Foundations for Circular Compositional Reasoning

Mahesh Viswanathan[1] and Ramesh Viswanathan[2]

[1] DIMACS & Telcordia Technologies,
Piscataway NJ 08854, USA
maheshv@dimacs.rutgers.edu
[2] Bell Laboratories, Holmdel NJ 07733, USA
rv@research.bell-labs.com

Abstract. Compositional proofs about systems of many components require circular reasoning principles in which properties of other components need to be assumed in proving the properties of each individual component. A number of such circular assume-guarantee rules have been proposed for different concurrency models and different forms of property specifications. In this paper, we provide a framework that unifies and extends these results. We define an assume-guarantee semantics for properties expressible as least or greatest fixed points, and a circular compositional rule that is sound with respect to this semantics. We demonstrate the utility of this general rule by applying it to trace semantics with linear temporal logic specifications, and trace tree semantics with automata refinement specifications. For traces, we derive a new assume-guarantee rule for the "until" and "weakly until" properties of linear temporal logic and show that previously proposed assume-guarantee rules can be seen as special instances of our rule. For trace trees, we derive a rule for parallel composition of Moore machines, and show that the rule of [7] is a special instance thus yielding an alternate proof of the results in [7].

1 Introduction

Program verification is concerned with determining whether a formal model of a system satisfies certain correctness properties. The most popular algorithmic technique, model checking, systematically steps through the global states of the system while checking various properties at each stage. However, such a method runs into the well-known problem of *state-space explosion* which severely limits the size of analyzable systems.

The classical solution to this problem is to verify systems compositionally or hierarchically in which suitable properties of individual components of a system are verified in isolation, and these "local" properties are then combined to prove the correctness of the system as a whole. For a system $P = P_1||P_2$, a rule that supports such compositional reasoning might take the form that if P_1 satisfies property φ_1 and P_2 satisfies φ_2, then P satisfies $\varphi_1 \wedge \varphi_2$. While this form of reasoning is sound for various specification languages and concurrency models,

it is often not helpful. This is because, in many cases, although the global system P may satisfy the property $\varphi_1 \wedge \varphi_2$, the decomposed proof obligations of P_1 and P_2 satisfying φ_1 and φ_2 may not necessarily hold. Typically, a component P_1 does not satisfy φ_1 "unconditionally" (*i.e.*, when composed with arbitrary components) but only when composed with another component that behaves like P_2.

In the assumption-guarantee paradigm [15] (also referred to as rely-guarantee and assumption-commitment in the literature), each component of a system is specified in terms of assumptions it makes about its environment (or other components), and properties it guarantees about its behavior, provided the assumptions hold. Using $\varphi \triangleright \psi$ to syntactically denote the property that under the assumptions φ, the property ψ is guaranteed, a *circular* compositional rule that one would like for the system $P = P_1||P_2$ would be that if P_1 satisfies the property $\varphi_2 \triangleright \varphi_1$ and P_2 satisfies the property $\varphi_1 \triangleright \varphi_2$ then P satisfies $\varphi_1 \wedge \varphi_2$. However, because of the circularity of each component making assumptions about the other component's yet-to-be proven guarantees, such rules are hard to construct and are in fact sound only for some special classes of properties. For example, such a rule is clearly not propositionally valid if we interpret assumption-guarantee to be propositional implication. That such a circular argument is indeed sound under some conditions, was first observed by Misra and Chandy [13], and later formalized by Abadi and Lamport [1, 2]. Subsequently, circular compositional rules have been proposed for a variety of computational models. Traces and safety properties are considered in [13, 8, 1, 2, 14, 10, 6, 9, 3, 4, 11, 5], and more recently, McMillan [12] has obtained a circular reasoning principle for certain liveness properties. This approach for traces has been extended to trace trees in which specifications are given via some form of automata-refinement [3, 4, 7]. Henzinger, Qadeer, Rajamani, and Tasiran [7] present such an assumption/guarantee rule for Moore machines with synchronous parallel composition. Liveness requirements can be specified in terms of additional Streett fairness conditions imposed on Moore and Mealy machines.

In this paper, we present a common unifying framework and reasoning rule which encompasses previously proposed circular composition rules and which can be applied to derive circular composition rules for more expressive classes of properties than have previously been considered. Our starting point is the well known observation that many important properties, including safety and liveness properties, can be formulated as least or greatest fixed points of appropriate functions. We therefore define a semantics for assume-guarantee specifications in which the assumptions and guarantees are least or greatest fixed points. We then formulate a rule for composing such local assume-guarantee specifications to derive a global assume-guarantee specification that is sound with respect to this semantics. Our assume-guarantee semantics and rule are formulated and proved sound in a general setting which requires limited assumptions about the semantics of processes or the form of specifications We next show how this general rule can be fruitfully applied to some specific contexts. First, we consider the setting where semantics for processes are defined as traces and specifications as

linear temporal logic formulas. Here, we derive, from our general framework, an assume-guarantee rule for the "weakly until" temporal logic operator, and show that it generalizes the most powerful rule previously known in this setting [12]. The second setting we consider is that of Moore machines interpreted in a semantic space consisting of labelled trace trees. We derive an assume-guarantee rule for synchronous parallel composition of Moore machines, and show that this rule subsumes the strongest known circular rule for Moore machines, that of [7]. Together, the two settings demonstrate that our general theorem unifies and can be applied to disparate process semantics and disparate specification formulations.

The rest of the paper is organized as follows. In Section 2, we present our semantics for assumption/guarantee specifications for fixed point properties, and general rules for composing such specifications. Section 3 applies these rules to traces and linear temporal logic, and Section 4 develops the results for Moore machines. We make concluding remarks in Section 5.

