On the Completeness of Compositional Reasoning

Kedar S. Namjoshi[1] and Richard J. Trefler[2]

[1] Bell Laboratories, Lucent Technologies
kedar@research.bell-labs.com
[2] AT&T Labs Research
trefler@research.att.com

Abstract. Several proof rules based on the assume-guarantee paradigm have been proposed for compositional reasoning about concurrent systems. Some of the rules are syntactically circular in nature, in that assumptions and guarantees appear to be circularly dependent. While these rules are sound, we show that several such rules are *incomplete*, i.e., there are true properties of a composition that cannot be deduced using these rules. We present a new sound and complete circular rule. We also show that circular and non-circular rules are closely related. For the circular rules defined here, proofs with circular rules can be efficiently transformed to proofs with non-circular rules and vice versa.

1 Introduction

In his landmark paper [Pnu77], Pnueli advocated the use of temporal logic as a formalism for describing the correct operation of reactive systems [HP85]. To show that a reactive system, M, is correct, one specifies the correctness condition for M as an assertion, f, of temporal logic and applies proof techniques, either automatic [CE81,QS82,CES86] or deductive [Pnu77,MP84], to show that M satisfies f ($M \models f$).

Model checking [CE81,QS82] (*cf.* [CES86] [VW86]) is an automatic technique for showing that $M \models f$. It is efficient, with complexity linear in the size of M ($|M|$) for temporal logics such as Computation Tree Logic (CTL) [CE81] and Linear Temporal Logic (LTL) [Pnu77]. However, when M is given as the parallel composition of n processes, each of size bounded by K, the size of M may be K^n. This *state explosion* problem is one of the main obstacles to the more wide-spread application of model checking.

Compositional reasoning techniques form a promising approach to ameliorating the state explosion problem. To prove that the parallel composition of M_1 with M_2, written as $M_1//M_2$, satisfies the correctness specification h, compositional techniques provide proof rules that justify the above correctness assertion from two proofs done in isolation, the first stating the correctness of M_1 and the second stating the correctness of M_2. For example, the following is a typical rule:

$$\frac{\{f\}M_1\{h\} \quad \{h\}M_2\{g\}}{\{f\}M_1//M_2\{g\}}$$

This rule works as follows. First, show $\{f\}M_1\{h\}$, that is, that M_1 satisfies h under the assumption f. Second, show that $\{h\}M_2\{g\}$. Then the correctness assertion $\{f\}M_1//M_2\{g\}$ may be concluded as a consequence of the soundness of the rule. Such compositional reasoning provides a benefit to the extent that direct reasoning about $M_1//M_2$ has been avoided. In general, however, determining the appropriate auxiliary assertion h may be highly non-trivial.

To ease the difficulty of determining the auxiliary assertions, several so called *circular* proof rules have been proposed. For example, consider the following rule (*cf.* [McM99])

$$\frac{\{f\}M_1\{\, g_2 \rhd g_1\} \quad \{f\}M_2\{\, g_1 \rhd g_2\}}{\{f\}M_1//M_2\{\mathsf{G}(g_1 \wedge g_2)\}}$$

Several points seem to differentiate this rule from the previous one. Firstly, the form of the postconditions has been restricted to specific operators. The property $q \rhd p$ (read as "q constrains p") is true of a computation if, for all i, p is true at point i of the computation if q holds at all points $j < i$; this can be expressed in LTL as $\neg(q \;\mathsf{U}\; \neg p)$. The property $\mathsf{G}(p)$ (read as "always p") is true of a computation if p holds at all points of the computation. Secondly, in the first sub-goal, $\{f\}M_1\{\, g_2 \rhd g_1\}$, g_1 is understood to be the correctness assertion of M_1 while g_2 is a helper assertion. This may be justified by thinking of M_1 as an open system, which interacts with an environment over which it has little control. Thus, the correct operation of M_1 may be dependent on the correct operation of its environment M_2, therefore, g_2 appears as a guarantee of the correct operation of the environment in the proof of the correctness of M_1. The appearance of g_1 in the proof sub-task for M_2 can be similarly justified – hence the use of the word "circular" in the name for such proof rules. The circularity helps to more easily encode the back-and-forth handshake protocols that designers typically use for connecting components of a system.

For any proof system, *soundness* is, of course, the most important property – it should not be possible to deduce false facts. A measure of the quality or usefulness of a proof system is obtained from an investigation into *completeness* – is it possible to deduce *all* true facts using the rules of the system? Existing compositional rules, including the circular ones, are known to be sound.

In this paper, we first investigate the completeness of existing compositional reasoning rules, focusing on rules that are known to be sound for arbitrary linear temporal properties and which have been used successfully to verify large systems. Surprisingly, several such rules turn out to be *incomplete*; that is, there are correctness assertions about $M_1//M_2$ which are true but are not provable from the proof rules. Typically these unprovable assertions are liveness properties, but some rules may also be incomplete for safety properties. The counter-examples for incompleteness are also quite simple, which indicates that the rules may be inadequate for handling many compositions that arise in practice. We propose a new circular reasoning rule similar to the one above and show that it is both sound and complete. The new rule strengthens the previous rule in a manner analogous to strengthening a proof of invariance by introducing auxiliary assertions – to show $\mathsf{G}p$ show $\mathsf{G}(p \wedge h)$. Furthermore, our new rule is backward compatible, in that any proof done using the previous rule is also a proof with the new circular rule.

We then investigate whether circularity is, in itself, essential for reasoning about composed systems. We show that the notion of circularity is a somewhat weak one for LTL properties, in that proofs carried out with circular rules can be efficiently translated to proofs with non-circular rules, and vice-versa.

