An Abstract Account of Composition

Martín Abadi[1] and Stephan Merz[2]

[1] Digital Equipment Corporation, Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, U.S.A.
[2] Institut für Informatik, Technische Universität München, Arcisstr. 21, 80290 München, Germany

Abstract. We present a logic of specifications of reactive systems. The logic is independent of particular computational models, but it captures common patterns of reasoning with assumption-commitment specifications. We use the logic for deriving proof rules for TLA and CTL* specifications.

1 Assumption-commitment specifications

Modularity is a central concern in the design of specification methods. In general terms, modularity is the ability to reduce reasoning about a complete system to reasoning about its components. These components are not expected to operate in fully arbitrary environments. In the context of the complete system, each component can assume that its environment is to some extent well behaved, for instance that it adheres to certain communication protocols. Therefore, it is common to specify each component by describing both the function required of the component and the properties assumed of its environment. In the realm of sequential programs, for example, the requirements are postconditions and the assumptions are preconditions. In the broader realm of reactive systems, which we consider in this paper, there are several forms of assumption-commitment specifications [17, 11, 18, 1, 2, 19, 6, 22, 7, 12].

An assumption-commitment specification for a component of a reactive system consists of a formula A, which expresses assumptions about the environment, and a formula C, which expresses the requirements that an implementation of the component has to meet. Clearly, the meaning of such a specification depends somehow on A and C. In the simplest approach, the meaning is that if A holds then C holds, and if A does not hold then the implementation is completely unconstrained. One may therefore write the specification in the form $A \Rightarrow C$, where $\Rightarrow$ is the classical implication connective. This formulation seems intuitive; as we will see, however, stronger forms of assumption-commitment specifications are preferable for reasoning.

Suppose that we have two components specified by $A_1 \Rightarrow C_1$ and $A_2 \Rightarrow C_2$, and that we would like to prove that their composition satisfies the property P. Representing composition by conjunction and implementation by implication, we would have to prove that the formula $(A_1 \Rightarrow C_1) \wedge (A_2 \Rightarrow C_2) \Rightarrow P$ is valid. In the composite system, each component is part of the other's environment;

therefore, the assumption of each component may reflect the commitment of the other. So we would like to be able to use the commitment of each component to discharge the other's assumption. This argument suggests the following proof rule:

$$\frac{\begin{array}{c} C_1 \Rightarrow A_2 \quad C_2 \Rightarrow A_1 \\ C_1 \wedge C_2 \Rightarrow P \end{array}}{(A_1 \Rightarrow C_1) \wedge (A_2 \Rightarrow C_2) \Rightarrow P}$$

Unfortunately, this rule is unsound—it is an instance of circular reasoning.

One remedy is to strengthen the hypotheses in order to break the circularity, for example by demanding that $A_1 \vee A_2$ be valid; with this, the rule becomes classically sound but weak. Another remedy, which is more useful and common, is to strengthen assumption-commitment specifications. For each specification $A \Rightarrow C$, one can add that C holds for longer than A in case A does not hold forever. The motivation for the stronger form of specification comes from the observation that no reasonable implementation of $A \Rightarrow C$ will produce an execution that violates first C and later A. To do this, an implementation would have to predict the future behavior of the environment. In some formalisms, such prophetic implementations are excluded altogether, for example by continuity requirements.

The literature contains a number of sound variants of the unsound rule. Typically, the sound variants are justified using induction along computations. In one of the simplest cases, the reasoning may go: C_1 is not falsified before A_1, which is implied by C_2, so C_1 is not falsified before C_2; analogously, C_2 is not falsified before C_1; hence neither C_1 nor C_2 is ever falsified; therefore, P must hold, since $C_1 \wedge C_2$ implies P. There are many more delicate and more powerful arguments.

Despite the breadth of the literature, we believe that a few general logical ideas account for an interesting part of the work on composition. The purpose of this paper is to present an abstract logic of specifications; the logic is independent of particular computational models, but it captures much of the reasoning common in formalisms with assumption-commitment specifications. Using the abstract logic, we derive proof rules for concrete specification methods. However, we do not attempt to capture every aspect of these specification methods; we focus on simple, basic results with broad applicability.