2 Assume-Guarantee Rule for Fixed Points

2.1 Preliminaries and Notation

Let $\mathcal{C}$ be a set; intuitively, $\mathcal{C}$ represents the semantic space of all valid computations (such as traces or trees). Thus, the semantics of a program or automaton P, written $[\![P]\!]$, will be a subset of $\mathcal{C}$. However, the framework and rule we present in this section will require no assumptions on the structure of computations and we therefore leave the elements of $\mathcal{C}$ uninterpreted. We use $\sigma, \sigma_1, \ldots$ to range over computations, *i.e.*, $\sigma, \sigma_1, \ldots \in \mathcal{C}$ and a specification $S \subseteq \mathcal{C}$. A computation σ satisfies a specification S, written $\sigma \models S$, if and only if $\sigma \in S$, and a program P satisfies a specification S, $P \models S$, if and only if $\forall \sigma \in [\![P]\!].\ \sigma \models S$.

We are interested in properties that are recursive or fixed point specifications. Writing $\mathcal{P}(\mathcal{C})$ to denote the power set of $\mathcal{C}$, let $F : \mathcal{P}(\mathcal{C}) \to \mathcal{P}(\mathcal{C})$ be a monotonic operator with respect to the subset ordering, $\subseteq$, on $\mathcal{P}(\mathcal{C})$. By the Tarski-Knaster theorem [17], F has both greatest and least fixed points. We will use $\nu(F)$ to denote the greatest fixed point of F, $\mu(F)$ for its least fixed point, and meta-variables $\rho, \rho_1, \ldots$ to stand for either μ or ν so that $\rho(F)$ will denote one of the two fixed points of F. The Tarski-Knaster construction approximates the fixed points through repeated iterations of F whose limit yields the desired fixed point. Let $\omega = \{0, 1, 2, \ldots\}$ be the set of all natural numbers; the k'th approximation of the fixed point $\rho(F)$, which we denote $[\rho(F)]^k$, is defined by induction:

$$\begin{array}{llll} [\mu(F)]^0 = \emptyset & [\mu(F)]^{k+1} = F([\mu(F)]^k) & [\mu(F)]^\omega = \bigcup_{k \in \omega} [\mu(F)]^k \\ [\nu(F)]^0 = \mathcal{C} & [\nu(F)]^{k+1} = F([\nu(F)]^k) & [\nu(F)]^\omega = \bigcap_{k \in \omega} [\nu(F)]^k \end{array}$$

Depending on the cardinality of S, this process of constructing progressively better approximations needs to be continued for transfinite ordinals before it stabilizes to the fixed point. However, all applications considered in this paper

converge to their fixed points within ordinal ω ;we say that a fixed point specification $\rho(F)$ (where ρ is μ or ν) is ω-convergent if $F([\rho(F)]^{\omega}) = [\rho(F)]^{\omega}$. For example, if F is ω-continuous, *i.e.*, for any ω-increasing chain $\{S_i \mid i \in \omega\}$ with $S_i \subseteq S_{i+1}$ for all i, we have that $F(\cup_i S_i) = \cup_i F(S_i)$, then $\mu(F)$ is ω-convergent. Similarly, if F is ω-cocontinuous, then $\nu(F)$ is ω-convergent. Thus, ω-continuity and ω-cocontinuity provide sufficient checks for ω-convergence of $\mu(F)$ and $\nu(F)$ respectively.

2.2 Semantics of Assume-Guarantee Specifications

We are now ready to define the semantics for assume-guarantee specifications, which we syntactically denote by $\rho(A) \triangleright \rho'(G)$, whose informal reading is that the guarantee specification $\rho'(G)$ is satisfied whenever the assumption specification $\rho(A)$ is satisfied. Classically, in the context of safety properties, it has been interpreted to require the guarantee to be satisfied upto time instant $k+1$ whenever the assumption is satisfied upto time instant k. Our definition is motivated by this classical notion with the key insight being that the k'th approximation provides the suitable analogue of "upto time instant k" . Further, to allow the weakest condition necessary for the guarantee to be satisfied when it is a least fixed point, we generalize the condition on the guarantee specification requiring it to be satisfied at any future approximation (rather than the immediately next one).

Definition 1 (Assume-Guarantee Semantics).

$$\sigma \models \rho(A) \triangleright \rho'(G) \quad \textit{iff} \quad \forall k \geq 0.\ \sigma \models [\rho(A)]^k \ \Rightarrow\ \exists k' > k.\ \sigma \models [\rho'(G)]^{k'}$$

An immediate consequence of our assume-guarantee semantics is that it generalizes propositional implication.

Proposition 1. *Suppose that the fixed point specification $\rho'(G)$ is ω-convergent. Then the following rule is sound*

$$\frac{\sigma \models \rho(A) \triangleright \rho'(G)}{\sigma \models \rho(A) \ \Rightarrow\ \sigma \models \rho'(G)} \qquad (Imp\ \triangleright)$$

When the consequent of an assume-guarantee specification is a greatest fixed point, then our general definition reduces to the more familiar classical notion. When the consequent is a least fixed point, then our semantics places no stronger requirements than that the least fixed point be satisfied.

