

An Assume-Guarantee Rule For Checking Simulation^{***}

Thomas A. Henzinger Shaz Qadeer Sriram K. Rajamani Serdar Taşiran

EECS Department, University of California at Berkeley, CA 94720-1770, USA
Email: {tah,shaz,sriramr,serdar}@eecs.berkeley.edu

Abstract. 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 $P \preceq_s Q$ 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 assumptions on transition systems.

1 Introduction

In hierarchical verification, we need 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 system as P but possibly on a coarser level of abstraction). For example, P may be an RTL-level description of a pipelined processor, and Q may be an ISA 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 .”

Mathematically, 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

* An abbreviated version of this paper appeared in the *Proceedings of the Second International Conference on Formal Methods in Computer-aided Design (FMCAD)*, Lecture Notes in Computer Science 1522, Springer-Verlag, 1998, pp. 421–432.

** This research was supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grant CCR-9504469, by the Defense Advanced Research Projects Agency grant NAG2-1214, by the Army Research Office MURI grant DAAH-04-96-1-0341, and by the Semiconductor Research Corporation contract 97-DC-324.041.

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, in top-down design, if system description P fleshes out detail that is left open in system description Q , 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 iff, 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 [Mil71]. 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 has been said that trace containment without simulation is more often than not due to coincidence rather than systematic design [Kur94].

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 [Kur94] and refinement mappings [AL91, Lyn96]. Second, even if no witness is available, 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 $P = P_1 \parallel P_2$ and $Q = Q_1 \parallel Q_2$, then in order to check $P \preceq Q$, it suffices to check both $P_1 \preceq Q_1$ 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 (namely, nonblocking and finite nondeterminism), the compositional principle can be strengthened to an

assume-guarantee principle [Sta85,CLM89,GL94,AL95,AH96,McM97]: in order to check $P \preceq Q$, it suffices to check both $P_1 \parallel Q_2 \preceq Q_1$ 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. Third, unlike the compositional principle, the assume-guarantee principle is circular and therefore 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,” 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 $P_1 \parallel Q_2 \preceq Q_1$ and $Q_1 \parallel P_2 \preceq Q_2$, lets us construct a witness for $P \preceq Q$. In this paper, we show that under the same modeling assumptions under which the assume-guarantee principle is sound for trace containment, it is also sound for simulation. 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 assumptions in system descriptions. As in the case of trace containment [AL95,AH96], the proof of soundness requires the modeling assumption of receptiveness [Dil89].

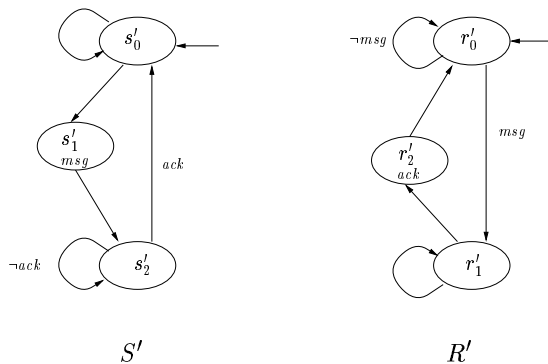


Fig. 1. Specifications of sender and receiver

We illustrate the assume-guarantee rule for simulation using an example. Figure 1 shows Moore-machine specifications for a sender S' and a receiver R' in a communication protocol. Each state is labeled with outputs that are true