The paper is organized as follows: Section 2 contains some preliminary definitions; Section 3 gives the details of several different styles of proof rules and develops our new sound and complete circular proof rule; Section 4 discusses the translations between proofs carried out with circular and non-circular rules. Finally, Section 5 contains a brief conclusion and discusses related work.

2 Background

In this section, we define the computational model and provide examples of circular and non-circular rules for compositional reasoning.

2.1 Temporal Logic

LTL was first suggested as a protocol specification language in [Pnu77]. Formulae in the logic define sets of *infinite* sequences. We define LTL formulae w.r.t. a set of *variable* symbols. As in first-order logic, one can construct *terms* over the set of variables using function symbols from a vocabulary $\mathcal{F}$, and *atomic predicates* from terms, using relational symbols from a vocabulary $\mathcal{R}$. We define *atomic predicates* and temporal formulas below. A *predicate* is a boolean combination of atomic predicates.

- For a relational symbol $r \in \mathcal{R}$ of arity n and terms $t_0, \ldots, t_{n-1}$, $r(t_0, \ldots, t_{n-1})$ is an atomic predicate and a formula,
- for formulae f and g, $(f \wedge g)$ and $\neg(f)$ are formulae,
- for formulae f and g, $\mathsf{X}(f), \mathsf{X}^-(f), (f \;\mathsf{U}\; g)$, and $(f \;\mathsf{U}^-\; g)$ are formulae.

The temporal operators are X (*next-time*), X^- (*previous-time*), U (*until*), and U^- (*since*). Given an interpretation $\mathcal{I}$ (which we assume fixed from now on) for the function and relation symbols, temporal formulae are interpreted w.r.t. infinite sequences of valuations of the variables. For a set of typed variables W, let a W-*state* be a function mapping each variable in W to a value in its type. The set of W-states is denoted by $\Sigma(W)$. A *W-sequence* σ is an infinite sequence of W-states, which is represented as a function $\sigma : \mathbf{N} \rightarrow \Sigma(W)$ ($\mathbf{N}$ is the set of natural numbers). We write $\sigma, i \models f$ to say that the infinite sequence σ *satisfies* the formula f at position i. The *language* of f, denoted by $\mathcal{L}(f)$, is the set $\{\sigma : \sigma, 0 \models f\}$. The satisfaction relation can be defined by induction on the structure of f. First, the value of a term t at location i on σ, denoted as $\sigma_i(t)$, may be defined by induction on the structure of terms. Next, the satisfaction relation for formulas is defined as follows.

- $\sigma, i \models r(t_0, \ldots, t_{n-1})$ iff $(\mathcal{I}(r))(\sigma_i(t_0), \ldots, \sigma_i(t_{n-1}))$ is true.
- $\sigma, i \models \neg(f)$ iff $\sigma, i \models f$ is false; $\sigma, i \models (f \wedge g)$ iff both $\sigma, i \models f$ and $\sigma, i \models g$ are true.
- $\sigma, i \models \mathsf{X}(f)$ iff $\sigma, i+1 \models f$.
- $\sigma, i \models \mathsf{X}^{-}(f)$ iff $i > 0$ and $\sigma, i-1 \models f$.
- $\sigma, i \models (f \;\mathsf{U}\; g)$ iff there exists j, $j \geq i$, such that $\sigma, j \models g$ and for every k, $i \leq k < j$, $\sigma, k \models f$.
- $\sigma, i \models (f \;\mathsf{U}^{-}\; g)$ iff there exists j, $j \leq i$, such that $\sigma, j \models g$ and for every k, $j < k \leq i$, $\sigma, k \models f$.

Other connectives can be defined in terms of these basic connectives: $(f \vee g)$ is $\neg(\neg f \wedge \neg g)$, $(f \Rightarrow g)$ is $\neg f \vee g$, $\mathsf{F}g$ ("eventually g") is $(\mathit{true} \;\mathsf{U}\; g)$, $\mathsf{F}^{-}g$ ("previously g") is $(\mathit{true} \;\mathsf{U}^{-}\; g)$, $\mathsf{G}f$ ("always f") is $\neg\mathsf{F}(\neg f)$, $(f \;\mathsf{W}\; g)$ ("f holds unless g") is $(\mathsf{G}(f) \vee (f \;\mathsf{U}\; g))$, $\overset{\infty}{\mathsf{F}}p$ ("infinitely often p") is $\mathsf{GF}p$, $\overset{\infty}{\mathsf{G}}p$ ("finitely often $\neg p$") is $\mathsf{FG}p$, and $q \rhd p$ (read as "q constrains p") is $\neg(q \;\mathsf{U}\; \neg p)$.

Quantified Temporal Logic: The expressive power of temporal logic can be enhanced by allowing variable quantification. The formula $(\exists W : f)$ is true of a V-sequence σ iff there is a $V \cup W$-sequence δ that agrees on the V-variables with σ and which satisfies f. In the finite case, the expressive power of quantified temporal logic is that of ω-regular expressions – see [Tho90] for a survey of these issues.

2.2 Computational Model

We adopt a definition of a process similar to those in [Pnu77,AL95,McM99]. A process is specified by giving an initial condition, a transition condition and a fairness condition over a set of variables.

Definition 0 (Process) *A process is specified by a tuple (V, I, T, F) where*

- *V is a finite, nonempty set of typed* variables*. We define a set of primed variables V' that is in 1-1 correspondence with V.*
- *$I(V)$, the* initial condition*, is a predicate on V,*
- *$T(V, V')$, the* transition condition*, is a predicate on $V \cup V'$, which is left-total.*
- *$F(V, V')$, the fairness condition, is a boolean combination of temporal formulas $\overset{\infty}{\mathsf{F}}(p)$ and $\overset{\infty}{\mathsf{G}}(p)$, for predicates p on $V \cup V'$.*

For W such that $V \subseteq W$, a *W-computation* σ of a process is a W-sequence such that $I(\sigma_0)$, and for each $i \in \mathbf{N}$, $T(\sigma_i, \sigma_{i+1})$. By considering $x' \in V'$ as a term that specifies the value of $x \in V$ in the next state, the set of computations can be defined by the temporal formula $I \wedge \mathsf{G}(T)$, interpreted over W-sequences.