Proposition 2. *1.* $\sigma \models \rho(A) \triangleright \nu(G) \quad \textit{iff} \quad \forall k \geq 0.\ \sigma \models [\rho(A)]^k \ \Rightarrow\ \sigma \models [\nu(G)]^{k+1}$
2. $\sigma \models \rho(A) \triangleright \mu(G) \quad \textit{iff} \quad \forall k \geq 0.\ (\sigma \models [\rho(A)]^k \ \Rightarrow\ \exists k' \geq 0.\ \sigma \models [\mu(G)]^{k'})$

2.3 Composing Assume-Guarantee Specifications

In this section, we describe inference rules for composing assume-guarantee specifications. We first present general rules that are applicable to arbitrary assume-guarantee specifications without making any assumptions about the form of fixed points. We next specialize these general rules to derive more powerful rules that are applicable when the assumption is a greatest fixed point.

General Fixed Points The general semantic rule prescribes the conditions under which it is sound to compose two known local behaviors, $\sigma_1 \models \rho(A_1) \triangleright \rho'(G_1)$ and $\sigma_2 \models \rho(A_1) \triangleright \rho'(G_2)$ to derive the behavior of the global system $\sigma \models \rho(A) \triangleright \rho'(G)$ (in typical applications of the rule, σ will be the parallel composition of σ_1 and σ_2). The conditions can be intuitively read as follows: *(1a), (1b)* If the global assumption is satisfied currently and one of the local guarantees is satisfied eventually, then the other local assumption is satisfied eventually. *(2)* If the local guarantees hold then the global guarantee holds eventually. *(3)* If the global assumption is satisfied then one of the local guarantees holds eventually. Conditions *(1a), (1b)* allow each local guarantee to be used circularly in satisfying the other's local assumptions, and Condition *(3)* breaks this circularity to ensure soundness.

Theorem 1 (Semantic Rule for General Fixed Points). *The following rule is sound:*

$$\frac{\begin{array}{l}\sigma_1 \models \rho(A_1) \triangleright \rho'(G_1) \\ \sigma_2 \models \rho(A_2) \triangleright \rho'(G_2) \\ (1a)\ \forall k, k' \geq k.\ \sigma \models [\rho(A)]^k \wedge \sigma_2 \models [\rho'(G_2)]^{k'} \Rightarrow \exists k'' \geq k.\ \sigma_1 \models [\rho(A_1)]^{k''} \\ (1b)\ \forall k, k' \geq k.\ \sigma \models [\rho(A)]^k \wedge \sigma_1 \models [\rho'(G_1)]^{k'} \Rightarrow \exists k'' \geq k.\ \sigma_2 \models [\rho(A_2)]^{k''} \\ (2)\ \ \forall k.\ \sigma_1 \models [\rho'(G_1)]^k \wedge \sigma_2 \models [\rho'(G_2)]^k \Rightarrow \exists k' \geq k.\ \sigma \models [\rho'(G)]^{k'} \\ (3)\ \ \forall k.\ \sigma \models [\rho(A)]^k \Rightarrow \exists k' \geq k.\ \sigma_1 \models [\rho'(G_1)]^{k'} \vee \sigma_2 \models [\rho'(G_2)]^{k'}\end{array}}{\sigma \models \rho(A) \triangleright \rho'(G)}\ (Comp\ \ \rho \triangleright \rho')$$

Observe that, the Rule $(Comp\ \ \rho \triangleright \rho')$ requires that the assumptions in the assume-guarantee specifications of σ_1, σ_2, and σ be the same form of fixed point (ρ) and similarly for the guarantees. However, certain variations are sound — we formally allow them using a subsumption rule, for which we first define a subtyping relation.

Definition 2. *For monotonic operators $F, G : \mathcal{P}(\mathcal{C}) \rightarrow \mathcal{P}(\mathcal{C})$, we say that $F \preccurlyeq G$ iff for all $S \subseteq \mathcal{C}$ we have that $F(S) \subseteq G(S)$. We say that a fixed point specification $\rho'(F')$ is a subtype of another fixed point specification $\rho(F)$, written $\rho'(F') <: \rho(F)$, if and only if the following conditions hold:*

- *We have that either $\rho' = \mu$ and $\rho = \nu$ or $\rho' = \rho$*
- *$F' \preccurlyeq F$.*

Theorem 2 (Subsumption). *The following rule is sound:*

$$\frac{\sigma \models \rho_1(A) \triangleright \rho_2(G) \qquad {\rho'}_1(A') <: \rho_1(A) \qquad \rho_2(G) <: {\rho'}_2(G')}{\sigma \models \rho'_1(A') \triangleright \rho'_2(G')} \qquad (Sub\ \triangleright)$$

Now, given two local assumption-guarantee specifications in which the forms of the fixed points in the assumptions (or guarantees) do not match, we can use Rule $(Sub\ \triangleright)$ on one or both of them to transform to a form in which the corresponding fixed points match and Rule $(Comp\ \ \rho \triangleright \rho')$ is applicable. Having derived the global behavior, a further use of subsumption allows one to change the form of the fixed point in the assumption or guarantee (but only in a manner consistent with subtyping).

Greatest Fixed Point Assumptions The true payoff of the general inference rules established in Section 2.3 occurs when the assumption specifications are greatest fixed points. In particular, we can support truly circular reasoning without having to explicitly break the circularity (through establishing Condition *(3)*). To this end, we first prove the following lemma.

Lemma 1. *Suppose that* $\sigma_i \models \nu(A_i) \triangleright \rho(G_i)$ *for* $i = 1, 2$ *and that*

$$\forall k, k' \geq k.\ \sigma \models [\nu(A)]^k \wedge \sigma_1 \models [\rho(G_1)]^{k'} \wedge \sigma_2 \models [\rho(G_2)]^{k'} \Rightarrow \sigma_1 \models [\nu(A_1)]^k \wedge \sigma_2 \models [\nu(A_2)]^k$$

Then we have that for any k*,*

$$\sigma \models [\nu(A)]^k \Rightarrow \exists k' \geq k.\ \sigma_1 \models [\rho(G_1)]^{k'} \wedge \sigma_2 \models [\rho(G_2)]^{k'}$$

Lemma 1 is established by induction on k, and the induction step makes crucial use of the fact that in our semantics of assumption guarantee-semantics we require the guarantee to hold at a *strictly greater* approximation. It therefore illustrates the precise origin of the condition $k' > k$ (rather than $k' \geq k$) in Definition 1.