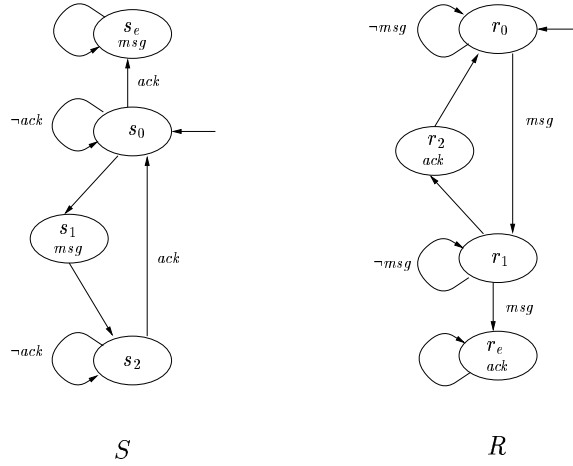


Fig. 2. Implementations of sender and receiver

in that state, and each arc is labeled with conditions on inputs that need to be satisfied for the arc to be taken. A state with no label means that no output propositions are true in that state, and an arc with no label means that there is no condition on input propositions to take that arc. The initial states s'_0 of the sender and r'_0 of the receiver are marked using arrows. The sender has one output proposition, namely msg , which is true whenever a message is produced, and one input proposition, namely 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 off at s'_0 and stays there for an arbitrary number of steps. It nondeterministically produces a message by moving to s'_1 . Once in s'_1 , it is forced to move to s'_2 in one step. Then, the sender waits in 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 stays in r'_1 for an arbitrary number of steps and nondeterministically moves to r'_2 . After acknowledging at r'_2 , the receiver moves back to r'_0 in the next step.

Figure 2 shows 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 is seen that $S \parallel R' \preceq_s S'$, because no acknowledgments can be received by S while at s_0 . The relation $\Theta_S = \{((s_0, r'_0), s'_0), ((s_1, r'_0), s'_1), ((s_2, r'_1), s'_2), ((s_2, r'_2), s'_2)\}$ is a witness to this simulation. Similarly, R' does not simulate R , but $S' \parallel R \preceq_s R'$ with the relation $\Theta_R = \{((s'_0, r_0), r'_0), ((s'_1, r_0), r'_0), ((s'_2, r_1), r'_1), ((s'_2, r_2), r'_2)\}$ 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 give a procedure to

construct Ω , from Θ_S and Θ_R . For our example, the witness simulation relation Ω is $\{(\langle s_0, r_0 \rangle, \langle s'_0, r'_0 \rangle), (\langle s_1, r_0 \rangle, \langle s'_1, r'_0 \rangle), (\langle s_2, r_1 \rangle, \langle s'_2, r'_1 \rangle), (\langle s_2, r_2 \rangle, \langle s'_2, r'_2 \rangle)\}$.

For simplicity we use Moore machines (with local Streett conditions) to model systems. Our results apply to other nonblocking, finitely nondeterministic, receptive models such as (Fair) Reactive Modules [AH96]. 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 proven in Section 5.

2 Simulation Relations on Moore Machines

Moore machines. A Moore machine is a tuple $P = \langle S^P, s^P, I^P, O^P, L^P, R^P \rangle$ where

- S^P is the set of states,
- $s^P \in S^P$ is the initial state,
- I^P is the set of input propositions,
- O^P is the set of output propositions disjoint from I^P ,
- $L^P : S^P \rightarrow \mathcal{P}(O^P)$ is a function that labels each state with the subset of output propositions true in that state, and
- $R^P \subseteq S^P \times \mathcal{P}(I^P) \times S^P$ is the transition relation. We write $R^P(s, i, s')$ as shorthand for $(s, i, s') \in R^P$.

We restrict our attention to Moore machines P satisfying the following two properties:

1. *Nonblocking:* For all $s \in S^P$ and $i \subseteq I^P$, there exists a state τ such that $R^P(s, i, \tau)$.
2. *Finite nondeterminism:* For all $s \in S^P$, $i \subseteq I^P$, and $o \subseteq O^P$, there are at most a finite number of states τ such that $R^P(s, i, \tau)$ and $L^P(\tau) = o$.¹

Run trees and trace trees. A (finite or infinite) *tree* is a set $\tau \subseteq \mathbb{N}^*$ such that if $xn \in \tau$, for $x \in \mathbb{N}^*$ and $n \in \mathbb{N}$, then $x \in \tau$ and $xm \in \tau$ for all $0 \leq m < n$. The elements of τ represent nodes: the empty word ϵ is the root of τ , and for each node x , the nodes of the form xn , for $n \in \mathbb{N}$, are the children of x . The number of children of node x is denoted by $\text{deg}(x)$. A tree τ is finite if τ is a finite set. The *depth* of a node $\tau \in x$ is defined inductively as follows: (1) the depth of ϵ is 0, and (2) if the depth of $x \in \tau$ is d , then the depth of xn is $d+1$. If τ is finite, then the depth of τ is defined as the maximum of depths over all nodes of τ . The nodes