Definition 1 (Language) *For a set of variables W such that $V \subseteq W$, the W-language of a process $M = (V, I, T, F)$, denoted by $\mathcal{L}_W(M)$, is the set of W-computations of M that satisfy the fairness condition F. Thus, $\mathcal{L}_W(M)$ can be expressed by the LTL formula $I \wedge \mathsf{G}(T) \wedge F$, interpreted over W-sequences.*

We define process composition so that the language of a composition $M_1//M_2$ is the *intersection* of the languages of M_1 and M_2. The semantics of most hardware description languages follows this model. In addition, as shown in [AL95], with some reasonable restrictions, it holds also of asynchronous models of computation.

Definition 2 (Process Composition) *The composition of processes $M_1 = (V_1, I_1, T_1, F_1)$ and $M_2 = (V_2, I_2, T_2, F_2)$ is denoted by $M_1//M_2$, and is defined as the process (V, I, T, F) where*

- $V = V_1 \cup V_2$,
- $I = I_1 \wedge I_2$,
- $T = T_1 \wedge T_2$,
- $F = F_1 \wedge F_2$

With this definition of composition, it is possible that T is not left-total even though T_1 and T_2 are left-total. In the rest of the paper, we restrict ourselves to those compositions where T is left-total.

Theorem 0 *For a composition $M = M_1//M_2$ and a set of variables W such that $(V_1 \cup V_2) \subseteq W$, $\mathcal{L}_W(M) = \mathcal{L}_W(M_1) \cap \mathcal{L}_W(M_2)$.* $\square$

Definition 3 (Model Checking) *The model checking question is to determine if a property f defined over a variable set W is true of all computations of a program M with a variable set that is a subset of W; i.e., if $(\forall W : \mathcal{L}_W(M) \Rightarrow f)$ holds.*

2.3 Compositional Reasoning

The model checking question for a composition $M_1//M_2$ may be phrased as $(\forall W : \mathcal{L}_W(M_1//M_2) \Rightarrow f)$, which is equivalent, by Theorem 0, to $(\forall W : \mathcal{L}_W(M_1) \wedge \mathcal{L}_W(M_2) \Rightarrow f)$. Compositional reasoning rules convert this question into two separate model checking questions, one explicitly involving M_1 and the other explicitly involving M_2. This separation is typically required in the proofs of large systems because even the symbolic representation (as BDD's) of the transition relation for $M_1//M_2$ is infeasible.

Assume-guarantee rules for composition attempt to generalize the pre- and post-condition reasoning of Hoare logic [Hoa69]. Informally, a triple $\{f\}M\{g\}$ asserts the property that every computation of M which satisfies the *assumption* f satisfies the *guarantee* g. Formally, this can be stated as $(\forall W : f \wedge \mathcal{L}_W(M) \Rightarrow g)$. We state below two typical compositional reasoning rules based on the assume-guarantee formulation: the first is syntactically non-circular, while the second is syntactically circular, in that the assumptions of one process form the guarantees of the other and vice-versa.

Definition 4 (Non-circular Reasoning (NC)) *Show* $\{f\}M_1//M_2\{g\}$ *holds by picking an intermediate property* h *such that* $\{f\}M_1\{h\}$ *and* $\{h\}M_2\{g\}$ *hold.*

Rule NC is sound, which can be shown with simple propositional reasoning from the definitions. It is also (trivially) complete; if $\{f\}M_1//M_2\{g\}$ holds, then choosing $h = f \wedge \mathcal{L}_W(M_1)$, one may show $\{f\}M_1\{h\} = (\forall W : f \wedge \mathcal{L}_W(M_1) \Rightarrow f \wedge \mathcal{L}_W(M_1)) = \mathit{true}$, and $\{h\}M_2\{g\} = (\forall W : f \wedge \mathcal{L}_W(M_1) \wedge \mathcal{L}_W(M_2) \Rightarrow g)$, which is true by the assumption.

The following rule is derived from the application of a property decomposition theorem to compositional reasoning in [McM99] (*cf.* Theorem 1 in [McM99]). Below we make use of the following notation: let $B = \{f_1, \ldots, f_k\}$ be a set of LTL formulae, then the formula B is a shorthand for $(\bigwedge i : f_i)$.

Definition 5 (Syntactically Circular Reasoning (C1)) *Consider the composition* $M = (//j : M_j)$. *Let* $\{g_i\}$ *be a set of properties. To show that* $\{f\}M\{\mathsf{G}(\bigwedge i : g_i)\}$ *holds,*

- *with each* i, *choose a composition* $M(i) = (//k : M_k)$ *where* k *ranges over a* strict subset *of the process indices,*
- *choose a well founded order* $\prec$ *and subsets* Θ_i *and* Δ_i *of the set of properties* $\{g_i\}$, *such that if* $g_j \in \Theta_i$, *then* $j \prec i$,

and show that $\{f\}M(i)\{\,\Delta_i \rhd (\neg\Theta_i \vee g_i)\}$ *holds for all* i.

The requirement that $M(i)$ is a strict sub-composition of M is imposed to prevent trivial applications of this rule with every $M(i)$ equal to M.

