_Q$ is a trace tree of Q .*

2.5 Simulation

Consider two Moore machines $P = \langle S^P, s_0^P, I^P, O^P, L^P, R^P \rangle$ and $Q = \langle S^Q, s_0^Q, I^Q, O^Q, L^Q, R^Q \rangle$ such that Q is refinable by P . A binary relation $\Theta \subseteq S^P \times S^Q$ is a *simulation relation* from P to Q if the following three conditions hold:

- (1) $(s_0^P, s_0^Q) \in \Theta$.
- (2) For all $(p, q) \in \Theta$, we have $L^P(p) \cap O^Q = L^Q(q)$.

- (3) For all $(p, q) \in \Theta$ and all $i \subseteq I^P$ and $\bar{p} \in S^P$ such that $R^P(p, i, \bar{p})$, there exists $\bar{q} \in S^Q$ such that $R^Q(q, (i \cup L^P(p)) \cap I^Q, \bar{q})$ and $(\bar{p}, \bar{q}) \in \Theta$.

Put in words, (1) the initial states of the two machines must be related by Θ , (2) the outputs of related states must correspond, and (3) from a related pair of states, on every input, Q must be able to match every move of P to a related pair of states. We say that Q *simulates* P , and write $P \preceq_s Q$, if (1) Q is refinable by P , and (2) there exists a simulation relation Θ from P to Q . The relation Θ is said to *witness* the simulation. Furthermore, if $(p, q) \in \Theta$, then the state q of Q is said to *simulate* the state p of P .

The binary relation \preceq_s is a preorder (i.e., reflexive and transitive) on Moore machines. It is also immediate from the definitions that Q simulates P iff in a game of a protagonist playing on the states of Q against an antagonist playing on the states of P , the protagonist can match every move of the antagonist ad infinitum. The game starts at the initial states s_0^P and s_0^Q , and proceeds in a sequence of rounds. In every round, if the game is at states p and q , the antagonist chooses an input i and a successor state \bar{p} of p , and the protagonist matches by choosing on the corresponding input $(i \cup L^P(p)) \cap I^Q$ a successor state \bar{q} of q such that the output of \bar{q} corresponds to the output of \bar{p} , that is, $L^Q(\bar{q}) = L^P(\bar{p}) \cap O^Q$.

PROPOSITION 2. *Consider two Moore machines P and Q . Then $P \preceq_s Q$ iff Q tree contains P .³*

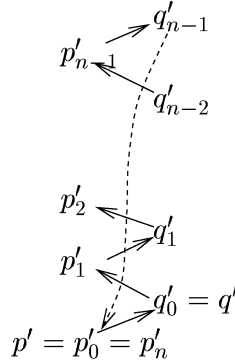
PROOF. Suppose that $P \preceq_s Q$ with witnessing simulation relation Θ . From every run tree T of P , we can construct a run tree T' of Q , by relabeling the nodes of T as prescribed by Θ , such that $[L^P(T)]_Q = L^Q(T')$. It follows that simulation implies tree containment. For the reverse implication, consider a maximal run tree T of P , and a run tree T' of Q such that $[L^P(T)]_Q = L^Q(T')$. By relating the labels of corresponding nodes in T and T' , we obtain a simulation relation from P to Q . \square

From Propositions 1 and 2, we can derive two useful relationships between composition and simulation that, together, we call the *compositionality property* of the simulation preorder \preceq_s on Moore machines. First, for all Moore machines P and Q , if P is composable with Q , then $P \parallel Q \preceq_s P$. Second, for all Moore machines P , P' , and Q , if $P \preceq_s P'$ and P is composable with Q , then P' is composable with Q and $P \parallel Q \preceq_s P' \parallel Q$. In the next section, we show a stronger relationship between composition and simulation, which is not implied by Propositions 1 and 2, but depends also on the nonblocking and finite-nondeterminism properties of Moore machines.

3. ASSUME-GUARANTEE RULE FOR SIMULATION

Suppose we are given a specification $P' \parallel Q'$ and an implementation $P \parallel Q$, where $P \parallel Q' \preceq_s P'$ and $P' \parallel Q \preceq_s Q'$. Consider a specific state $\langle p, q' \rangle$ of $P \parallel Q'$. Suppose

³If the assumption of finite nondeterminism is used, then a stronger version of this proposition can be proved, namely, that the simulation preorder is equivalent to the containment of *finite* trace trees [Rajamani 1999].