¹ For simplicity, we consider Moore machines with single initial states. Our results apply if there are multiple initial states, as long as the initial states satisfy finite nondeterminism: for all $o \subseteq O^P$, there are only a finite number of initial states s such that $L^P(s) = o$.

of τ with no children are called *leaves* of τ . 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 $n \in \mathbb{N}$ 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, $\lambda : \tau \rightarrow A$ is a node labeling function that maps each node of τ to an element in A , and $\delta : \tau \times \tau \rightarrow B$ is an edge labeling function that maps each edge $\langle x, xn \rangle$ of τ to an element in B . Then, every path $\rho = \{\epsilon, n_0, n_0n_1, \dots\}$ of τ generates a sequence $\Gamma(\rho) = \lambda(\epsilon) \cdot \langle \delta(\epsilon, n_0), \lambda(n_0) \rangle \cdot \langle \delta(n_0, n_0n_1), \lambda(n_0n_1) \rangle \cdots$ in $A \times (B \times A)^* \cup A \times (B \times A)^\omega$.

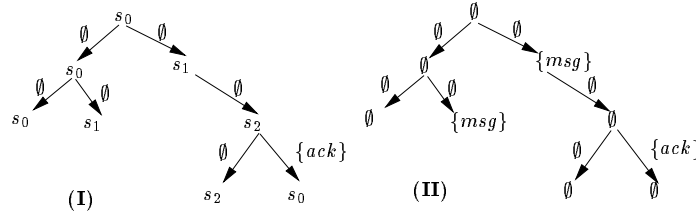


Fig. 3. (I) is an run tree for the Moore machine S in Figure 2, and (II) is the corresponding trace tree

An *run tree* T of P is a $\langle S^P, \mathcal{P}(I^P) \rangle$ -labeled tree $\langle \tau, \lambda, \delta \rangle$ such that $\lambda(\epsilon) = s^P$, and for all edges $\langle x, xn \rangle$ we have $R^P(\lambda(x), \delta(x, xn), \lambda(xn))$. Note that for any depth, P has at least one run tree, and possibly many. A *trace tree* $T' = \langle \tau, \lambda', \delta' \rangle$ of P is a $\langle \mathcal{P}(O^P), \mathcal{P}(I^P) \rangle$ -labeled tree such that there is a run tree of $T = \langle \tau, \lambda, \delta \rangle$ of P , and for every $x \in \tau$ we have $L^P(\lambda(x)) = \lambda'(x)$. For brevity we say $T' = L^P(T)$, and call T a *witness* to T' . See Figure 3 for an example.

Tree containment. Consider two Moore machines $P = \langle S^P, s^P, I^P, O^P, L^P, R^P \rangle$ and $Q = \langle S^Q, s^Q, I^Q, O^Q, L^Q, R^Q \rangle$. We say that Q is *refinable* by P if (1) $O^Q \subseteq O^P$, and (2) $I^Q \subseteq I^P \cup O^P$.

Suppose Q is refinable by P . Let $T = \langle \tau, \lambda, \delta \rangle$ be a trace tree of P . We say that the *projection* of T on Q is a trace tree $[T]_Q = \langle \tau, \lambda', \delta' \rangle$ of Q , such that for all $x \in \tau$ we have $\lambda'(x) = \lambda(x) \cap O^Q$ and for all $x, xn \in \tau$, we have $\delta'(x, xn) = (\delta(x, xn) \cup \lambda(x)) \cap I^Q$. 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 .

Composition. The *composition* of P and Q , denoted by $P \parallel Q$, exists if $O^P \cap O^Q = \emptyset$, and is defined to be the Moore machine $K = \langle S^K, s^K, I^K, O^K, L^K, R^K \rangle$ where

- $S^K = S^P \times S^Q$,
- $s^K = \langle s^P, s^Q \rangle$,
- $I^K = (I^P \cup I^Q) \setminus (O^P \cup O^Q)$,
- $O^K = O^P \cup O^Q$,

- $L^K(\langle p, q \rangle) = L^P(p) \cup L^Q(q)$ for all $\langle p, q \rangle \in S^K$,
- $R^K(\langle p_1, q_1 \rangle, i, \langle p_2, q_2 \rangle)$ iff $R^P(p_1, (i \cup L^Q(q_1)) \cap I^P, p_2)$ and $R^Q(q_1, (i \cup L^P(p_1)) \cap I^Q, q_2)$ for all $\langle p_1, p_2 \rangle, \langle q_1, q_2 \rangle \in S^K$ and $i \subseteq I^K$.