3 (In)Complete Proof Rules

We have shown in the previous section that the non-circular rule **NC** is both sound and complete. In this section, we consider the proof rule **C1** and the assume-guarantee rule from [AL95] and show that these circular rules are incomplete, even for *finite-state* processes. Our choice of these rules is guided by two considerations: (i) these rules, unlike many other compositional rules (as discussed in Section 5), are sound for arbitrary linear temporal properties and (ii) they have been used successfully (cf. [McM98]) to verify large systems. We then present a new sound and complete circular rule.

3.1 Incompleteness

To demonstrate that rule **C1** is incomplete, consider the programs below where, informally, M_1 and M_2 juggle four tokens by throwing them back and forth in an circular pattern (the l, r variables indicate left and right "hands", respectively).

program M_1	program M_2
variables l_1, r_1, r_2 : *boolean*	variables l_2, r_2, r_1 : *boolean*
initially $l_1 \wedge r_1$	initially $l_2 \wedge r_2$
transition $(l_1{}' \equiv r_2) \wedge (r_1{}' \equiv l_1)$	transition $(l_2{}' \equiv r_1) \wedge (r_2{}' \equiv l_2)$

As can be checked easily, the property $\mathsf{G}(l_1 \wedge l_2)$ holds of the composition $M_1//M_2$. Applying the substitution $f = \mathit{true}, g_1 = l_1, g_2 = l_2$ to rule **C1**, we obtain the property $\{\mathit{true}\}M_1//M_2\{\mathsf{G}(l_1 \wedge l_2)\}$. However, as can be checked by enumeration, there is no way to define the well founded order $\prec$, and the subsets Θ_i and Δ_i such that the sub-goals of rule **C1** are satisfied. Intuitively, this is because the next value of l_1 is determined by the current value of r_2, which is unconstrained by the assumptions. Hence, the original property, which is true of the composition, cannot be shown using the proof rule **C1**.

One reason that this rule is incomplete is that it does not permit a choice of auxiliary assertions, as in rule **NC**. If auxiliary assertions were allowed, it is easy to see that the strengthened property $\mathsf{G}(l_1 \wedge r_1 \wedge l_2 \wedge r_2)$ can be shown by properly instantiating rule **C1**. This is similar to the incompleteness of the inductive invariance rule for establishing $\mathsf{G}(p)$; one often needs to strengthen p to $p \wedge h$ and show that this strengthened formula is an inductive invariant. We say more about strengthening in Section 3.2.

The circular proof rule presented in [AL95] is given below. In this rule, // represents asynchronous composition and $\hat{\mathcal{L}}(M)$ is defined so that it is insensitive to stuttering, however, $\hat{\mathcal{L}}(M_1//M_2)$ is defined so that it equals $\hat{\mathcal{L}}(M_1) \wedge \hat{\mathcal{L}}(M_2)$. Although this rule allows the choice of auxiliary assertions E_i, it turns out that it is still incomplete, because of the restricted form of the hypotheses. In the definition below, $C(f)$, for an LTL property f, is the strongest safety property that is weaker than f while f_{+v} asserts that if the formula f should become false then the variable v becomes constant (for more details see [AL95]).

Definition 6 (Circular Rule C2 [AL95]) *To show that $E \wedge \hat{\mathcal{L}}(N_1//N_2) \Rightarrow \hat{\mathcal{L}}(M_1//M_2)$ holds, pick E_i and show that for each i in $\{1,2\}$ all the following hold.*

- $C(E) \wedge \bigwedge_{j \in \{1,2\}} C(\hat{\mathcal{L}}(M_j)) \Rightarrow E_i$
- $C(E_i)_{+v} \wedge C(\hat{\mathcal{L}}(N_i)) \Rightarrow C(\hat{\mathcal{L}}(M_i))$
- $E_i \wedge \hat{\mathcal{L}}(N_i) \Rightarrow \hat{\mathcal{L}}(M_i)$

Consider the programs M_1, M_2, N_1 and N_2 given below, all of which have initial condition *true* and a weak-fairness condition on the actions a_1, a_2.

program M_1	program M_2
variables x : *boolean*	variables y : *boolean*
transition a_1: $x := \mathit{true}$, b_1: $x := \mathit{false}$	transition a_2: $y := \mathit{true}$, b_2: $y := \mathit{false}$

Thus, the specification programs M_1 and M_2 define the properties $\mathsf{GF}x$ and $\mathsf{GF}y$ respectively. The implementation programs N_1 and N_2 are as follows.

program N_1
variables x, y : *boolean*
transition a_1: $x := y$

program N_2
variables x, y : *boolean*
transition a_2: $y := true$, b_2: $y := \neg x \wedge y$

It is easy to check that $E \wedge \hat{\mathcal{L}}(N_1//N_2) \Rightarrow \hat{\mathcal{L}}(M_1//M_2))$ with $E \equiv true$. By a result in [AL95], $C(\hat{\mathcal{L}}(M))$ is just the temporal formula for process M without the fairness condition. Thus, $C(\hat{\mathcal{L}}(M_1)) = true \wedge \mathsf{G}(x' \equiv true \vee x' \equiv false \vee x' \equiv x)$, which simplifies to *true*, as does $C(\hat{\mathcal{L}}(M_2))$. Hence, if the first hypothesis is to hold, both E_1 and E_2 must equal *true*. Therefore, by the third hypothesis, $\hat{\mathcal{L}}(N_1) \Rightarrow \hat{\mathcal{L}}(M_1)$, which is false as N_1 admits computations where x is *false* at every point. Hence, rule **C2** is incomplete.

3.2 A sound and complete circular rule

We present a rule that allows the choice of *auxiliary* properties over process interfaces, while retaining the overall style of the proof obligations of rule **C1**. This rule is shown to be sound and complete. For clarity, we restrict the discussion below to the two process case – there is a straightforward generalization to the n process case.