 Fig. 4. Illustration of the definition of Ω .

that this state is simulated by state p' of P' . Further, suppose that there exists q such that state $\langle p', q \rangle$ of $P' \parallel Q$ is simulated by state q' of Q' . Then, it seems plausible (and indeed it is true, as we show below) that state $\langle p, q \rangle$ of $P \parallel Q$ is simulated by state $\langle p', q' \rangle$ of $P' \parallel Q'$. A difficulty arises when state $\langle p', q \rangle$ of $P' \parallel Q$ is simulated by a state q'' of Q' that is different from q' . We then examine the state $\langle p, q'' \rangle$ of $P \parallel Q'$ and find a state p'' of P' that simulates it. We continue by finding a state of Q' that simulates state $\langle p'', q \rangle$ of $P' \parallel Q$, etc. In this way, if we reach a cycle that includes p' and q' , we are still able to show that state $\langle p, q \rangle$ of $P \parallel Q$ is simulated by state $\langle p', q' \rangle$ of $P' \parallel Q'$. Since our Moore machines have only single initial states, such a cycle must exist for the initial states, satisfying condition (1) for simulation relations. Finite nondeterminism ensures that condition (3) for simulation relations is satisfied as well.

THEOREM 3. *Let P , Q , P' , and Q' be Moore machines such that P is composable with Q , and P' is composable with Q' , and every input proposition of $P' \parallel Q'$ is either an input or output proposition of $P \parallel Q$. Suppose that $P \parallel Q' \preceq_s P'$ and $P' \parallel Q \preceq_s Q'$ with witnessing simulation relations Θ_P and Θ_Q , respectively. Then, we can construct a simulation relation Ω from $P \parallel Q$ to $P' \parallel Q'$.*

PROOF. First, we show that $P' \parallel Q'$ is refinable by $P \parallel Q$. Since $P \parallel Q' \preceq_s P'$, every output proposition of P' is an output proposition of $P \parallel Q'$. However, as P' and Q' are composable, the output propositions of P' and Q' are disjoint. Thus, every output proposition of P' is an output proposition of P , and symmetrically, $O^{Q'} \subseteq O^Q$. It follows that $O^{P' \parallel Q'} \subseteq O^{P \parallel Q}$. The condition on inputs, $I^{P' \parallel Q'} \subseteq I^{P \parallel Q} \cup O^{P \parallel Q}$, holds by assumption.

Define the relation $\Omega \subseteq (S^P \times S^Q) \times (S^{P'} \times S^{Q'})$ as follows: $(\langle p, q \rangle, \langle p', q' \rangle) \in \Omega$ iff there exist states $p'_0, p'_1, \dots, p'_n \in S^{P'}$ and $q'_0, q'_1, \dots, q'_{n-1} \in S^{Q'}$ such that (see Figure 4)

- $p'_0 = p'_n = p'$ and $q'_0 = q'$, and
- for all $0 \leq i < n$, we have $(\langle p'_i, q \rangle, q'_i) \in \Theta_Q$ and $(\langle p, q'_i \rangle, p'_{i+1}) \in \Theta_P$.

We show that conditions (1)–(3) of the definition of a simulation relation from $P \parallel Q$ to $P' \parallel Q'$ are satisfied by Ω . In the following (for (2) and (3)), assume that $(\langle p, q \rangle, \langle p', q' \rangle) \in \Omega$, that is, there exist $p'_0, p'_1, \dots, p'_n \in S^{P'}$ and $q'_0, q'_1, \dots, q'_{n-1} \in S^{Q'}$ that satisfy the conditions mentioned above.