Using Lemma 1, we can instantiate Rule $(Comp\ \ \rho \triangleright \rho')$ to derive a stronger rule, $(Comp\ \ \nu \triangleright \nu)$, for greatest fixed points. First, the stronger rule does not require Condition *(3)* to be established. Second, in establishing the local assumptions (Conditions *(1a)* and *(1b)*), we are now allowed to assume *both* local guarantees and that they hold *currently* (rather than eventually). Rule $(Conj\ \ \nu)$ is a further application of $(Comp\ \ \nu \triangleright \nu)$, which illustrates the circular reasoning supported by Rule $(Comp\ \ \nu \triangleright \nu)$ more clearly.

Corollary 1 (Semantic Rule for Greatest Fixed Points). *The following rules are sound:*

$$\frac{\begin{array}{c}\sigma_1 \models \nu(A_1) \triangleright \nu(G_1) \\ \sigma_2 \models \nu(A_2) \triangleright \nu(G_2) \\ (1)\ \forall k.\ \sigma \models [\nu(A)]^k \wedge \sigma_1 \models [\nu(G_1)]^k \wedge \sigma_2 \models [\nu(G_2)]^k \Rightarrow \\ \qquad \sigma_1 \models [\nu(A_1)]^k \wedge \sigma_2 \models [\nu(A_2)]^k \\ (2)\ \forall k. \sigma_1 \models [\nu(G_1)]^k \wedge \sigma_2 \models [\nu(G_2)]^k \Rightarrow \sigma \models [\nu(G)]^k\end{array}}{\sigma \models \nu(A) \triangleright \nu(G)} \qquad (Comp\ \ \nu \triangleright \nu)$$

$$\frac{\sigma \models \nu(F_1) \triangleright \rho(F_2) \qquad \sigma \models \nu(F_2) \triangleright \rho(F_1)}{\sigma \models \rho(F_1) \wedge \sigma \models \rho(F_2)} \qquad (Conj\ \ \nu)$$

Rule $(Conj\ \ \nu)$ illustrates the importance of the distinction between $\triangleright$ and propositional implication. Had we interpreted $\triangleright$ to be propositional implication, the rule would not have been sound.

In many applications, specifications are most easily described through mutual recursion and we conclude this section by briefly addressing this. Suppose that $F_0, \ldots, F_{n-1} : \mathcal{P}(\mathcal{C})^n \to \mathcal{P}(\mathcal{C})$ are monotonic operators. We write $\rho_i(F_0, \ldots, F_{n-1})$ to denote the i'th solution, where as usual, ρ can be μ or ν. The defining property of these solutions is that

$$\rho_i(F_0, \ldots, F_{n-1}) = F_i(\rho_0(F_0, \ldots, F_{n-1}), \ldots, \rho_{n-1}(F_0, \ldots, F_{n-1}))$$

We define the k'th approximation of such mutually dependent fixed point specifications as follows:

$$\begin{aligned} [\rho_i(F_0, \ldots, F_{n-1})]^0 &= \begin{cases} \emptyset \text{ if } \rho = \mu \\ \mathcal{C} \text{ if } \rho = \nu \end{cases} \\ [\rho_i(F_0, \ldots, F_{n-1})]^{k+1} &= F_i([\rho_1(F_0, \ldots, F_{n-1})]^k, \ldots, [\rho_n(F_0, \ldots, F_{n-1})]^k) \end{aligned}$$

Having defined the k'th approximations, assume-guarantee specifications involving mutually dependent fixed points are interpreted as in Definition 1. The semantic rule $(Comp\ \ \rho \triangleright \rho')$ is then sound. The semantic rule $(Comp\ \ \nu \triangleright \nu)$ is sound when both assumptions and guarantees are of the form ν_i.

3 Assumption-Guarantee for Traces Semantics

In this section, we consider the framework of linear time temporal logic and traces. Using our general assume-guarantee semantics, we obtain the definition of assume-guarantee specifications for traces, and then derive composition rules for linear time temporal logic. In particular, we derive a circular reasoning principle for "weak until" specifications which are a generalization of properties defined using the "always" modality. Finally, we show that the reasoning principle proposed in [12] is a special case of our rule for until properties.

We take the semantic space of computations $\mathcal{C}$ to be the set of functions $\sigma : \omega \to \mathcal{P}(\text{Prop})$, where Prop is a set of propositions. In other words, computations are viewed as infinite sequences of subsets of propositions. For a computation σ, we use σ_i to denote its i'th element $\sigma(i)$. The suffix of a computation starting at the i'th position will be denoted as $\sigma|_i$; so $\sigma|_i(k) = \sigma(i+k)$. Following [16, 1, 2, 9, 11, 12], we do not define a syntactic computational model such as a process or automaton; instead a program will simply be a collection of computations. It is often convenient to express such a process as a linear temporal logic property.

A formula in temporal logic is described by the following BNF grammar.