Definition 7 (Circular Reasoning (C3)) *For properties g_1 over V_1 and g_2 over V_2, to show $\{f\}M_1//M_2\{\mathsf{G}(g_1 \wedge g_2)\}$, pick properties h_1 and h_2 for which the following obligations hold.*

1. $\{f\}M_1\{\,(h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}$
2. $\{f\}M_2\{\,(h_1 \wedge g_1) \rhd ((h_1 \Rightarrow g_2) \wedge h_2)\}$

Note that an instance of rule **C1** can be obtained from **C3** by making the substitution $h_1 = true, h_2 = true$.

Theorem 1 (Soundness) *Rule **C3** is sound.*

Proof. Assume that the hypotheses of the rule hold for some choice of h_1 and h_2. Then the guarantees of both hypotheses are true for any computation of $M_1//M_2$ that satisfies f. Consider any such computation σ, and any point i on σ. Assume inductively that for all points j, $j < i$, the property $(g_1 \wedge h_1 \wedge g_2 \wedge h_2)$ holds. By the first hypothesis of rule **C3**, $(h_2 \Rightarrow g_1) \wedge h_1$ holds at point i, and by the second hypothesis, so does $(h_1 \Rightarrow g_2) \wedge h_2$. Hence, the inductive hypothesis holds at point $i + 1$. This shows that $\mathsf{G}(g_1 \wedge h_1 \wedge g_2 \wedge h_2)$ holds of σ, from which it follows that the weaker property $\mathsf{G}(g_1 \wedge g_2)$ also holds of σ. $\Box$

Theorem 2 (Completeness) *Rule **C3** is complete. Furthermore, if f is defined over the interface variables $V_1 \cap V_2$, it is always possible to choose h_1 and h_2 as properties over $V_1 \cap V_2$.*

Proof. In the following, let $V = V_1 \cup V_2$, and suppose that f is defined over $V_1 \cap V_2$. Suppose that $\{f\}M_1//M_2\{\mathsf{G}(g_1 \wedge g_2)\}$ holds. By definition, this is equivalent to $(\forall V : f \wedge \mathcal{L}_V(M_1) \wedge \mathcal{L}_V(M_2) \Rightarrow \mathsf{G}(g_1 \wedge g_2))$, which is equivalent to

$(\forall V : f \wedge \mathcal{L}_V(M_1) \wedge \mathcal{L}_V(M_2) \Rightarrow \mathsf{G}(g_1))$ and $(\forall V : f \wedge \mathcal{L}_V(M_1) \wedge \mathcal{L}_V(M_2) \Rightarrow \mathsf{G}(g_2))$ both being true. Consider the first property:

$$(\forall V : f \wedge \mathcal{L}_V(M_1) \wedge \mathcal{L}_V(M_2) \Rightarrow \mathsf{G}(g_1)) \tag{1}$$

Let $\alpha_1 = (\exists V \backslash V_2 : f \wedge \mathcal{L}_V(M_1))$ and $\alpha_2 = (\exists V \backslash V_1 : f \wedge \mathcal{L}_V(M_2))$. Define h_1 as $\mathsf{F}^{-}\alpha_1$ and h_2 as $\mathsf{F}^{-}\alpha_2$. As M_1 and g_1 are defined over V_1, expression 1 can be re-written as:

$$(\forall V_1 : \mathcal{L}_V(M_1) \wedge \alpha_2 \Rightarrow \mathsf{G}(g_1)) \tag{2}$$

By definition of α_1, it is also true that:

$$(\forall V : f \wedge \mathcal{L}_V(M_1) \Rightarrow \alpha_1) \tag{3}$$

Consider the first hypothesis: $\{f\}M_1\{\,(h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}$, and consider any sequence satisfying $f \wedge \mathcal{L}_V(M_1)$. By equation 3, α_1 is true initially, so $\mathsf{G}(h_1)$ is true. Now consider an arbitrary position i on the sequence such that $(h_2 \wedge g_2)$ holds for all positions j, $j < i$.

case $i = 0$: We have to show that $(h_2 \Rightarrow g_1)$ at position 0. If h_2 is true initially, then α_2 is true initially. By equation 2, g_1 must then be true.

case $i > 0$: As $i > 0$, h_2 holds initially, which implies that α_2 is true initially. By equation 2, $\mathsf{G}(g_1)$ holds at the origin, so that $(h_2 \Rightarrow g_1)$ is true at point i.

Hence, the first hypothesis is true. In a similar manner, one may argue that the second hypothesis is also true; so the rule is complete. Note that the auxiliary assertions h_1 and h_2 are defined over the common interface variables $V_1 \cap V_2$. $\square$

For the first example, which showed the incompleteness of rule **C1**, the following choices for h_1 and h_2 over the common variables $\{r_1, r_2\}$ ensure that the hypotheses of rule **C3** hold: $h_1 = r_1, h_2 = r_2$.

For the second example, which showed the incompleteness of rule **C2**, the property to be satisfied by $N_1//N_2$ may be written as $\mathsf{G}(\mathsf{GF}x \wedge \mathsf{GF}y)$. The following choices for h_1 and h_2 over the common variables x, y ensure that the hypotheses of rule **C3** hold: $h_1 = \mathit{true}$, $h_2 = \mathsf{GF}y$.