The branching behavior of a Moore machine is characterized by its set of trace trees. Composition of two Moore machines results in the intersection of the sets of trace trees for the component machines.

Proposition 1. *Consider two Moore machines P and Q such that the composition $P \parallel Q$ exists. 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 .*

Simulation. Consider two Moore machines P and Q such that Q is refinable by P . A binary relation $\Theta \subseteq S^P \times S^Q$ is said to be a *simulation relation* from P to Q if and only if the following three conditions hold:

1. $(s^P, s^Q) \in \Theta$.
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 for all $i \subseteq I^P$ and $\tilde{p} \in S^P$ such that $R^P(p, i, \tilde{p})$, there exists $\tilde{q} \in S^Q$ such that $R^Q(q, (i \cup L^P(p)) \cap I^Q, \tilde{q})$ and $(\tilde{p}, \tilde{q}) \in \Theta$.

If such a relation Θ exists, Q is said to *simulate* P (written as $P \preceq_s Q$) with Θ as the *witnessing* simulation relation. Further, if such Θ exists and $(p, q) \in \Theta$, we say that state q of Q *simulates* state p of P . It is well known that Q simulates P iff in a game of a protagonist playing in Q against an adversary playing in P , the protagonist can match every move of the adversary by moving to a state with the same observation ad infinitum. It is also known that each strategy in such a game corresponds to a trace tree, and consequently, simulation is equivalent to tree containment.

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

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 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 P' . 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 state q'' of P' 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 should 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 1. Let P, Q, P', Q' be Moore machines such that $P \parallel Q$ and $P' \parallel Q'$ exist. 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, and every input of $P' \parallel Q'$ is either an input or output of $P \parallel Q$. Then, we can construct a simulation relation Ω from $P \parallel Q$ to $P' \parallel Q'$.

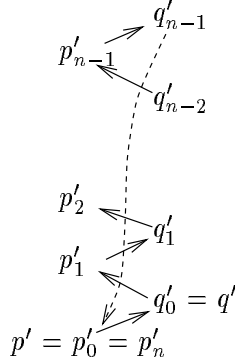


Fig. 4. Figure demonstrating the definition of Ω

Proof. Let $\Omega \subseteq (S^P \times S^Q) \times (S^{P'} \times S^{Q'})$ be defined as follows: $(\langle p, q \rangle, \langle p', q' \rangle) \in \Omega$ iff there exist $p'_0, p'_1, \dots, p'_n \in S^{P'}$ 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 (1) $(\langle p'_i, q \rangle, q'_i) \in \Theta_Q$, and (2) $(\langle p, q'_i \rangle, p'_{i+1}) \in \Theta_P$.

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

1. Since Θ_P and Θ_Q are simulation relations, we have $(\langle s^P, s^{Q'} \rangle, s^{P'}) \in \Theta_P$ and $(\langle s^{P'}, s^Q \rangle, s^{Q'}) \in \Theta_Q$. Let $n = 1$, $p'_0 = p'_1 = s^{P'}$, and $q'_0 = s^{Q'}$. Then, $(\langle s^P, s^Q \rangle, \langle s^{P'}, s^{Q'} \rangle) \in \Omega$.
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 i from 0 to n . Since $p'_n = p'_0 = p'$, we have $L^P(p) \cap O^{P'} = L^{P'}(p')$. Hence, we get $L^P \parallel^Q (\langle p, q \rangle) \cap O^{P'} \parallel^{Q'} = L^{P'} \parallel^{Q'} (\langle p', q' \rangle)$.
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 $i \in \mathcal{P}(I^P \parallel^Q)$. Define $i' = i \cup L^P(p) \cup L^Q(q)$. Clearly, i' includes all inputs of the machines P, Q, P' and 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 inputs of K . We want to show that there exists $(\langle \tilde{p}, \tilde{q} \rangle, \langle \tilde{p}', \tilde{q}' \rangle) \in \Omega$ 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 \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 $R^{P'}(p'_0, i', \tilde{p}'_{k_n})$ and $R^{Q'}(q'_0, i', \tilde{q}'_{k_n})$, 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 the finite nondeterminism property, 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

Fair Moore machines. Consider a Moore machine $P = \langle S^P, s^P, I^P, O^P, L^P, R^P \rangle$.