- (1) Since Θ_P and Θ_Q are simulation relations, we have $(\langle s_0^P, s_0^{Q'} \rangle, s_0^{P'}) \in \Theta_P$ and $(\langle s_0^{P'}, s_0^Q \rangle, s_0^{Q'}) \in \Theta_Q$. Let $n = 1$, $p'_0 = p'_1 = s_0^{P'}$, and $q'_0 = s_0^{Q'}$. Then, $(\langle s_0^P, s_0^Q \rangle, \langle s_0^{P'}, s_0^{Q'} \rangle) \in \Omega$.
- (2) Since $(\langle p'_0, q \rangle, q'_0) \in \Theta_Q$ and Θ_Q is a simulation relation, $L^Q(q) \cap O^{Q'} = L^Q(q'_0)$. Since $q'_0 = q'$, we have $L^Q(q) \cap O^{Q'} = L^Q(q')$. Also, $(\langle p, q'_i \rangle, p'_{i+1}) \in \Theta_P$ for all i from 0 to $n - 1$. Therefore, $L^P(p) \cap O^{P'} = L^P(p'_{i+1})$ for all $0 \leq i < n$. Since, $p'_n = p'_0 = p'$, we have $L^P(p) \cap O^{P'} = L^P(p')$. Hence, we obtain $L^{P \parallel Q}(\langle p, q \rangle) \cap O^{P \parallel Q'} = L^{P' \parallel Q'}(\langle p', q' \rangle)$.
- (3) Let $R^{P \parallel Q}(\langle p, q \rangle, i, \langle \tilde{p}, \tilde{q} \rangle)$ for some $\langle \tilde{p}, \tilde{q} \rangle$ and $i \subseteq I^{P \parallel Q}$. Define $i' = i \cup L^P(p) \cup L^Q(q)$. In the following, whenever we use i' as the input for some Moore machine K , the intention is to take the projection $i' \cap I^K$ onto the set of input propositions of K . We want to show that there exists $(\langle \tilde{p}', \tilde{q}' \rangle) \in \Omega$ such that $R^{P' \parallel Q'}(\langle p', q' \rangle, i' \cap I^{P' \parallel Q'}, \langle \tilde{p}', \tilde{q}' \rangle)$.

—Since all machines are nonblocking, we have $R^{P'}(p'_0, i', \tilde{p}'_0)$ for some \tilde{p}'_0 . Hence, $R^{P' \parallel Q'}(\langle p'_0, q \rangle, i', \langle \tilde{p}'_0, \tilde{q} \rangle)$.

—The fact that $(\langle p'_0, q \rangle, q'_0) \in \Theta_Q$ implies that there exists some \tilde{q}'_0 such that $R^{Q'}(q'_0, i', \tilde{q}'_0)$ and $(\langle \tilde{p}'_0, \tilde{q} \rangle, \tilde{q}'_0) \in \Theta_Q$.

— $R^{Q'}(q'_0, i', \tilde{q}'_0)$ implies that $R^{P' \parallel Q'}(\langle p, q'_0 \rangle, i', \langle \tilde{p}, \tilde{q}'_0 \rangle)$.

— $(\langle p, q'_0 \rangle, p'_1) \in \Theta_P$, therefore there exists a state \tilde{p}'_1 such that $R^{P'}(p'_1, i', \tilde{p}'_1)$ and $(\langle \tilde{p}, \tilde{q}'_0 \rangle, \tilde{p}'_1) \in \Theta_P$.

Repeating this process, we obtain $\tilde{p}'_0, \tilde{p}'_1, \tilde{p}'_2, \dots$ and $\tilde{q}'_0, \tilde{q}'_1, \tilde{q}'_2, \dots$ such that

—for all $k \in \mathbb{N}$, we have $R^{P'}(p'_k, i', \tilde{p}'_{kn})$ and $R^{Q'}(q'_k, i', \tilde{q}'_{kn})$, and

—for all $j \geq 0$, we have that $(\langle \tilde{p}'_j, \tilde{q} \rangle, \tilde{q}'_j) \in \Theta_Q$ and $(\langle \tilde{p}, \tilde{q}'_j \rangle, \tilde{p}'_{j+1}) \in \Theta_P$.