Interestingly[1], if we modify the hypotheses of rule **C3** so that they have the form $\{f\}M_1\{\mathsf{G}(h_1) \wedge (g_2 \wedge h_2) \rhd (h_2 \Rightarrow g_1)\}$, which is also sound and complete, then these hypotheses may be obtained from rule **C1** for the composition $M_1//M_2$ with the following substitutions: $g_1 = g_1, g_2 = g_2, g_3 = h_1, g_4 = h_2, \Delta_1 = \{g_2, g_4\}, \Theta_1 = \{g_4\}, \Delta_2 = \{g_1, g_3\}, \Theta_2 = \{g_3\}$. The other Θ and Δ variables equal $\emptyset$. We also choose $M(1) = M_1, M(2) = M_2, M(3) = M_1, M(4) = M_2$ and the relation $\prec$: $3 \prec 2, 4 \prec 1$. This shows that modifying rule **C1** by allowing the g_i's to be *augmented with auxiliary assertions* h_i to form $(g_i \wedge h_i)$, results in a sound and complete rule.

4 Translating Proofs

In this section we show how proofs derived using the circular rule **C3** can be translated into proofs using the non-circular rule **NC** and vice-versa. We also

[1] We thank an anonymous referee for this observation.

discuss some of the consequences of these translations. In the sequel, when W is clear from the context, we will write $(\forall W : f)$ simply as f.

Theorem 3 *(From Circular to Non-Circular) Suppose* $\{f\}M_1//M_2\{\mathsf{G}(g_1 \wedge g_2)\}$ *has been derived from the circular rule **C3**. Then* $\{f\}M_1//M_2\{\mathsf{G}(g_1 \wedge g_2)\}$ *may be derived by application of the rule **NC** by letting the intermediate assertion* h *equal* $f \wedge\ (g_2 \wedge h_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)$.

Proof. $\{f\}M_1\{f \wedge\ (g_2 \wedge h_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}$, the first requirement of rule **NC**, follows as a direct result of the first proof obligation from **C3** in the premise. We now show why the second requirement of rule **NC** also holds.

$$
\begin{array}{ll}
 & \{f \wedge\ (h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}M_2\{\mathsf{G}(g_1 \wedge g_2)\} \\
\equiv & \quad (\text{ by the definition of } \{f\}M\{g\}\ ) \\
 & \{f\}M_2\{\ (h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1) \Rightarrow \mathsf{G}(g_1 \wedge g_2)\} \\
\Leftarrow & \quad (\text{ by the second proof obligation of rule } \mathbf{C3}\ ) \\
 & (h_1 \wedge g_1) \rhd ((h_1 \Rightarrow g_2) \wedge h_2) \Rightarrow \\
 & [\,(h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1) \Rightarrow \mathsf{G}(g_1 \wedge g_2)] \\
\equiv & \quad (\text{ re-arranging }) \\
 & [\,(h_1 \wedge g_1) \rhd ((h_1 \Rightarrow g_2) \wedge h_2)] \wedge [\,(h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)] \Rightarrow \\
 & \mathsf{G}(g_1 \wedge g_2) \\
\equiv & \quad (\text{ temporal logic (see proof of Theorem 1) }) \\
 & \mathit{true}
\end{array}
$$

□

Theorem 4 *(From Non-Circular to Circular) Suppose* $\{f\}M_1//M_2\{g\}$ *has been derived from the non-circular rule **NC** using the intermediate assumption* h. *Then the conclusion* $\{f\}M_1//M_2\{g\}$ *may be derived by application of the rule **C3** using the substitution* $h_1 = \mathsf{F}^{-}h$, $h_2 = \mathit{true}$, $g_1 = \mathit{true}$ *and* $g_2 = \mathsf{F}^{-}g$.

Proof. Firstly, we note that the conclusion of the rule is the desired one. It is straightforward to show that, at the initial point, any computation satisfies $\mathsf{GF}^{-}g$ iff it satisfies g. Therefore, $\{f\}M_1//M_2\{\mathsf{G}(g1 \wedge g2)\}$ is equivalent to $\{f\}M_1//M_2\{g\}$.
Consider the first proof obligation.

$$
\begin{array}{ll}
 & \{f\}M_1\{\ (h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\} \\
\equiv & \quad (\text{ substituting and simplifying }) \\
 & \{f\}M_1\{\ g_2 \rhd h_1\} \\
\Leftarrow & \quad (\ \rhd \text{ is anti-monotone in its first argument }) \\
 & \{f\}M_1\{\ \mathit{true} \rhd \mathsf{F}^{-}(h)\} \\
\equiv & \quad (\text{ as } \mathit{true} \rhd \mathsf{F}^{-}(h) \equiv \mathsf{GF}^{-}(h)\ ) \\
 & \{f\}M_1\{\mathsf{GF}^{-}(h)\} \\
\equiv & \quad (\text{ as } \mathsf{GF}^{-}(p) \text{ and } p \text{ are equivalent at the initial point }) \\
 & \{f\}M_1\{h\} \\
\equiv & \quad (\text{ by the first premise in the supposition }) \\
 & \mathit{true}
\end{array}
$$

Now we show that the second premise is also true.

$$
\begin{array}{ll}
 & \{f\}M_2\{\,(h_1 \wedge g_1) \rhd ((h_1 \Rightarrow g_2) \wedge h_2)\} \\
\equiv & \quad (\text{ substituting and simplifying }) \\
 & \{f\}M_2\{\,h_1 \rhd (h_1 \Rightarrow g_2)\} \\
\equiv & \quad (\text{ definition of } h_1, g_2 \text{ }) \\
 & \{f\}M_2\{\,\mathsf{F}^-(h) \rhd (\mathsf{F}^-(h) \Rightarrow \mathsf{F}^-(g))\}
\end{array}
$$

This follows by an induction on positions of any sequence satisfying $f \wedge \mathcal{L}(M_2)$. At the initial position, by the second part of **NC**, $(h \Rightarrow g)$ holds, thus, $(\mathsf{F}^-h \Rightarrow \mathsf{F}^-g)$ holds at the initial position. At any other position i, by the assumption F^-h for 0 up to $i-1$, h must be true initially, hence, g is true initially by the second part of **NC**, so that F^-g is true at position i. □

We note that the translations make no use of quantified formulae, which justifies the following corollary.