An *run* of P is a finite or infinite sequence $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} s_3 \dots$ such that $s_0 = s^P$ and $R^P(s_k, i_{k+1}, s_{k+1})$ for all $k \geq 0$. The set of all finite runs of P is denoted by Σ^P . Let $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots \xrightarrow{i_{n-1}} s_{n-1}$. Then the length of \bar{s} , denoted by $|\bar{s}|$, is n , and the k th prefix for $0 \leq k \leq n$, denoted by \bar{s}_k , is $s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots \xrightarrow{i_{k-1}} s_{k-1}$. If \bar{s} is infinite, then $|\bar{s}|$ is defined to be ω . A *fairness constraint* F^P for P is a function that maps every infinite run of P to the binary set $\{fair, unfair\}$. A *fair Moore machine* $\mathcal{P} = \langle P, F^P \rangle$ consists of a Moore machine P and a fairness constraint F^P for P . A *fair run* of \mathcal{P} is either a finite run of P or an infinite run \bar{s} of P such that $F^P(\bar{s}) = fair$. A *fair run tree* of \mathcal{P} is a run tree $\langle \tau, \lambda, \delta \rangle$ of P such that for every path ρ of τ , the run $\Gamma(\rho)$ of P is a fair run of \mathcal{P} . A *fair trace tree* of \mathcal{P} is a trace tree of P that is witnessed by a fair run tree of \mathcal{P} .

We restrict ourselves to fair Moore machines \mathcal{P} that are receptive [Dil89], i.e., in an infinite game of the system \mathcal{P} against the environment, no matter what the environment does the system has a strategy to produce a fair run as the outcome of the game. Formally, a *receptiveness strategy* of \mathcal{P} is a function from $\Sigma^P \times I^P$ to S^P . An infinite run $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots$ of P is an *outcome* of the receptiveness strategy σ if for all $k \geq 1$, we have $s_k = \sigma(\bar{s}_k, i_k)$. The fair Moore machine \mathcal{P} is *receptive* if there is a receptiveness strategy σ such that every outcome \bar{s} of σ is a fair run of \mathcal{P} .

In the following, we consider two fair Moore machines, $\mathcal{P} = \langle P, F^P \rangle$ and $\mathcal{Q} = \langle Q, F^Q \rangle$.

Fair tree containment and composition. 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 *composition* of \mathcal{P} and \mathcal{Q} , denoted by $\mathcal{P} \parallel \mathcal{Q}$, exists if $\mathcal{P} \parallel \mathcal{Q}$ exists, and is defined to be the fair Moore machine $\mathcal{K} = \langle K, F^K \rangle$, where

- $K = \mathcal{P} \parallel \mathcal{Q}$,
- $F^K(\bar{s}) = \text{fair}$ iff both $F^{\mathcal{P}}([\bar{s}]_{\mathcal{P}}) = \text{fair}$ and $F^{\mathcal{Q}}([\bar{s}]_{\mathcal{Q}}) = \text{fair}$, where $[\bar{s}]_{\mathcal{P}}$ is the projection of \bar{s} on \mathcal{P} and $[\bar{s}]_{\mathcal{Q}}$ is the projection of \bar{s} on \mathcal{Q} .