Since P' has finite nondeterminism, there exist $a, b \in \mathbb{N}$ with $b > a$ such that $\tilde{p}'_{an} = \tilde{p}'_{bn}$. Consider the states $\tilde{p}'_{an}, \tilde{p}'_{an+1}, \dots, \tilde{p}'_{bn} \in S^{P'}$ and $\tilde{q}'_{an}, \tilde{q}'_{an+1}, \dots, \tilde{q}'_{bn-1} \in S^{Q'}$. We know that for all $an \leq i < bn$, $(\langle \tilde{p}'_i, \tilde{q} \rangle, \tilde{q}'_i) \in \Theta_Q$ and $(\langle \tilde{p}, \tilde{q}'_i \rangle, \tilde{p}'_{i+1}) \in \Theta_P$. It follows from the definition of Ω that $(\langle \tilde{p}, \tilde{q} \rangle, \langle \tilde{p}'_{an}, \tilde{q}'_{an} \rangle) \in \Omega$. We also have $R^{P' \parallel Q'}(\langle p', q' \rangle, i' \cap I^{P' \parallel Q'}, \langle \tilde{p}'_{an}, \tilde{q}'_{an} \rangle)$. \square

4. SIMULATION RELATIONS ON FAIR MOORE MACHINES

4.1 Fair Moore Machines

Consider a Moore machine $P = \langle S, s_0, I, O, L, R \rangle$. A *run* of P is a finite or infinite sequence $\sigma = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} p_2 \xrightarrow{i_3} p_3 \dots$ of alternating states $p_k \in S$ and inputs $i_k \subseteq I$ such that $p_0 = s_0$, and $R(p_k, i_{k+1}, p_{k+1})$ for all $k \geq 0$. If σ is finite, then its *length* $|\sigma|$ is defined to be the number of states in σ ; if σ is infinite, then $|\sigma| = \omega$. For $0 \leq k < |\sigma|$, the k th *prefix* of σ is the finite run $\sigma_k = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \dots \xrightarrow{i_k} p_k$. The set of all finite runs of P is denoted by Σ^P .

A *fairness constraint* for the Moore machine P is a function that maps every infinite run of P to the binary set $\{fair, unfair\}$.⁴ We write F_{\emptyset}^P for the trivial fairness constraint that maps every infinite run of P to *fair*. A *fair Moore machine* $\mathcal{P} = \langle P, F \rangle$ consists of a Moore machine P and a fairness constraint F for P . A *fair run* of \mathcal{P} is either a finite run of P , or an infinite run σ of P such that $F(\sigma) = fair$. Note that every path ρ of a run tree $T = \langle \tau, \lambda, \delta \rangle$ of P induces a run $\Gamma_{\lambda, \delta}(\rho)$ of P . The run tree T of P is a *fair run tree* of \mathcal{P} if for every path ρ of τ , the induced run $\Gamma_{\lambda, \delta}(\rho)$ is a fair run of \mathcal{P} . A *fair trace tree* of \mathcal{P} is a trace tree T' of P such that there exists a fair run tree T of \mathcal{P} with $T' = L(T)$.

4.2 Fair Tree Containment and Composition

Consider two fair Moore machines $\mathcal{P} = \langle P, F^P \rangle$ and $\mathcal{Q} = \langle Q, F^Q \rangle$. We say that \mathcal{Q} *fair-tree contains* \mathcal{P} if (1) \mathcal{Q} is refinable by \mathcal{P} , and (2) for every fair trace tree T of \mathcal{P} , the projection $[T]_{\mathcal{Q}}$ is a fair trace tree of \mathcal{Q} . The fair Moore machines \mathcal{P} and \mathcal{Q} are *composable* if \mathcal{P} is composable with \mathcal{Q} . For a run $\sigma = \langle p_0, q_0 \rangle \xrightarrow{i_1} \langle p_1, q_1 \rangle \xrightarrow{i_2} \dots$ of $P \parallel Q$, the projection onto P is $[\sigma]_P = p_0 \xrightarrow{i'_1} p_1 \xrightarrow{i'_2} \dots$, where $i'_{k+1} = (i_{k+1} \cup L^Q(q_k)) \cap I^P$ for all $k \geq 0$. Note that $[\sigma]_P$ is a run of P . The projection $[\sigma]_Q$ onto Q is defined symmetrically. If \mathcal{P} and \mathcal{Q} are composable, then the *composition* is the fair Moore machine $\mathcal{P} \parallel \mathcal{Q} = \langle P \parallel Q, F \rangle$ such that for every infinite run σ of $P \parallel Q$, we have $F(\sigma) = fair$ iff both $F^P([\sigma]_P) = fair$ and $F^Q([\sigma]_Q) = fair$.