Our logic borrows from that of Abadi and Plotkin [5]. In particular, we take the idea of using intuitionistic reasoning for assumption-commitment specifications. However, for the sake of simplicity and generality, we do not adopt some non-standard constructs of that logic (for example, "constrains at most").

Cau, Collette, and Xu have given another interesting, unifying perspective on rules for composition [22, 7]. Their work treats abstract rules semantically; concurrent processes, with either shared variables or message passing, are then embedded in a common semantic structure based on labelled sequences. This structure could provide a model for our logic (much like the model of section 3). In comparison, our abstract treatment of composition is primarily syntactic. We resort to semantic reasoning about computations only in applications to particular formalisms. This approach enables us to consider non-linear well-founded structures, for example in the context of branching-time logics.

The next section introduces our abstract logic. Sections 3 and 4 apply the logic to justify rules for TLA [16] and for CTL* [8]. Section 5, in conclusion, discusses the results.

2 A logic of specifications

Our logic of specifications is a propositional intuitionistic logic. We use the standard connectives $\wedge$ and $\rightarrow$. In addition, we introduce a new connective, $\xrightarrow{+}$; this connective will be useful in treating assumption-commitment specifications. In the models of interest to us, $P \xrightarrow{+} Q$ is equivalent to $(Q \rightarrow P) \rightarrow Q$. Next we discuss the models in some detail.

Assume given a nonempty set Σ and a pre-order $\sqsubseteq$ on Σ. We define $\sigma \sqsubset \tau$ as $\sigma \sqsubseteq \tau$ and $\tau \not\sqsubseteq \sigma$. A set $S \subseteq \Sigma$ is downward closed if $\tau \in S$ and $\sigma \sqsubseteq \tau$ imply that $\sigma \in S$. We take Σ as the set of worlds of a Kripke frame [20, p.77], whose accessibility relation is the inverse of $\sqsubseteq$ (that is, τ is accessible from σ iff $\tau \sqsubseteq \sigma$). Since this accessibility relation is reflexive and transitive, we obtain a Kripke model of propositional intuitionistic logic. The interpretation of atomic propositions, $\wedge$, and $\rightarrow$ is the standard one; we give an interpretation for $\xrightarrow{+}$:

- Each atomic proposition has a truth value at each element of Σ. It is required that atomic propositions are true on downward-closed subsets of Σ: if p_i is an atomic proposition, $\sigma \models p_i$, and $\sigma' \sqsubseteq \sigma$, then $\sigma' \models p_i$.
- For the connectives, we have:
  $\sigma \models P \wedge Q$ iff $\sigma \models P$ and $\sigma \models Q$
  $\sigma \models P \rightarrow Q$ iff for all $\tau \sqsubseteq \sigma$: if $\tau \models P$ then $\tau \models Q$
  $\sigma \models P \xrightarrow{+} Q$ iff for all $\tau \sqsubseteq \sigma$: if $\rho \models P$ for all $\rho \sqsubset \tau$ then $\tau \models Q$

It follows from these definitions that all formulas are true on downward-closed subsets of Σ. Below, we sometimes identify propositions and downward-closed subsets of Σ.

Somewhat surprisingly, $\xrightarrow{+}$ can be defined from $\rightarrow$ if $\sqsubset$ is a well-founded relation:

Proposition 1. *Assume that $(\Sigma, \sqsubseteq)$ is a pre-order and that $\sqsubset$ is a well-founded relation on Σ. For all $\sigma \in \Sigma$ and all formulas P and Q,*

$$\sigma \models P \xrightarrow{+} Q \quad \textit{iff} \quad \sigma \models (Q \rightarrow P) \rightarrow Q$$

Proof. "only if": The proof proceeds by well-founded induction on $\sqsubset$, exploiting the hypothesis that $\sqsubset$ is well-founded. Assume that $\sigma \models P \xrightarrow{+} Q$, that $\tau \sqsubseteq \sigma$, and that $\tau \models Q \rightarrow P$, to prove that $\tau \models Q$. Since $\sigma \models P \xrightarrow{+} Q$, if $\rho \models P$ for all $\rho \sqsubset \tau$ then $\tau \models Q$. Therefore, we let $\rho \sqsubset \tau$ and prove that $\rho \models P$. Since $\sigma \models P \xrightarrow{+} Q$, downward closure yields $\tau \models P \xrightarrow{+} Q$ and $\rho \models P \xrightarrow{+} Q$. By induction hypothesis,

$\rho \models (Q \rightarrow P) \rightarrow Q$. Since $\tau \models Q \rightarrow P$, downward closure yields $\rho \models Q \rightarrow P$. Finally, $\rho \models Q$ and $\rho \models P$ follow by intuitionistic logic.