Proposition 3. *Consider two fair Moore machines \mathcal{P} and \mathcal{Q} such that the composition $\mathcal{P} \parallel \mathcal{Q}$ exists. Then T is a fair trace tree of $\mathcal{P} \parallel \mathcal{Q}$ iff (1) $[T]_{\mathcal{P}}$ is a fair trace tree of \mathcal{P} , and (2) $[T]_{\mathcal{Q}}$ is a fair trace tree of \mathcal{Q} .*

Fair simulation. Suppose that \mathcal{Q} is refinable by \mathcal{P} . Intuitively, \mathcal{Q} fairly simulates \mathcal{P} [HKR97] if there is a strategy in the simulation game that matches every fair run of \mathcal{P} with a fair run of \mathcal{Q} . Formally, a *simulation strategy* of \mathcal{Q} with respect to \mathcal{P} is a partial function from $\Sigma^{\mathcal{P}} \times \Sigma^{\mathcal{Q}}$ to $S^{\mathcal{Q}}$. If $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots \xrightarrow{i_{n-1}} s_n \in \Sigma^{\mathcal{P}}$, and $\bar{s}' = s'_0 \xrightarrow{i'_1} s'_1 \xrightarrow{i'_2} s'_2 \xrightarrow{i'_3} \dots \xrightarrow{i'_{m-1}} s'_m \in \Sigma^{\mathcal{Q}}$, then the following three conditions are necessary for $\sigma(\langle \bar{s}, \bar{s}' \rangle)$ to be defined: (1) $n = m + 1$, (2) for all $0 \leq k < n$, we have $L^{\mathcal{P}}(s_k) \cap O^{\mathcal{Q}} = L^{\mathcal{Q}}(s'_k)$, and (3) for all $1 \leq k < m$, we have $(L^{\mathcal{P}}(s_{k-1}) \cup i_k) \cap I^{\mathcal{Q}} = i'_k$. It is required that if $\sigma(\langle \bar{s}, \bar{s}' \rangle) = s'_n$, then $L^{\mathcal{P}}(s_n) \cap O^{\mathcal{Q}} = L^{\mathcal{Q}}(s'_n)$. Given a finite or infinite run $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots$ of \mathcal{P} , the *outcome* $\kappa[\bar{s}]$ of the simulation strategy κ is the finite or infinite run $\bar{s}' = s'_0 \xrightarrow{i'_1} s'_1 \xrightarrow{i'_2} s'_2 \xrightarrow{i'_3} \dots$ of \mathcal{Q} such that (1) $|\kappa[\bar{s}]| = |\bar{s}|$, and (2) for all $k \geq 1$, we have $s'_k = \kappa(\bar{s}_{k+1}, \bar{s}_k)$. A binary relation $\Theta \subseteq S^{\mathcal{P}} \times S^{\mathcal{Q}}$ is a *fair simulation relation* of \mathcal{P} by \mathcal{Q} if the following three conditions hold:

1. $(s^{\mathcal{P}}, s^{\mathcal{Q}}) \in \Theta$.
2. For all $(p, q) \in \Theta$, we have $L^{\mathcal{P}}(p) \cap O^{\mathcal{Q}} = L^{\mathcal{Q}}(q)$.
3. There exists a simulation strategy κ of \mathcal{Q} with respect to \mathcal{P} such that, if $(p, q) \in \Theta$ and $\bar{s} = s_0 \xrightarrow{i_1} s_1 \xrightarrow{i_2} s_2 \xrightarrow{i_3} \dots$ is a fair run of \mathcal{P} , then the outcome $\kappa[\bar{s}] = s'_0 \xrightarrow{i'_1} s'_1 \xrightarrow{i'_2} s'_2 \xrightarrow{i'_3} \dots$ is a fair run of \mathcal{Q} and $\kappa[\bar{s}] \Theta$ -matches \bar{s} (that is, $(s_k, s'_k) \in \Theta$ for all $0 \leq k < |\bar{s}|$). We say that κ is a *witness* to the fair simulation Θ .

If such a relation Θ exists, \mathcal{Q} is said to *fairly simulate* \mathcal{P} , written $\mathcal{P} \preceq_s^F \mathcal{Q}$. We will state some properties of fair simulation. First, since all finite runs are fair by definition, we have the following proposition.

Proposition 4. *Consider two fair Moore machines $\mathcal{P} = \langle P, F^P \rangle$ and $\mathcal{Q} = \langle Q, F^Q \rangle$. If $\mathcal{P} \preceq_s^F \mathcal{Q}$, then $P \preceq_s Q$.*

The proof of Proposition 4 depends on the receptiveness of \mathcal{Q} . Analogous to the equivalence between simulation and tree containment, fair simulation can be proved to be equivalent to fair tree containment [HKR97].