PROPOSITION 4. *Consider two composable fair Moore machines \mathcal{P} and \mathcal{Q} . Then T is a fair trace tree of $\mathcal{P} \parallel \mathcal{Q}$ iff (1) $[T]_P$ is a fair trace tree of \mathcal{P} , and (2) $[T]_Q$ is a fair trace tree of \mathcal{Q} .*

4.3 Fair Simulation

Consider two fair Moore machines $\mathcal{P} = \langle P, F^P \rangle$ and $\mathcal{Q} = \langle Q, F^Q \rangle$ such that \mathcal{Q} is refinable by \mathcal{P} . Intuitively, \mathcal{Q} fairly simulates \mathcal{P} if there is a strategy in the simulation game that matches every fair run of \mathcal{P} with a fair run of \mathcal{Q} [Henzinger et al. 1997]. Formally, a *simulation strategy* κ of \mathcal{Q} with respect to \mathcal{P} is a function from $\Sigma^P \times \Sigma^Q$ to S^Q . Let $\sigma = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \dots \xrightarrow{i_n} p_n$ be a finite run of P , and let $\sigma' = q_0 \xrightarrow{i'_1} q_1 \xrightarrow{i'_2} \dots \xrightarrow{i'_m} q_m$ be a finite run of Q . The result $q_{m+1} = \kappa(\langle \sigma, \sigma' \rangle)$ of applying the simulation strategy to the pair $\langle \sigma, \sigma' \rangle$ must satisfy the following: if

- (1) $n = m + 1$,
- (2) $(i_{k+1} \cup L^P(p_k)) \cap I^Q = i'_{k+1}$ for all $0 \leq k < m$, and
- (3) $\kappa(\langle \sigma_{k+1}, \sigma'_k \rangle) = q_{k+1}$ for all $0 \leq k < m$,

then $R^Q(q_m, (i_{m+1} \cup L^P(p_m)) \cap I^Q, q_{m+1})$ and $L^P(p_{m+1}) \cap O^Q = L^Q(q_{m+1})$. Given a finite or infinite run $\sigma = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \dots$ of P , the *outcome* $\kappa[\sigma]$ of the

⁴In practice, one often prefers fairness constraints that do not constrain the inputs [Abadi and Lamport 1995; Alur and Henzinger 1996; Lynch 1996]. However, such a *receptiveness* condition is not necessary for our results.

simulation strategy κ is the (unique) run $\sigma' = q_0 \xrightarrow{i'_1} q_1 \xrightarrow{i'_2} \dots$ of \mathcal{Q} such that (1) $|\kappa[\sigma]| = |\sigma|$, (2) for all $k \geq 0$, we have $i'_{k+1} = (i_{k+1} \cup L^P(p_k)) \cap I^{\mathcal{Q}}$, and (3) for all $k \geq 0$, we have $q_{k+1} = \kappa(\langle \sigma_{k+1}, \sigma'_k \rangle)$. The simulation strategy κ is *fair* if for every fair run σ of \mathcal{P} , the outcome $\kappa[\sigma]$ is a fair run of \mathcal{Q} . We say that \mathcal{Q} *fairly simulates* \mathcal{P} , and write $\mathcal{P} \preceq_s^F \mathcal{Q}$, if (1) \mathcal{Q} is refinable by \mathcal{P} , (2) $L^P(s_0^P) \cap O^{\mathcal{Q}} = L^{\mathcal{Q}}(s_0^{\mathcal{Q}})$, and (3) there exists a fair simulation strategy κ of \mathcal{Q} with respect to \mathcal{P} .

The relation \preceq_s^F is a preorder on fair Moore machines. Every simulation strategy κ of \mathcal{Q} with respect to \mathcal{P} induces a simulation relation Θ from \mathcal{Q} to \mathcal{P} , namely, $(p, q) \in \Theta$ iff there exists a finite run σ of \mathcal{Q} with final state p , and the outcome $\kappa[\sigma]$ has final state q .