Corollary 0 *Compositional Rule **C3** is complete for linear temporal logic.*

Proof. As in the proof of the previous theorem, one can write a property f as $\mathsf{G}(\mathit{true} \wedge \mathsf{F}^-(f))$ and apply rule **C3**. Since **C3** was shown to be complete for properties of the form $\mathsf{G}(g_1 \wedge g_2)$ it follows that **C3** is sufficient for proving any linear temporal logic property. We note that the proof of Theorem 2, presented above, does make use of quantified temporal properties. This use of quantification is beneficial, in that the properties constructed in the proof of completeness may be restricted to the variables which are mentioned in the interface between M_1 and M_2. However, it is possible to prove Theorem 2 without reference to quantified formulae (this proof has been left out for space reasons) and hence the result follows. □

4.1 The Cost of a Proof

In this section we discuss the computational cost of applying the various rules. The goal is to calculate the cost of translating an instance of the use of one proof rule into the use of another proof rule and to get a measure on the complexity of a proof rule.

Consider the proof obligation $\{f\}M\{g\}$ where f and g are pure LTL formulae (they contain no quantifiers). If such a proof were done by hand then any complexity measure would be, at best, highly subjective. Suppose, on the other hand, that this proof were given to a model checker. The obligation amounts to showing that every W-computation of M that satisfies f also satisfies g. Typically, this is done as follows [VW86]. First, translate f, M and $\neg g$ into automata A_f, A_M and $A_{\neg g}$, respectively, such that the set of sequences accepted by the automaton is exactly the set of sequences accepted by the corresponding formula. Then $\mathcal{L}(A_f \cap A_M \cap A_{\neg g}) = \emptyset$ iff the set of computations of M which satisfy f all satisfy g. The complexity of the translation is approximately linear in $|M|$ and exponential in the lengths of f and g [VW86] [LPZ85]. This leads to the following definition.

Definition 8 (Proof Cost) *The cost of a proof obligation $\{f\}M\{g\}$, for temporal logic formulae f and g and finite state structure M is $2^{|f|+|g|}|M|$.*

Note that $|\, q \rhd p| = |\neg(q \mathsf{U} \neg(p))|$, which equals $|p| + |q| + 3$. Suppose that we have translated a circular proof to a non-circular proof via the method outlined in Theorem 3. Without loss of generality, assume that $|M_1| \leq |M_2|$, $|h_1| \leq |h_2|$ and $|g_1| \leq |g_2|$. The cost, c, of showing the premises of rule **C3**

- $\{f\}M_1\{\,(h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}$ and
- $\{f\}M_2\{\,(h_1 \wedge g_1) \rhd ((h_1 \Rightarrow g_2) \wedge h_2)\}$

is bounded by $2^{|f|+3|h_2|+2|g_2|+8}|M_2|$. The cost, nc, of showing the translated premises for rule **NC**

- $\{f\}M_1\{f \wedge\, (h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}$ and
- $\{f \wedge\, (h_2 \wedge g_2) \rhd ((h_2 \Rightarrow g_1) \wedge h_1)\}M_2\{\mathsf{G}(g_1 \wedge g_2)\}$

is bounded by $2^{2|f|+3|h_2|+4|g_2|+11}|M_2|$.

So the cost of the circular proof is bounded by $2^{\alpha}|M_2|$, where α is a function linear in the sizes of f, g_i and h_i and the cost of the non-circular proof is bounded by $2^{2\alpha}|M_2|$, so the translation process can be said to be efficient.

Now suppose that we have translated a non-circular proof to a circular proof via the method outlined in Theorem 4. The cost, nc, of showing the premises for rule **NC**

- $\{f\}M_1\{h\}$ and
- $\{h\}M_2\{g\}$

is bounded by $2^{|f|+|h|+|g|+2}|M_2|$. The cost, c of showing the simplified translated premises for rule **C3**

- $\{f\}M_1\{\,\mathsf{F}^-(g) \rhd \mathsf{F}^-(h)\}$ and
- $\{f\}M_2\{\,\mathsf{F}^-(h) \rhd (\mathsf{F}^-(h) \Rightarrow \mathsf{F}^-(g))\}$

is bounded by $2^{|f|+2|h|+|g|+6}|M_2|$. Hence translating from circular to non-circular is, essentially, efficient.

5 Related Work

There are several proposals for compositional reasoning rules in the literature, but only a few investigations of the completeness of these rules – a good survey of the field appears in the COMPOS97 proceedings [dRLP97]. The earliest proposals for assume-guarantee reasoning are from [Jon81,CM81] – these are concerned with establishing safety properties of networks of processes. Zwiers' book [Zwi89] contains much of the groundwork necessary for reasoning about compositional proof systems. Proofs of the completeness of compositional reasoning systems for safety properties are found in [ZdRvE84][Pan88][PJ91] [dRdBH+99]. Other assume-guarantee rules for safety properties are proposed in [Sta85] [Pnu85]

[Kur87] [AH96] [McM97]. More general rules that apply to both safety and liveness properties are proposed in [Pnu85] [Jos87] [CLM89] [GL94] [AL95] [McM99].