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \circ \varphi \mid \varphi \ \mathrm{U} \ \varphi \mid \varphi \ \mathrm{U_W} \ \varphi$$

where $p \in \text{Prop}$ is a proposition. Apart from classical conjunction and disjunction, the logic includes the modal connectives "next" ($\circ$), "until" (U), and "weak-until" (U_W). Formally, we define a set of computations $[\![\varphi]\!]$ such that $\sigma \models \varphi$ if and only if $\sigma \in [\![\varphi]\!]$.

$$[\![p]\!] = \{\sigma \mid p \in \sigma_0\} \qquad [\![\neg p]\!] = \{\sigma \mid p \notin \sigma_0\}$$
$$[\![\varphi \wedge \psi]\!] = [\![\varphi]\!] \cap [\![\psi]\!] \qquad [\![\varphi \vee \psi]\!] = [\![\varphi]\!] \cup [\![\psi]\!]$$
$$[\![\circ \varphi]\!] = \{\sigma \mid \sigma|_1 \in [\![\varphi]\!]\}$$
$$[\![\varphi \ \text{U} \ \psi]\!] = \mu(F[[\![\varphi]\!], [\![\psi]\!]]) \qquad [\![\varphi \ \text{U}_\text{W} \ \psi]\!] = \nu(F[[\![\varphi]\!], [\![\psi]\!]])$$

where $F[S_1, S_2]$ is the following monotonic operator:

$$F[S_1, S_2](S) = \{\sigma \mid \sigma \in S_2 \text{ or } (\sigma \in S_1 \text{ and } \sigma|_1 \in S)\}$$

As usual, $\Diamond \varphi$ (eventually φ) is equivalent to (true U φ), and $\Box \varphi$ (always φ) is equivalent to (φ U_W false).

Example 1. [**Assume-Guarantee Semantics for Safety**] Consider the property $\Box p$, where p is a proposition. As seen above, $\Box p = \nu(F)$ for the operator,

$$F(S) = \{\sigma \mid p \in \sigma_0 \text{ and } \sigma|_1 \in S\}$$

Now $\sigma \in [\nu(F)]^k$ if and only if $p \in \sigma_i$ for every $0 \leq i \leq k-1$. Hence for an assume-guarantee specification $S = \Box p \triangleright \Box q$ (where q is also a proposition), a computation σ satisfies S if and only if $q \in \sigma_0$ and if $p \in \sigma_i$, for $i \leq k-1$, then $q \in \sigma_j$, for $j \leq k$.

The above discussion can be generalized to any safety property. Consider formulas $\Box \varphi$ and $\Box \psi$, where φ and ψ are past time properties. A computation $\sigma \models \Box \varphi \triangleright \Box \psi$ if and only if σ_0 satisfies ψ and if any prefix $\sigma_0.\sigma_1 \ldots \sigma_{k-1}$ of σ satisfies φ then $\sigma_0.\sigma_1 \ldots \sigma_{k-1}.\sigma_k$ satisfies ψ. Thus, for safety properties our definition for assume-guarantee coincides with the classical definition in [1].

We apply Rule (*Comp* $\nu \triangleright \nu$) to derive a rule for specifications whose outermost connective is U_W. Theorem 3, stated below, highlights the pay-off when the general rule (*Comp* $\nu \triangleright \nu$) is instantiated for a specific computational framework: premises about computations and arbitary fixed point approximations are instead replaced by logical tautologies in Rule (*Comp* $\text{U}_\text{W} \triangleright \text{U}_\text{W}$).

Theorem 3. *The following rule is sound.* [1]

$$\frac{\begin{array}{ll} \sigma \models (\varphi_1^1 \ \text{U}_\text{W} \ \varphi_2^1) \triangleright (\psi_1^1 \ \text{U}_\text{W} \ \psi_2^1) & \sigma \models (\varphi_1^2 \ \text{U}_\text{W} \ \varphi_2^2) \triangleright (\psi_1^2 \ \text{U}_\text{W} \ \psi_2^2) \\ (1a) \ (\varphi_1 \wedge \psi_1^1 \wedge \psi_1^2) \models (\varphi_1^1 \wedge \varphi_1^2) & (1b) \ (\varphi_2 \wedge \psi_2^1 \wedge \psi_2^2) \models (\varphi_2^1 \wedge \varphi_2^2) \\ (2a) \ (\psi_1^1 \wedge \psi_1^2) \models \psi_1 & (2b) \ (\psi_2^1 \wedge \psi_2^2) \models \psi_2 \end{array}}{\sigma \models (\varphi_1 \ \text{U}_\text{W} \ \varphi_2) \triangleright (\psi_1 \ \text{U}_\text{W} \ \psi_2)} \quad (\textit{Comp} \ \text{U}_\text{W} \triangleright \text{U}_\text{W})$$

[1] Recall, $\varphi \models \psi$ iff for every σ, $\sigma \models \varphi$ implies $\sigma \models \psi$.

McMillan's rule for liveness. McMillan [12] presents a rule for reasoning about $\Box\,\varphi$ properties. His rule says that if a computation satisfies a collection of specially constructed until properties built up from formulas in some set P, then the computation satisfies $\Box\,\varphi$ for every $\varphi \in P$. We make the key observation that any computation satisfying the until properties in McMillan's theorem also satisfies an assumption-guarantee specification of $\Box\,\varphi$ formulas. More precisely, we show that if $\sigma \models \neg(\varphi\ \mathrm{U}\ \psi)$ then $\sigma \models \Box\,\varphi \triangleright \Box\,(\neg\psi)$. Since properties of the form $\Box\,\varphi$ are a special class of weak until specifications, the soundness of McMillan's rule then follows by repeated applications of Rule $(\mathit{Comp}\ \mathrm{U_W} \triangleright \mathrm{U_W})$.