PROPOSITION 5. *Consider two fair Moore machines $\mathcal{P} = \langle P, F^P \rangle$ and $\mathcal{Q} = \langle Q, F^{\mathcal{Q}} \rangle$. If $\mathcal{P} \preceq_s^F \mathcal{Q}$, then $P \preceq_s Q$.*

Conversely, for Moore machines P and Q , if $P \preceq_s Q$, then $\langle P, F_\emptyset^P \rangle \preceq_s^F \langle Q, F_\emptyset^{\mathcal{Q}} \rangle$. Using an argument similar to the equivalence proof between simulation and tree containment, fair simulation can be proved equivalent to fair-tree containment [Henzinger et al. 1997].

PROPOSITION 6. *Consider two fair Moore machines \mathcal{P} and \mathcal{Q} . Then $\mathcal{P} \preceq_s^F \mathcal{Q}$ iff \mathcal{Q} fair-tree contains \mathcal{P} .*

From Propositions 4 and 6, we obtain the compositionality property (as defined in Section 2) of the fair-simulation preorder \preceq_s^F on fair Moore machines.

5. ASSUME-GUARANTEE RULE FOR FAIR SIMULATION

For a fair Moore machine $\mathcal{P} = \langle P, F^P \rangle$, let $\text{Safe}(\mathcal{P}) = \langle P, F_\emptyset^P \rangle$, that is, the fairness constraint of \mathcal{P} is replaced by the trivial fairness constraint that maps every infinite run to *fair*. The following assume-guarantee rule for fair simulation was proved previously, by a similar argument, for fair trace containment [Alur and Henzinger 1996].

THEOREM 7. *Let \mathcal{P} , \mathcal{Q} , \mathcal{P}' , and \mathcal{Q}' be fair Moore machines such that \mathcal{P} is composable with \mathcal{Q} , and \mathcal{P}' is composable with \mathcal{Q}' , and every input proposition of $\mathcal{P}' \parallel \mathcal{Q}'$ is either an input or output proposition of $\mathcal{P} \parallel \mathcal{Q}$. If $\mathcal{P} \parallel \text{Safe}(\mathcal{Q}') \preceq_s^F \mathcal{P}'$ and $\mathcal{P}' \parallel \mathcal{Q} \preceq_s^F \mathcal{Q}'$, then $\mathcal{P} \parallel \mathcal{Q} \preceq_s^F \mathcal{P}' \parallel \mathcal{Q}'$.*

PROOF. Let $\mathcal{P} = \langle P, F^P \rangle$, $\mathcal{Q} = \langle Q, F^{\mathcal{Q}} \rangle$, $\mathcal{P}' = \langle P', F^{P'} \rangle$, and $\mathcal{Q}' = \langle Q', F^{Q'} \rangle$. From Proposition 5, we know that $P \parallel Q' \preceq_s P'$ and $P' \parallel Q \preceq_s Q'$. Consequently, by Theorem 3, we know that $P \parallel Q \preceq_s P' \parallel Q'$. Therefore, from Proposition 2, it follows that $P' \parallel Q'$ tree contains $P \parallel Q$. Hence, if T is a trace tree of $P \parallel Q$, then $[T]_{P' \parallel Q'}$ is a trace tree of $P' \parallel Q'$. It remains to be proved that if T is a fair trace tree of $\mathcal{P} \parallel \mathcal{Q}$, then $[T]_{P' \parallel Q'}$ is a fair trace tree of $\mathcal{P}' \parallel \mathcal{Q}'$.