"if": Assume that $\sigma \models (Q \rightarrow P) \rightarrow Q$, that $\tau \sqsubseteq \sigma$, and that $\rho \models P$ for all $\rho \sqsubset \tau$; we have to show that $\tau \models Q$. Assume, to the contrary, that $\tau \not\models Q$. First, we show that $\tau \models Q \rightarrow P$. Assume that $\rho \models Q$ for some $\rho \sqsubseteq \tau$. Either $\tau \sqsubseteq \rho$ or $\rho \sqsubset \tau$. If $\tau \sqsubseteq \rho$ then $\tau \models Q$ by downward closure, in contradiction with our assumptions. Hence, $\rho \sqsubset \tau$, so $\rho \models P$; it follows that $\tau \models Q \rightarrow P$. Since $\sigma \models (Q \rightarrow P) \rightarrow Q$, downward closure yields $\tau \models (Q \rightarrow P) \rightarrow Q$. Finally, $\tau \models Q$ follows by intuitionistic logic. $\diamondsuit$

From now on, we assume that $\sqsubset$ is well-founded, and treat $P \xrightarrow{+} Q$ as if it were a shorthand for $(Q \rightarrow P) \rightarrow Q$. The original semantic definition of $P \xrightarrow{+} Q$ is still important, as it gives the meaning of $(Q \rightarrow P) \rightarrow Q$ directly and clearly.

We can reason syntactically about $\rightarrow$ and $\xrightarrow{+}$, using any of the standard axiomatizations of propositional intuitionistic logic. We adopt sequent notation; the sequent $P_1, \ldots, P_n \vdash P$ means that the conjunction of $P_1, \ldots, P_n$ implies P.

Proposition 2. *The following sequents are derivable:*

$$P \xrightarrow{+} Q, P \xrightarrow{+} (Q \rightarrow R) \vdash P \xrightarrow{+} R \tag{1}$$

$$\bigwedge_{i \in I}(P_i \xrightarrow{+} Q_i) \vdash (\bigwedge_{i \in I} P_i) \xrightarrow{+} (\bigwedge_{i \in I} Q_i) \tag{2}$$

$$P \xrightarrow{+} Q \vdash P \rightarrow Q \tag{3}$$

$$P \xrightarrow{+} P \vdash P \tag{4}$$

Sequents (1) and (2) state implication-like properties of $\xrightarrow{+}$. Sequent (3) says that $\xrightarrow{+}$ is stronger than $\rightarrow$. Sequent (4) can be understood as an abstract formulation of computational induction.

Beyond these elementary results, we are interested in sequents that represent rules for refining specifications, as we explain below. Adopting the convention that $\wedge$ binds tighter than $\rightarrow$ and $\xrightarrow{+}$, we obtain the following results:

Theorem 3. *The following sequents are derivable:*

$$P \rightarrow (Q \rightarrow P') \vdash (P' \xrightarrow{+} Q) \rightarrow (P \xrightarrow{+} Q) \tag{5}$$

$$P \rightarrow (P' \wedge Q \xrightarrow{+} P') \vdash (P' \rightarrow Q) \rightarrow (P \rightarrow Q) \tag{6}$$

We present two concrete interpretations of the logic in the remainder of the paper. Very roughly, Σ represents a set of computations, and $\sigma \sqsubseteq \tau$ holds if computation σ may evolve to computation τ. A proposition represents a specification; $\sigma \models P$ means that σ is allowed by P. From this perspective, $\sigma \models P \rightarrow Q$ means that Q holds for at least as long as P along σ; similarly, $\sigma \models P \xrightarrow{+} Q$ means that Q holds for strictly longer than P (or forever) along σ. We can write assumption-commitment specifications in either of the forms $P \rightarrow Q$ and $P \xrightarrow{+} Q$.