Theorem 4. *Given a set P of formulas, a well founded order $\prec$ on P, and for all $\varphi \in P$, sets $\Theta_\varphi \subseteq \Delta_\varphi \subseteq P$, such that $\psi \in \Theta_\varphi$ implies $\psi \prec \varphi$. If $\sigma \models \neg(\Delta_\varphi\ \mathrm{U}\ (\Theta_\varphi \wedge \neg\varphi))$, for all $\varphi \in P$, then $\sigma \models \Box\,\varphi$, for all $\varphi \in P$.*

4 Assume-Guarantee for Tree Semantics

In this section, we consider a semantic setting of trace trees with Moore machines defining our programming model. We derive a composition rule, (*Comp Moore*), for Moore machines with synchronous parallel composition and show how the rule for simulation preorder in [7] is a special instance of (*Comp Moore*). For details on the advantages of trace-trees and using automata-refinement specifications, we refer the reader to [7]. Finally, we note that while our results are developed for Moore machine, they apply as well to other non-blocking, finitely non-deterministic, receptive models such as Reactive Modules [4], where parallel composition corresponds to logical conjunction.

4.1 Moore Machines and their Semantics

Preliminaries. A (rooted) tree is a set $\tau \subseteq \omega^*$ such that if $xn \in \tau$, for $x \in \omega^*$ and $n \in \omega$, then $x \in \tau$ and $xm \in \tau$ for all $0 \leq m \leq n$. The elements of τ represent nodes: the empty string ϵ is the root, and for each node x, the nodes of the form xn, for $n \in \omega$, are the children of x. The edges of the tree are pairs $\langle x, xn\rangle$, where $x, xn \in \tau$. The number of children of a node x is degree of the node, and it is denoted by $deg(x)$. A tree τ is finitely branching if for every $x \in \tau$, $deg(x)$ is finite. A tree τ is finite if the set τ is finite; otherwise τ is said to be infinite. The set of finite branching, infinite trees is denoted by $\mathbf{T}^\omega_f$. A node $x \in \tau$ is said to be at depth $|x|$, where $|x|$ denotes the length of string x. For a tree τ, the subtree rooted at $x \in \tau$, is the tree $\tau|_x = \{i \mid xi \in \tau\}$.

Given sets A and B, an $\langle A, B\rangle$-labeled tree is a triple $\hat{\tau} = \langle \tau, \lambda, \delta\rangle$, where τ is a tree, $\lambda : \tau \to A$ is a labeling function that that maps each node of the tree to an element of A, and $\delta : \tau \times \omega \to B$ is a function that labels each edge $\langle x, xn\rangle$ in τ with $\delta(x, n) \in B$. Given a labeled tree $\hat{\tau}$, the labeled tree rooted at x is $\hat{\tau}|_x = \langle \tau|_x, \lambda', \delta'\rangle$, where $\lambda'(y) = \lambda(xy)$, and $\delta'(y, n) = \delta(xy, n)$.

Definition 3. [Moore Machines] *A Moore machine is a tuple $P = \langle S, s_0, I, O, L, R\rangle$ 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 disjoint from I, $L : S \to \mathcal{P}(O)$ is a function that labels each state with the set of output propositions true in that state, and $R \subseteq S \times \mathcal{P}(I) \times S$ is the transition relation. We write $R(s,i,t)$ instead of $(s,i,t) \in R$.

In what follows we only consider Moore machines that are *non-blocking* and *finitely nondeterministic*. We denote the *parallel composition* of machines P and Q by $P||Q$, and write $P \preceq_s Q$ to denote that a machine Q *simulates* machine P. The definitions of non-blocking, finitely nondeterminitic, parallel composition, and simulation are as in [7].

Semantic Space. We will define the semantics of a Moore machine in terms of labeled, finitely branching infinite trees. A run tree of a machine P is a $\langle S, \mathcal{P}(I)\rangle$-labeled tree $\langle \tau, \lambda, \delta\rangle$, such that $\tau \in \mathbf{T}^{\omega}_{f}$, $\lambda(\epsilon) = s_0$, and for every edge $\langle x, xn\rangle$ of τ we have $R(\lambda(x), \delta(x,n), \lambda(xn))$. Note that since our machines are non-blocking, every machine has at least one run tree. A trace tree $\langle \hat{\tau}, \langle O, I\rangle\rangle \in \mathcal{C}$ of P is a $\langle \mathcal{P}(O), \mathcal{P}(I)\rangle$-labeled tree $\hat{\tau} = \langle \tau, \lambda, \delta\rangle$ such that there is a run tree $T_r = \langle \tau, \lambda', \delta\rangle$ such that for all $x \in \tau$, $\lambda(x) = L(\lambda'(x))$. The semantics of a machine P, denoted by $[\![P]\!]$, is the collection of all trace trees of P.

An alphabet $\langle O, I\rangle$ is said to refine another alphabet $\langle O', I'\rangle$ (denoted as $\langle O, I\rangle \preceq \langle O', I'\rangle$) if $O' \subseteq O$ and $I' \subseteq (I \cup O)$. Consider $\langle O, I\rangle \preceq \langle O', I'\rangle$ and a computation $\sigma = \langle\langle \tau, \lambda, \delta\rangle, \langle O, I\rangle\rangle$. The projection of σ onto $\langle O', I'\rangle$ is $[\sigma]_{\langle O', I'\rangle} = \langle\langle \tau, \lambda', \delta'\rangle, \langle O', I'\rangle\rangle$, where for all $x \in \tau$, $\lambda'(x) = \lambda(x) \cap O'$, and for every edge $\langle x, xn\rangle$ of τ, $\delta'(x,n) = (\delta(x,n) \cup \lambda(x)) \cap I'$.