For notational simplicity, we omit explicit projections in the following. Suppose that T is a fair trace tree of $\mathcal{P} \parallel \mathcal{Q}$. From Proposition 4, it follows that T is a fair trace tree of both \mathcal{P} and \mathcal{Q} . Furthermore, we know that T is a trace tree of $P' \parallel Q'$. Thus, T is a fair trace tree of $\text{Safe}(\mathcal{Q}')$. Again, by Proposition 4, it follows that T is a fair trace tree of $\mathcal{P} \parallel \text{Safe}(\mathcal{Q}')$. Since $\mathcal{P} \parallel \text{Safe}(\mathcal{Q}') \preceq_s^F \mathcal{P}'$, we conclude from Proposition 6 that T is a fair trace tree of \mathcal{P}' . Therefore, by

Proposition 4, as T is a fair trace tree of both \mathcal{P}' and \mathcal{Q} , it is a fair trace tree of $\mathcal{P}'\|\mathcal{Q}$. Since $\mathcal{P}'\|\mathcal{Q} \preceq_s^F \mathcal{Q}'$, we know that T is a fair trace tree of \mathcal{Q}' . Again by Proposition 4, as T is a fair trace tree of both \mathcal{P}' and \mathcal{Q}' , it is a fair trace tree of $\mathcal{P}'\|\mathcal{Q}'$. \square

Note the asymmetry in the antecedent of Theorem 7. The premise $\mathcal{P}\|\text{Safe}(\mathcal{Q}') \preceq_s^F \mathcal{P}'$ cannot be weakened to $\mathcal{P}\|\mathcal{Q}' \preceq_s^F \mathcal{P}'$ [Alur and Henzinger 1996]. The asymmetry is required to break the circularity in the premises. A similar verification rule for timed transition systems is presented in Tasiran [1998].

6. CONCLUSION

The focus of the work presented in this article is compositional refinement checking, where the notion of refinement is the simulation preorder \preceq_s . Under mild assumptions, we proved the soundness of an assume-guarantee principle that enables a refinement check of the form $P_1\|P_2 \preceq_s Q_1\|Q_2$ to be decomposed into the refinement checks $P_1\|Q_2 \preceq_s Q_1$ and $Q_1\|P_2 \preceq_s Q_2$. In this way, it is possible to avoid computation on the large state space of $P_1\|P_2$, which is typically the bottleneck in hierarchical verification. We presented a constructive proof of this rule: we build a witness for $P_1\|P_2 \preceq_s Q_1\|Q_2$ from simulation relations that witness $P_1\|Q_2 \preceq_s Q_1$ and $Q_1\|P_2 \preceq_s Q_2$. We further showed that, with the usual modifications, our result generalizes to systems with fairness constraints.

Assume-guarantee reasoning for trace containment is well studied. That the simulation preorder can be used in conjunction with an assume-guarantee principle has practical significance, because simulation checks can be performed locally and are therefore much less costly than trace-containment checks. Using our method for combining simulations that relate components in a design hierarchy, it is possible to construct simulations that relate all levels of the hierarchy, which, in practice, provides more confidence that the system is refined correctly through all levels.

REFERENCES

- ABADI, M. AND LAMPORT, L. 1991. The existence of refinement mappings. *Theoret. Comput. Sci.* 82, 2, 253–284.
- ABADI, M. AND LAMPORT, L. 1995. Conjoining specifications. *ACM Trans. Prog. Lang. Syst.* 17, 3, 507–534.
- ALUR, R. AND HENZINGER, T. 1996. Reactive modules. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, Los Alamitos, Calif., pp. 207–218.
- HENZINGER, T., KUPFERMAN, O., AND RAJAMANI, S. 1997. Fair simulation. In *CONCUR 97: Theories of Concurrency*. Lecture Notes in Computer Science, vol. 1243. Springer-Verlag, New York, 273–287.
- KURSHAN, R. 1994. *Computer-Aided Verification of Coordinating Processes*. Princeton University Press, Princeton, N.J.
- LYNCH, N. 1996. *Distributed Algorithms*. Morgan-Kaufmann, Reading, Pa.
- McMILLAN, K. 1997. A compositional rule for hardware design refinement. In *CAV 97: Computer-Aided Verification*. Lecture Notes in Computer Science, vol. 1254. Springer-Verlag, New York, 24–35.

- MILNER, R. 1971. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*. The British Computer Society, 481–489.
- RAJAMANI, S. 1999. New directions in refinement checking. Ph.D. dissertation, Univ. California, Berkeley, Calif.
- TASIRAN, S. 1998. Compositional and hierarchical techniques for the formal verification of real-time systems. Ph.D. dissertation, University of California, Berkeley, Calif.

Received December 1998; accepted August 1999