We also view $P \rightarrow Q$ as asserting that P refines Q, because $P \rightarrow Q$ is valid iff every computation allowed by P is also allowed by Q. Correspondingly, refinement rules for assumption-commitment specifications establish formulas of the forms $(P' \rightarrow Q') \rightarrow (P \rightarrow Q)$ or $(P' \overset{+}{\rightarrow} Q') \rightarrow (P \overset{+}{\rightarrow} Q)$. Theorem 3 deals with special cases of such formulas. That theorem allows us to use the commitment Q to establish the assumption P'. In this respect, it contains the essence of rules for composing mutually dependent assumption-commitment specifications. Despite its circular flavor, it is sound because of the distinction between $\rightarrow$ and $\overset{+}{\rightarrow}$.

3 Composition in TLA

In our first application of the general logic, we consider specifications written in linear-time temporal logics. For concreteness we emphasize a particular linear-time temporal logic, TLA [16]. Using the tools of section 2, we reproduce part of the previous work on assumption-commitment specifications in TLA [2].

Formulas of linear-time temporal logics are normally interpreted over infinite sequences $\sigma = \langle s_0, s_1, \ldots \rangle$ of states. A formula is valid if it is holds of all sequences of states. For the formulation of assumption-commitment specifications, it is convenient to interpret formulas also over finite sequences, as follows: a formula F holds of a finite sequence ρ if ρ is empty or if there exists some infinite sequence σ that extends ρ such that F holds of σ. A formula F is a safety property if F holds of an infinite sequence whenever it holds of all its finite prefixes.

The connective $\wedge$ and $\Rightarrow$ are the usual, classical ones; several interesting, additional connectives are definable in TLA [3]:

- $\mathcal{C}(F)$ holds of a sequence σ iff F holds of all finite prefixes of σ.
- $F \mathrel{-\!\triangleright} G$ holds of σ iff, for all (finite or infinite) prefixes ρ of σ, if F holds of ρ then so does G. Although $\mathrel{-\!\triangleright}$ is strictly stronger than $\Rightarrow$, $F \Rightarrow G$ and $F \mathrel{-\!\triangleright} G$ are equivalid.
- $F \overset{+}{\mathrel{-\!\triangleright}} G$ holds of $\sigma = \langle s_0, s_1, \ldots \rangle$ iff both:
  1. for all $n \geq 0$, if F holds of $\langle s_0, \ldots, s_{n-1} \rangle$, then G holds of $\langle s_0, \ldots, s_n \rangle$;
  2. if F holds of σ then G holds of σ.

It follows that $\mathcal{C}(F)$ denotes the strongest safety property implied by F; and F is equivalent to $\mathcal{C}(F)$ iff F is a safety property. If F and G are safety properties then $F \mathrel{-\!\triangleright} G$ and $F \overset{+}{\mathrel{-\!\triangleright}} G$ are safety properties too. For all F and G, we have:

$$F \mathrel{-\!\triangleright} G \;\equiv\; (\mathcal{C}(F) \mathrel{-\!\triangleright} \mathcal{C}(G)) \wedge (F \Rightarrow G) \tag{7}$$
$$F \overset{+}{\mathrel{-\!\triangleright}} G \;\equiv\; (\mathcal{C}(F) \overset{+}{\mathrel{-\!\triangleright}} \mathcal{C}(G)) \wedge (F \Rightarrow G) \tag{8}$$

Finite sequences yield a model of the abstract logic of section 2. Specifically, let Σ be the set of finite sequences of states, and $\sqsubseteq$ be the prefix order on Σ. Clearly, $\sqsubset$ is well-founded on Σ. To each formula F corresponds the set $\mathcal{M}(F)$ of finite sequences of which F holds; this is a downward-closed subset of Σ, and we may treat it as a proposition of the abstract logic. We have:

$$\mathcal{M}(F \mathrel{-\!\triangleright} G) \;=\; \mathcal{M}(F) \rightarrow \mathcal{M}(G) \tag{9}$$
$$\mathcal{M}(F \overset{+}{\mathrel{-\!\triangleright}} G) \;=\; \mathcal{M}(F) \overset{+}{\rightarrow} \mathcal{M}(G) \tag{10}$$

Furthermore, if F_i is a safety property for every $i \in I$, then:

$$\mathcal{M}(\bigwedge_{i\in I} F_i) \;=\; \bigwedge_{i\in I} \mathcal{M}(F_i) \tag{11}$$

The correspondence between the abstract logic and this model is close enough for our purposes, but not complete. In particular, there are downward-closed subsets of Σ that are not denoted by any TLA formula, for example the empty set. In addition, this model validates some formulas that are not intuitionistically valid, for example:

$$((P_1 \rightarrow P_2) \rightarrow Q) \wedge ((P_2 \rightarrow P_1) \rightarrow Q) \rightarrow Q$$

This formula is a disjunction-free version of the traditional formula $(P_1 \rightarrow P_2) \vee (P_2 \rightarrow P_1)$, which expresses a kind of linearity [10].