Under suitable projections, parallel composition of Moore machines corresponds to the "intersection" of the set of trace trees for the component machines, while simulation corresponds to trace tree containment. This is formally stated and proved in Propositions 1 and 2 in [7].

4.2 Assume-Guarantee for Moore machines

We will now present a compositional reasoning principle for Moore machines that allows one to compose specifications that contain both safety and liveness constraints. As we shall later show, this rule is more general than previous composition rules for Moore machines.

We will call a set $T \subseteq \mathcal{C}$ projection closed if for any $\sigma \in \mathcal{C}$, with alphabet $\langle O, I\rangle$, $\sigma \in T$ if and only if for σ' such that $[\sigma']_{\langle O, I\rangle} = \sigma$, $\sigma' \in T$. In other words, a set is projection closed if it contains the projections of all of its elements and in addition, contains the computations whose projections are in the set. An operator $F : \mathcal{C} \to \mathcal{C}$ is projection preserving if for any projection closed set T, $F(T)$ is also projection closed.

Theorem 5. *For Moore machines P_1 and P_2 such that $P_1||P_2$ exists, and projection preserving operators F and G, the following rule is sound.*

$$\frac{\begin{array}{c} P_1 \models \nu(F) \triangleright \rho(G) \\ P_2 \models \nu(G) \triangleright \rho(F) \end{array}}{P_1||P_2 \models \rho(F) \wedge \rho(G)} \qquad (Comp\ \ Moore)$$

Theorem 5 illustrates how our general compositional rules can be applied to a specific computational model to obtain circular reasoning principles. Conditions (*1*) and (*2*) in the general rule (*Comp* $\nu \triangleright \rho$) can be seen as describing the relationship between the composition of computations and assume-guarantee specifications. They are, therefore, often eliminated when a specific model is considered.

4.3 Compositional reasoning for Simulation

We now show how the simulation pre-order rule of [7] can be derived from (*Comp Moore*). We begin by defining a natural generalization of projection, which we call observational equivalence. The formal relationship between observational equivalence and projection is captured in Proposition 3.

Definition 4. *Trace trees* $\sigma = \langle\langle \tau, \lambda, \delta\rangle, \langle O, I\rangle\rangle$ *and* $\sigma' = \langle\langle \tau, \lambda', \delta'\rangle, \langle O', I'\rangle\rangle$ *are said to be observationally equivalent if and only if for every* $x \in \tau$ *and edge* $\langle x, xn\rangle$ *in* τ*, the following two conditions hold.*

(1) $\lambda(x) \cap O' = \lambda'(x) \cap O$*, and*
(2) $(\delta(x,n) \cup \lambda(x)) \cap (O' \cup I') = (\delta'(x,n) \cup \lambda'(x)) \cap (O \cup I)$

When σ *is observationally equivalent to* σ'*, we write* $\sigma \approx \sigma'$*. If conditions (1) and (2) hold only for nodes* $x \in \tau$ *such that* $|x| < k$ *then we say* σ *and* σ' *are observationally equivalent upto depth* k *and we denote this by* $\sigma \approx_{<k} \sigma'$*.*

Proposition 3. *Consider computations* $\sigma = \langle\langle \tau, \lambda, \delta\rangle, \langle O, I\rangle\rangle$, $\sigma' = \langle\langle \tau, \lambda', \delta'\rangle, \langle O', I'\rangle\rangle$, *and* $\sigma'' = \langle\langle \tau, \lambda'', \delta''\rangle, \langle O'', I''\rangle\rangle$.

1. *If* $\langle O, I\rangle \preceq \langle O', I'\rangle$ *then* $\sigma \approx \sigma'$ *iff* $[\sigma]_{\langle O', I'\rangle} = \sigma'$.
2. $\sigma' \approx \sigma''$ *iff there is some* σ *such that* $\sigma' = [\sigma]_{\langle O', I'\rangle}$ *and* $\sigma'' = [\sigma]_{\langle O'', I''\rangle}$.

To apply our general assumption-guarantee rule, we recast simulation as a fixed-point definition. Consider a machine $P = \langle S, s_0, I, O, L, R\rangle$, with $S = \{s_0, s_1, \ldots s_n\}$. We will define a specification $\Phi(P)$ as a mutually recursive greatest fixed point, using operators F_{s_i}, for each $s_i \in S$. The intuition behind the definition is as follows: each F_{s_i} constructs trace trees that are observationally equivalent to computations that start in state s_i of machine P. Since simulation corresponds to trace tree containment, the mutually recursive fixed point of F_{s_0} (which will be $\Phi(P)$) will capture the class of all machines Q such that $Q \preceq_s P$.

$$\Phi(P) = \nu_{s_0}(F_{s_0}, F_{s_1}, \ldots, F_{s_n})$$

$$\text{where, } F_{s_i}(Z_{s_0}, \ldots, Z_{s_n}) = \{\sigma = \langle \hat{\tau}, \langle O', I'\rangle\rangle \mid \forall i \in \hat{\tau}.\exists j \leq n.\ \ \sigma|_i \in Z_{s_j},\ \ \lambda(\epsilon) \cap O = L(s_i) \cap O',\ (\delta(\epsilon, i) \cup \lambda(\epsilon)) \cap (O \cup I) = (k \cup L(s_i)) \cap (O' \cup I') \text{ where } R(s_i, k, s_j)\}$$

The intuitions behind the definition of $\Phi(P)$ are captured in the following proposition.