Proposition 5. *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} .*

5 Assume-Guarantee Rule for Fair Simulation

Let $\text{Safe}(\mathcal{P})$ be the fair Moore machine obtained by replacing the fairness constraint of \mathcal{P} with the trivial fairness constraint that maps every infinite run to *fair*. We now present the assume-guarantee proof rule for fair simulation. The same rule was proved for fair trace containment in [AH96], with an essentially similar proof (we just use trace trees instead of traces to get the proof below).

Theorem 2. *Let $\mathcal{P}, \mathcal{Q}, \mathcal{P}'$ and \mathcal{Q}' be Moore machines such that $\mathcal{P} \parallel \mathcal{Q}$ and $\mathcal{P}' \parallel \mathcal{Q}'$ exist. Suppose that $\mathcal{P} \parallel \text{Safe}(\mathcal{Q}') \preceq_s^F \mathcal{P}'$, $\mathcal{P}' \parallel \mathcal{Q} \preceq_s^F \mathcal{Q}'$, and every input of $\mathcal{P}' \parallel \mathcal{Q}'$ is either an input or output of $\mathcal{P} \parallel \mathcal{Q}$. Then, $\mathcal{P} \parallel \mathcal{Q} \preceq_s^F \mathcal{P}' \parallel \mathcal{Q}'$.*

Proof. Let $\mathcal{P} = \langle P, F^P \rangle$, $\mathcal{Q} = \langle Q, F^Q \rangle$, $\mathcal{P}' = \langle P', F^{P'} \rangle$, and $\mathcal{Q}' = \langle Q', F^{Q'} \rangle$. From Proposition 4, we know that $P \parallel Q' \preceq_s P'$ and $P' \parallel Q \preceq_s Q'$. Consequently, by Theorem 1 we know that $P \parallel Q \preceq_s P' \parallel Q'$. Therefore, from Proposition 2, we get 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 T is a fair trace tree of $\mathcal{P} \parallel \mathcal{Q}$. From Proposition 3, T is a fair trace tree of both \mathcal{P} and \mathcal{Q} . Further, 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 3, 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 5) that T is a fair trace tree of \mathcal{P}' . Therefore, from Proposition 3, since T is a fair trace tree of both \mathcal{P}' and \mathcal{Q}' , it is a fair trace tree of $\mathcal{P}' \parallel \mathcal{Q}$. Since $\mathcal{P}' \parallel \mathcal{Q} \preceq_s^F \mathcal{Q}'$, T is a fair trace tree of \mathcal{Q}' . Finally from Proposition 3, since T is a fair trace tree of both \mathcal{P}' and \mathcal{Q}' , it is a fair trace tree of $\mathcal{P}' \parallel \mathcal{Q}'$. \square

References

- [AH96] R. Alur and T.A. Henzinger. Reactive modules. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 207–218. IEEE Computer Society Press, 1996.
- [AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991.
- [AL95] M. Abadi and L. Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, 1995.
- [CLM89] E.M. Clarke, D.E. Long, and K.L. McMillan. Compositional model checking. In *Proceedings of the 4th Annual Symposium on Logic in Computer Science*, pages 353–362. IEEE Computer Society Press, 1989.
- [Dil89] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits*. The MIT Press, 1989.
- [GL94] O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994.
- [HKR97] T.A. Henzinger, O. Kupferman, and S. K. Rajamani. Fair simulation. In *CONCUR 97: Theories of Concurrency*, Lecture Notes in Computer Science 1243, pages 273–287. Springer-Verlag, July 1997.

- [Kur94] R.P. Kurshan. *Computer-aided Verification of Coordinating Processes*. Princeton University Press, 1994.
- [Lyn96] N.A. Lynch. *Distributed Algorithms*. Morgan-Kaufmann, 1996.
- [McM97] K.L. McMillan. A compositional rule for hardware design refinement. In *CAV 97: Computer-Aided Verification*, Lecture Notes in Computer Science 1254, pages 24–35. Springer-Verlag, 1997.
- [Mil71] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 481–489. The British Computer Society, 1971.
- [Sta85] E.W. Stark. A proof technique for rely/guarantee properties. In *Proceedings of the 5th Conference on Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science 206, pages 369–391. Springer-Verlag, 1985.