In the previous work on TLA, the composition of specifications is their conjunction, and refinement is implication. The assumption-commitment specification with assumption A and commitment C is either $A \mathrel{-\!\!\triangleright} C$ or $A \overset{+}{\mathrel{-\!\!\triangleright}} C$. When $A \mathrel{-\!\!\triangleright} C$ is chosen [1, 5], semantic conditions guarantee the equivalence of $A \mathrel{-\!\!\triangleright} C$ and $A \overset{+}{\mathrel{-\!\!\triangleright}} C$; therefore, we consider only $A \overset{+}{\mathrel{-\!\!\triangleright}} C$. In [2] there is a rule for proving that a conjunction of specifications, each of the form $A_i \overset{+}{\mathrel{-\!\!\triangleright}} C_i$, implies another specification $A \overset{+}{\mathrel{-\!\!\triangleright}} C$. The following result is a variation of that rule, restricted to safety properties:

Theorem 4. *If the TLA formulas A, C, A_i, C_i are safety properties (for $i \in I$), then the following formula is valid:*

$$(A \wedge \bigwedge_{i\in I} C_i \mathrel{-\!\!\triangleright} \bigwedge_{i\in I} A_i) \;\wedge\; (A \overset{+}{\mathrel{-\!\!\triangleright}} (\bigwedge_{i\in I} C_i \mathrel{-\!\!\triangleright} C)) \mathrel{-\!\!\triangleright} (\bigwedge_{i\in I}(A_i \overset{+}{\mathrel{-\!\!\triangleright}} C_i) \mathrel{-\!\!\triangleright} (A \overset{+}{\mathrel{-\!\!\triangleright}} C))$$

Proof. Since the formula is a safety property, it suffices to show that it is valid on finite sequences. Using (9), (10), and (11), we prove the validity of the sequent:

$$(A \wedge \bigwedge_{i\in I} C_i \rightarrow \bigwedge_{i\in I} A_i),\; (A \xrightarrow{+} (\bigwedge_{i\in I} C_i \rightarrow C)) \;\vdash\; \bigwedge_{i\in I}(A_i \xrightarrow{+} C_i) \rightarrow (A \xrightarrow{+} C)$$

1. Assume $(A \wedge \bigwedge_{i\in I} C_i \rightarrow \bigwedge_{i\in I} A_i)$ and $(A \xrightarrow{+} (\bigwedge_{i\in I} C_i \rightarrow C))$ and $\bigwedge_{i\in I}(A_i \xrightarrow{+} C_i)$.
2. $(\bigwedge_{i\in I} A_i \xrightarrow{+} \bigwedge_{i\in I} C_i) \rightarrow (A \xrightarrow{+} \bigwedge_{i\in I} C_i)$
   From step 1, which implies $A \rightarrow (\bigwedge_{i\in I} C_i \rightarrow \bigwedge_{i\in I} A_i)$, by Theorem 3(5).
3. $\bigwedge_{i\in I} A_i \xrightarrow{+} \bigwedge_{i\in I} C_i$
   From step 1 by Proposition 2(2).
4. $A \xrightarrow{+} \bigwedge_{i\in I} C_i$
   From steps 2 and 3.
5. $A \xrightarrow{+} C$
   From steps 1 (which says $A \xrightarrow{+} (\bigwedge_{i\in I} C_i \rightarrow C)$) and 4, by Proposition 2(1).