Proposition 4. *$\sigma \models [\Phi(P)]^k$ iff for some $\sigma' \in [\![P]\!]$, $\sigma \approx_{<k} \sigma'$.*

Corollary 2. *1. $\sigma \models \Phi(P)$ iff for some $\sigma' \in [\![P]\!]$, $\sigma \approx \sigma'$.*
2. If $P' \preceq_s P$ then $P' \models \Phi(P)$.

Theorem 6 ([7]). *Let P, Q, P', Q' be Moore machines such that $P||Q$ and $P'||Q'$ exist. Suppose that $P||Q' \preceq_s P'$ and $P'||Q \preceq_s Q'$. Furthermore, every input of $P'||Q'$ is either an input or output of $P||Q$. Then $P||Q \preceq_s P'||Q'$.*

In the theorem above, P' and Q' should be seen as "specifications" of machines P and Q. Our proof of this theorem is based on the following lemma which captures the relationship between parallel composition and assumption-guarantee specifications. The lemma says that if $P||Q \preceq_s P'$ then under suitable conditions on the alphabets of the machines, $P \models \Phi(Q) \triangleright \Phi(P')$. The proof uses Proposition 4 and Corollary 2 which establish the relationship between approximations of $\Phi(P)$ and the simulation relation $\preceq_s$. The proof also crucially relies on the machines being non-blocking to "extend" any correct finite trace tree.

Lemma 2. *Consider machines P with alphabet $\langle O_P, I_P \rangle$, Q with alphabet $\langle O_Q, I_Q \rangle$, and P' with alphabet $\langle O_{P'}, I_{P'} \rangle$ such that $P||Q$ exists. Furthermore, let $O_{P'} \subseteq O_P$ and $I_{P'} \subseteq (O_P \cup I_P) \cup (O_Q \cup I_Q)$. Then $P||Q \preceq_s P'$ iff $P \models \Phi(Q) \triangleright \Phi(P')$.*

5 Conclusion

In this paper, we have proposed a semantics for assume-guarantee specifications of the form $\rho(A) \triangleright \rho'(G)$ where ρ, ρ' are the least or greatest fixed point operators, and sound rules for composing such specifications. A salient feature of our semantics and these rules is that they provide a common formulation for both least and greatest fixed points and arbitrary combinations of them in assumption-guarantee specifications. Using these general rules, we have derived the first formulation of assume-guarantee rules for linear temporal logic formulas φ U ψ and φ $\mathrm{U_W}$ ψ, and a formulation of assume-guarantee rules for Moore machine with a trace-tree semantics.

The general rule also allows us to show that the previously proposed rules of [12, 7] can be derived as special instances. It has been well-known that assume-guarantee rules are not propositionally valid and their soundness requires non-propositional reasoning some guise. In our proofs of the results of [12, 7], it is worth noting that once it was shown that the premises of their rules can be cast as satisfaction of assume-guarantee properties with respect to our semantics, the conclusions of their rules followed *only* by application of our rules (*Imp* $\triangleright$), (*Comp* $\nu \triangleright \nu$), and (*Sub* $\triangleright$) without requiring any non-propositional reasoning such as induction. In this sense, the three rules can be seen as a precise, uniform, and concise distillation of the non-propositional content of the soundness of any assume-guarantee reasoning principle. It is therefore our hope that a logic consisting of these three reasoning rules together with those of propositional logic would yield a powerful general logic for supporting circular compositional reasoning.

References

1. M. Abadi, and L. Lamport. Composing Specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, 1993.
2. M. Abadi, and L. Lamport. Conjoining Specifications. *ACM Transactions on Programming Languages and Systems*, 17(3):507–534, 1995.
3. R. Alur, and T. A. Henzinger. Local liveness for compositional modeling of fair reactive systems. In *Proceedings of the Conference on Computer-Aided Verification*, pages 166–179, 1995.
4. R. Alur, and T. A. Henzinger. Reactive Modules. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 207–218, 1996.
5. A. Cau, and P. Collette. Parallel composition of assumption-commitment specifications: A unifying approach for shared variable and distributed message passing concurrency. *Acta Informatica*, 33:153–176, 1996.
6. O. Grumberg, and D. E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, 1994. Earlier version in *Proceedings of CONCUR 91: Concurrency Theory*, 1991.
7. T. A. Henzinger, S. Qadeer, S. K. Rajamani, and S. Tasiran. An assume-guarantee rule for checking simulation. In *FMCAD 98: Formal Methods in Computer-aided Design*, pages 421–432, 1998.
8. C. B. Jones. Tentative steps towards a development method for infering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619, 1983.
9. B. Jonsson, and Y. -K. Tsay. Assumption/guarantee specifications in linear-time temporal logic. *Theoretical Computer Science*, 167:47–72, 1996
10. R. P. Kurshan. *Computer-aided Verification of Coordinating Processes.* Princeton University Press, 1994.
11. K. McMillan. A compositional rule for hardware design refinement. In *Proceedings of the Conference on Computer-Aided Verification*, pages 24–35, 1997.
12. K. McMillan. Circular compositional reasoning about liveness. In *CHARME 99: Correct Hardware Design and Verification*, pages 342–345, 1999.
13. J. Misra, and K. M. Chandy. Proofs of network processes. *IEEE Transactions on Software Engineering*, SE-7(4):417–426, 1981.
14. P. K. Pandya, and M. Joseph. P-A logic — A compositional proof system for distributed programs. *Distributed Computing*, 5:37–54, 1991.
15. A. Pnueli. In transition from global to modular temporal reasoning about programs. In *Logics and Models of Concurrent Systems*, pages 123–144, 1984.
16. E. W. Stark. A proof technique for rely-guarantee properties In *Proceedings of the Conference on the Foundations of Software Technology and Theoretical Computer Science*, pages 369–391, 1985.
17. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.