We have concentrated on the completeness question for general rules that apply to both safety and liveness properties. As shown in Section 3, the circular rules in [AL95] and the rule **C1** derived from [McM99] are incomplete. The circular rule presented in [HQRT98] for the simulation-based verification paradigm is also incomplete – for lack of space, this proof is left for the full paper. The simplicity of the counter-examples suggests that the incompleteness may indeed impact the verification of systems in practice. We present a new circular rule, which is a modification of rule **C1**, and show it to be sound and complete for all of LTL – in fact it is straightforward to generalize these ideas so that the rule is complete for the ω-regular languages. The proofs carried out using rule **C1**, including that of the Tomasulo algorithm [McM98], can be carried out in exactly the same manner with the new rule. We also investigate whether circularity is, in itself, essential for reasoning about composed systems, and show that for assume-guarantee reasoning in LTL, the notion of circularity is a somewhat weak one, in that proofs carried out with circular rules are efficiently translatable into proofs with non-circular rules, and vice-versa.

The computational complexity of establishing an assume-guarantee triple has been studied extensively in [GL94,KV95,KV97], for various combinations of specification logics. We have considered a different question, that of the complexity of translating between proofs obtained with different compositional rules, whenever this is possible.

There are a number of ways one could choose to strengthen the circular proof rules found in the literature in order to make them complete. We have chosen one in particular, rule **C3**. Our choice was motivated by a desire to remain as close as possible to the spirit of the original circular proof rule - namely, to avoid the "direct" use of $M_1//M_2$ when proving properties about this composition. Specifically, we have decided not to allow the use of temporal implication, $h \Rightarrow g$, as a proof rule [MP95]. Our results show that implication is not necessary in order to obtain a sound and complete rule. Furthermore, rules that include implication may allow the proof of $f \wedge \mathcal{L}_W(M_1) \wedge \mathcal{L}_W(M_2) \Rightarrow g$ to be instantiated directly as an implication without use of the, hopefully, better rules mentioning only M_1 or M_2. That this goal has been, to some extent, mitigated against in our proof of completeness should not come as a surprise in light of the difficulty of the problem.

Acknowledgment: The authors thank Willem-Paul de Roever and the anonymous referees for many interesting and helpful comments.

References

[AH96] R. Alur and T. Henzinger. Reactive modules. In *IEEE LICS*, 1996.

[AL95] M. Abadi and L. Lamport. Conjoining specifications. *ACM Trans. on Programming Languages and Systems (TOPLAS)*, May 1995.

[CE81] E.M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, volume 131 of *LNCS*. Springer-Verlag, 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2), 1986.

[CLM89] E.M. Clarke, D.E. Long, and K.L. McMillan. Compositional model checking. In *IEEE LICS*, 1989.

[CM81] K.M. Chandy and J. Misra. Proofs of networks of processes. *IEEE Transactions on Software Engineering*, 7(4), 1981.

[dRdBH+99] W-P. de Roever, F. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, and J. Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Proof Methods.* Draft book, 1999.

[dRLP97] W-P. de Roever, H. Langmaack, and A. Pnueli, editors. *Compositionality: The Significant Difference*, volume 1536 of *LNCS*. Springer-Verlag, 1997.

[GL94] O. Grümberg and D.E. Long. Model checking and modular verification. *ACM Trans. on Programming Languages and Systems (TOPLAS)*, 1994.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 1969.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. In K. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 477–498. Springer-Verlag, 1985.

[HQRT98] T.A. Henzinger, S. Qadeer, S.K. Rajamani, and S. Tasiran. An assume-guarantee rule for checking simulation. In *FMCAD*, volume 1522 of *LNCS*, 1998.

[Jon81] C.B. Jones. *Development methods for computer programs including a notion of interference.* PhD thesis, Oxford University, 1981.

[Jos87] B. Josko. Model checking of CTL formulae under liveness assumptions. In *ICALP*, volume 267 of *LNCS*, 1987.

[Kur87] R.P. Kurshan. Reducibility in analysis of coordination. In *Discrete Event Systems: Models and Applications*, volume 103 of *LNCIS*, 1987.

[KV95] O. Kupferman and M. Vardi. On the complexity of branching modular model checking. In *CONCUR*, volume 962 of *LNCS*, 1995.

[KV97] O. Kupferman and M. Vardi. Module checking revisited. In *CAV*, volume 1254 of *LNCS*, 1997.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. of the Conf. on Logics of Programs*, 1985.

[McM97] K.L. McMillan. A compositional rule for hardware design refinement. In *CAV*, volume 1254 of *LNCS*, 1997.

[McM98] K.L. McMillan. Verification of an implementation of Tomasulo's algorithm by compositional model checking. In *CAV*, volume 1427 of *LNCS*, 1998.

[McM99] K.L. McMillan. Circular compositional reasoning about liveness. In *CHARME*, volume 1703 of *LNCS*, 1999.

[MP84] Z. Manna and A. Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Science of Computer Programming*, 1984.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, 1995.

[Pan88] P. Pandya. *Compositional Verification of Distributed Programs.* PhD thesis, University of Bombay, 1988.

[PJ91] P. Pandya and M. Joseph. P-A logic - a compositional proof system for distributed programs. *Distributed Computing*, 1991.

[Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS*, 1977.

[Pnu85] A. Pnueli. In transition from global to modular reasoning about programs. In *Logics and Models of Concurrent Systems*, NATO ASI Series, 1985.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. of the 5th International Symposium on Programming*, volume 137 of *LNCS*, 1982.

[Sta85] E. Stark. A proof technique for rely/guarantee properties. In *FST&TCS*, volume 206 of *LNCS*, 1985.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, edited by J. van Leeuwen.* Elsevier and MIT Press, 1990.

[VW86] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *IEEE Symposium on Logic in Computer Science*, 1986.

[ZdRvE84] J. Zwiers, W.P. de Roever, and P. van EmdeBoas. Compositionality and concurrent networks: Soundness and completeness of a proof system. *Technical Report, University of Nijmegen*, 1984.

[Zwi89] J. Zwiers. *Compositionality, Concurrency and Partial Correctness*, volume 321 of *LNCS*. Springer-Verlag, 1989.