$\diamondsuit$

This proof shows that the abstract logic of section 2 accounts for the rule for composing specifications in the case of safety properties. Starting from Theorem 4, classical reasoning justifies a rule for arbitrary properties; the extra argument requires only the validity of $F \Rightarrow \mathcal{C}(F)$ and of the equivalences (7) and (8):

Theorem 5. *For any TLA formulas A, C, A_i, C_i (for $i \in I$), the following formula is valid:*

$$\left(\begin{array}{l} \mathcal{C}(A) \wedge \bigwedge_{i \in I} \mathcal{C}(C_i) \mathrel{-\!\!\triangleright} \bigwedge_{i \in I} A_i \\ \quad \wedge \\ A \wedge \bigwedge_{i \in I} C_i \mathrel{-\!\!\triangleright} C \\ \quad \wedge \\ \mathcal{C}(A) \mathrel{\stackrel{+}{-\!\!\triangleright}} (\bigwedge_{i \in I} \mathcal{C}(C_i) \mathrel{-\!\!\triangleright} \mathcal{C}(C)) \end{array}\right) \Rightarrow \ (\bigwedge_{i \in I} (A_i \mathrel{\stackrel{+}{-\!\!\triangleright}} C_i) \Rightarrow (A \mathrel{\stackrel{+}{-\!\!\triangleright}} C))$$

Theorem 5 yields a rule for composing specifications quite similar to that of Abadi and Lamport [2]. That work also develops techniques for establishing the hypotheses of the rule, for example techniques for proving formulas of the form $\mathcal{C}(A) \mathrel{\stackrel{+}{-\!\!\triangleright}} (\bigwedge_{i \in I} \mathcal{C}(C_i) \mathrel{-\!\!\triangleright} \mathcal{C}(C))$. Those techniques rely on TLA-specific ideas, outside the scope of our abstract logic. With this caveat, we believe that Theorems 4 and 5 reproduce the previous work faithfully and clarify its logical contents.

The same line of reasoning can be used to justify rules for other linear-time temporal logics, provided $\mathcal{C}$, $\mathrel{-\!\!\triangleright}$, and $\mathrel{\stackrel{+}{-\!\!\triangleright}}$ are definable. We treat branching-time logics in the next section.

4 Composition in CTL*

Next we apply the logic of section 2 to assumption-commitment specifications in the branching-time temporal logic CTL* [8]. This application is somewhat more tentative than that of section 3, in part because of the expressiveness of CTL*, which allows many different styles of assumption-commitment specifications. We restrict attention to the fragment of CTL* where formulas are invariant under finite stuttering [15]—specifically, we do not allow the next-time operator.

Formulas of branching-time temporal logics are normally interpreted over infinite trees. They include state formulas, which are evaluated at a state in a tree, and path formulas, which are evaluated on a path in a tree. We write $M, s \models F$ if the state formula F is true at state s in tree M.

We extend the semantics of state formulas to finite trees. We say that M is a subtree of N, and write $M \sqsubseteq N$, if M has the same root as N and M's accessibility relation is included in N's. For a finite tree M and a state s, we write $M, s \models F$ if s is not a node of M or if there exists some infinite tree N such that $M \sqsubseteq N$ and $N, s \models F$.

When s is the root of M, we may simply say that F is true of M, and write $M \models F$. A specification is given by a state formula F; it describes the set of trees N such that $N \models F$. A state formula F is a safety property if $N \models F$ whenever $M \models F$ for all finite subtrees $M \sqsubseteq N$.

As in the linear-time case, we have the connectives $\mathcal{C}$, $\mathrel{-\!\!\triangleright}$, and $\mathrel{\stackrel{+}{-\!\!\triangleright}}$:

- If F is a state formula, then $\mathcal{C}(F)$ is a state formula, with $N, s \models \mathcal{C}(F)$ iff $M, s \models F$ holds for all finite subtrees $M \sqsubseteq N$.
- If F and G are state formulas, then $F \mathrel{-\!\!\triangleright} G$ is a state formula, with $N, s \models F \mathrel{-\!\!\triangleright} G$ iff for all (finite and infinite) subtrees M of N, if $M, s \models F$ then $M, s \models G$.
- If F and G are state formulas, then $F \stackrel{+}{\mathrel{-\!\!\triangleright}} G$ is a state formula, with $N, s \models F \stackrel{+}{\mathrel{-\!\!\triangleright}} G$ iff both:
  1. for all finite subtrees $M \sqsubseteq N$, if $T, s \models F$ for all $T \sqsubset M$ then $M, s \models G$;
  2. $N, s \models F \mathrel{-\!\!\triangleright} G$.

Again, $\mathcal{C}(F)$ denotes the strongest safety property implied by F; and F is equivalent to $\mathcal{C}(F)$ iff F is a safety property. If F and G are safety properties then $F \mathrel{-\!\!\triangleright} G$ and $F \stackrel{+}{\mathrel{-\!\!\triangleright}} G$ are safety properties too. For all F and G, we have:

$$(F \mathrel{-\!\!\triangleright} G) \;\Rightarrow\; (\mathcal{C}(F) \mathrel{-\!\!\triangleright} \mathcal{C}(G)) \wedge (F \Rightarrow G) \tag{12}$$

$$F \stackrel{+}{\mathrel{-\!\!\triangleright}} G \;\equiv\; (\mathcal{C}(F) \stackrel{+}{\mathrel{-\!\!\triangleright}} \mathcal{C}(G)) \wedge (F \mathrel{-\!\!\triangleright} G) \tag{13}$$

Note the differences with the corresponding definitions and results for TLA. The differences arise because an infinite tree may have infinite proper subtrees, while the proper subsequences of an infinite sequence are all finite.

Finite trees yield another model of the abstract logic of section 2. Specifically, let Σ be the set of finite trees ordered by $\sqsubseteq$. Clearly, $\sqsubset$ is well-founded on Σ. To each state formula F corresponds the set $\mathcal{T}(F)$ of finite trees of which F is true; this is a downward-closed subset of Σ. For finite trees, we get the analogues of (9) and (10), with $\mathcal{M}$ replaced by $\mathcal{T}$. For safety properties, the analogue of (11) holds as well; this would not be true if we had allowed the next-time operator.

We represent the assumption-commitment specification with assumption A and commitment C by the formula $A \stackrel{+}{\mathrel{-\!\!\triangleright}} C$. We obtain the following theorem for specifications of the form $A \stackrel{+}{\mathrel{-\!\!\triangleright}} C$:

Theorem 6. *If the CTL* formulas A, C, A_i, C_i are safety properties and do not contain the next-time operator (for $i \in I$), then the following formula is valid:*

$$(A \wedge \bigwedge_{i \in I} C_i \mathrel{-\!\!\triangleright} \bigwedge_{i \in I} A_i) \wedge (A \stackrel{+}{\mathrel{-\!\!\triangleright}} (\bigwedge_{i \in I} C_i \mathrel{-\!\!\triangleright} C)) \mathrel{-\!\!\triangleright} (\bigwedge_{i \in I}(A_i \stackrel{+}{\mathrel{-\!\!\triangleright}} C_i) \mathrel{-\!\!\triangleright} (A \stackrel{+}{\mathrel{-\!\!\triangleright}} C))$$

The proof uses exactly the same reasoning as the corresponding proof for TLA. Going beyond safety properties, we can obtain a CTL* analogue for Theorem 5; the proof is basically the same as that of Theorem 5, and relies on the validity of $F \Rightarrow \mathcal{C}(F)$, the implication-like properties of $\mathrel{-\!\!\triangleright}$, and (12) and (13).

In branching-time temporal logics, the composition of modules does not in general implement the conjunction of the specifications of the modules. The application of our theorems as composition rules will therefore require additional arguments. This complication is not unique to our work; several authors [13, 9] have advocated restricting commitments to the fragment $\forall$CTL* in order to ensure that specifications are preserved by composition.

Josko [13, 14] has suggested representing an assumption-commitment specification as a pair (A, C) where A is a linear-time formula and C is a branching-time formula; Vardi [21] has studied the complexity of model-checking for specifications of this form. With Josko's definitions, a tree M satisfies a specification (A, C) with assumption A and commitment C iff $M' \models C$ where M' is the subtree of M that consists of those paths that satisfy A. Instead of Josko's (A, C), we can write $(\forall A) \overset{+}{\rhd} C$, which is similar but logically stronger. The similarity between (A, C) and $(\forall A) \overset{+}{\rhd} C$ is even closer under the substantial hypotheses of Josko's rules for dealing with mutual dependencies [14].

5 Conclusion

We have studied specifications in a general logical framework. We then inferred concrete proof rules for composing specifications from general logical facts. We believe that this approach explains some of the principles that underly the rules and helps in comparing rules.

Both of the applications described in detail in this paper are for temporal logics. However, our approach is not intrinsically limited to temporal logics: we have also used it on specifications of stream-processing functions [6, 19]. An assumption-commitment specification for a stream-processing function gives a property of the result of the function under assumptions about the inputs of the functions. Inductive reasoning arises when the function is defined as a fixpoint. We can represent that reasoning in our abstract logic, and thus prove variants of the proof rules of Stølen et al. [19]. We omit the details, which are long and perhaps not so natural.

Our exposition has been confined to the propositional level; we did not address the interplay of quantification and composition. In particular, existential quantification corresponds to hiding, which we have largely ignored. However, a general logical treatment of hiding may well be possible, and quite desirable. (Such a treatment was once started but not completed [4].)

References

1. Martín Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, January 1993.
2. Martín Abadi and Leslie Lamport. Conjoining specifications. Research Report 118, Digital Equipment Corporation, Systems Research Center, 1993. To appear in *ACM Transactions on Programming Languages and Systems*.
3. Martín Abadi and Stephan Merz. On TLA as a logic. In Manfred Broy, editor, *Deductive Program Design*, NATO ASI Series. Springer-Verlag, Berlin, 1995. To appear.
4. Martín Abadi and Gordon Plotkin. A logical view of composition and refinement. In *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Programming Languages*, pages 323–332. ACM, January 1991.
5. Martín Abadi and Gordon Plotkin. A logical view of composition. *Theoretical Computer Science*, 114(1):3–30, June 1993.

6. Manfred Broy. A functional rephrasing of the assumption/commitment specification style. SFB-Bericht 342/10/94, TUM-I9417, Techn. Univ. München, Munich, April 1994.
7. Antonio Cau and Pierre Collette. Parallel composition of assumption-commitment specifications: a unifying approach for shared variable and distributed message passing concurrency. *Acta Informatica*, 1995. To appear.
8. E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of theoretical computer science*, pages 997–1071. Elsevier Science Publishers B.V., 1990.
9. Orna Grumberg and David E. Long. Model checking and modular verification. *ACM Transactions on Programming Languages and Systems*, 16(3):843–871, May 1994.
10. Alfred Horn. Logic with truth values in a linearly ordered Heyting algebra. *Journal of Symbolic Logic*, 34(3):395–408, September 1969.
11. Cliff B. Jones. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems*, 5(4):596–619, October 1983.
12. Bengt Jonsson and Yih-Kuen Tsay. Assumption/guarantee specifications in linear-time temporal logic. In *Proceedings of TAPSOFT '95*, Lecture Notes in Computer Science, Berlin, May 1995. Springer-Verlag.
13. Bernhard Josko. Verifying the correctness of AADL modules using model checking. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, volume 430 of *Lecture Notes in Computer Science*, pages 386–400. Springer-Verlag, Berlin, 1989.
14. Bernhard Josko. Modular specification and verification of reactive systems. Habilitationsschrift, Univ. Oldenburg, Fachbereich Informatik, April 1993.
15. Leslie Lamport. What good is temporal logic? In R. E. A. Mason, editor, *Information Processing 83: Proceedings of the IFIP 9th World Congress*, pages 657–668, Paris, September 1983. IFIP, North-Holland.
16. Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
17. Jayadev Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Trans. Software Engineering*, 7(4):417–426, July 1981.
18. Amir Pnueli. In transition from global to modular temporal reasoning about programs. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, NATO ASI Series, pages 123–144. Springer-Verlag, October 1984.
19. Ketil Stølen, Frank Dederichs, and Rainer Weber. Assumption/commitment rules for networks of asynchronously communicating agents. SFB-Bericht 342/2/93, TUM-I9303, Techn. Univ. München, Munich, February 1993.
20. A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction*, volume 1. North Holland, Amsterdam, 1988.
21. Moshe Vardi. On the complexity of modular model checking. In *Proceedings of the Tenth Symposium on Logic in Computer Science*. IEEE, June 1995.
22. Qiwen Xu, Antonio Cau, and Pierre Collette. On unifying assumption-commitment style proof rules for concurrency. In Bengt Jonsson and Joachim Parrow, editors, *Proceedings of the Fifth International Conference on Concurrency Theory (CONCUR '94)*, volume 836 of *Lecture Notes in Computer Science*, pages 267–282, Berlin, 1994. Springer-Verlag.