Assume-Guarantee Reasoning for Safe Component Behaviours

Chris Chilton, Bengt Jonsson, Marta Kwiatkowska

Assume-Guarantee Reasoning for Safe Component Behaviours

Chris Chilton[1], Bengt Jonsson[2], and Marta Kwiatkowska[1]

[1] Department of Computer Science, University of Oxford, UK
[2] Department of Information Technology, Uppsala University, Sweden

Abstract. We formulate a sound and complete assume-guarantee framework for reasoning compositionally about safety properties of component behaviours. The specification of a component, which constrains the temporal ordering of input and output interactions with the environment, is expressed in terms of two prefix-closed sets of traces: an assumption and guarantee. The framework supports dynamic reasoning about components and specifications, and includes rules for parallel composition, logical conjunction corresponding to independent development, and quotient for incremental synthesis. Practical applicability of the framework is demonstrated by considering a simple printing example.

Keywords: assume-guarantee, specification theory, components, compositionality, parallel, conjunction, quotient.

1 Introduction

Component-based design methodologies enable both design- and run-time assembly of software systems from heterogeneous components, thus facilitating component reuse, incremental development and independent implementability. To improve the reliability and predictability of such systems, specification theories have been proposed that permit the mixing of specifications and implementations, and allow for the construction of new components from existing ones by means of compositional operators [1,2,3]. A specification should make explicit the *assumptions* that a component can make about the environment, and the corresponding *guarantees* that it will provide about its own behaviour. This allows for the use of compositional assume-guarantee (AG) reasoning, in order to combat issues of complexity and state space explosion during system development and verification.

In earlier work [4], we introduced a component-based specification theory, in which components communicate by synchronisation of I/O actions, with the understanding that inputs are controlled by the environment, while outputs (which are non-blocking) are under the control of the component. The component-model is conceptually similar to the interface automata of de Alfaro and Henzinger [5], except that our refinement is based on classical sets of traces, as opposed to alternating simulation, and that we allow explicit specification of inconsistent traces,

which can model underspecification and errors, etc. With both trace-based and operational representations for components, a distinguishing feature of our theory is the inclusion of conjunction and quotient operators (which generalise those of [2,6]) for supporting independent and incremental development, respectively. Logical disjunction and hiding can also be added. The theory enjoys strong algebraic properties with all the operators being compositional under refinement, and we prove full abstraction with respect to a simple testing framework.

In [4] and [5], the assumptions and guarantees of components are merged into one behavioural representation. In many cases, this avoids duplication of common information, although there are situations in which it is desirable to manipulate the assumptions and guarantees separately. For instance, we may want to express a simple guarantee (such as "no failure will occur") without having to weave it into a complex assumption. Another advantage of separation is specification reuse, in that the same guarantees (or assumptions) can be used for several related interfaces, each representing different versions of a component.

Contributions. In this paper, we present a complete specification theory for reasoning about AG specifications of components (as modelled in [4]). Assumptions and guarantees are prefix-closed sets of traces, meaning our framework facilitates reasoning about safety behaviours, and differs from (arguably) more complex approaches based on modal specifications and alternating simulation. Building upon the theory in [4], we define the operators of parallel, conjunction and quotient directly on AG specifications (the last being the first such definition), and prove their compositionality. By treating AG specifications as first-class citizens, the theory supports flexible development and verification of component-based systems using AG principles. A component can be characterised by its weakest AG specification, and, in the opposite direction, we can infer the least refined component satisfying a given specification. From this, a notion of refinement corresponding to implementation containment is defined. In relating implementations with AG specifications by means of satisfaction, we formulate a collection of sound and complete AG reasoning rules for the preservation of safety properties under the operations and refinement preorder of the specification theory. These rules are inspired by the Compositionality Principle of [7,8] for parallel composition, which we generalise to the operations of conjunction and quotient. The rules allow us to infer properties of compositions for both AG specifications and components, thus enabling designers to deduce whether it is safe to substitute a component, for example one synthesised at run-time by means of the quotient operator, with another.

Related work. Compositional AG reasoning has been extensively studied in the literature, where traditionally the work was concerned with compositional reasoning for processes, components and properties expressed in temporal logics [9,10,11]. A variety of rule formats have been proposed, although Maier demonstrates through a set-theoretic setting in [12] that compositional circular AG rules for parallel composition (corresponding to intersection) cannot both be sound and complete. This seems to contradict the work of Namjoshi and Tre-

fler [13], although the discrepancy is attributed to the fact that the sound and complete circular rule presented in [13] is non-compositional.

Compositional reasoning about AG specifications in the form of AG pairs, similar to what we consider in this paper, is discussed in [7] for the generic setting of state-based processes. The authors formulate a Compositionality Principle for parallel composition, and observe that this is sound for safety properties. A logical formulation for specifications is then discussed in [8], where intuitionistic and linear logic approaches are put forward. The main difference with our approach is that we consider an action-based component model and have a richer set of composition operators, including conjunction and quotient. We also prove completeness, by relying on the convention that an output is controlled by at most one component, which can be used to break circularity.

More recent proposals focus on compositional verification for interface theories [14,15], namely interface and I/O automata, which are closest to our work. In [14], Emmi *et al.* extend a learning-based compositional AG method to interface automata. They only consider the much more limited asymmetric rules for safety properties, which are shown to be both sound and complete. The rules are supplied for the original subset of operators and relations defined in [5], namely compatibility, parallel composition and refinement based on alternating simulation. Thus, no consideration is given to conjunction or quotient. Other notable work concerning compositional reasoning for interface theories is the AG framework defined by Larsen *et al.* in [15] for I/O automata, where assumptions and guarantees are themselves specified as I/O automata. The authors consider a parallel composition operator on AG specifications that is the weakest specification for composed components respecting independent implementability, for which they present a sound and complete rule. Our work allows a more general component model that does not require input-enabledness, and allows for specifications to have non-identical interfaces to their implementations. We go beyond [15] by defining conjunction and quotient operations directly on AG specifications, thus providing a significantly richer basis for AG based reasoning and development, and we do not require input-enabledness of guarantees.

A compositional specification theory based on modal specifications has been developed in [3], which includes all the operations we consider in this paper, but for systems without I/O distinction. Larsen *et al.* consider a cross between modal specifications and interface automata [1], where refinement is given in terms of alternating simulation/modal refinement (which is stronger than our trace containment), and no operations for conjunction and quotient are given. Surveying [16], Bauer *et al.* provide a generic construction for obtaining a contract framework based on AG pairs from a component-based specification theory. The abstract ideas share similarity with our framework, and it is interesting to note how parallel composition of contracts is defined in terms of the conjunction and quotient operators of the specification theory. Our work differs in that we define both of these operators directly on contracts. A definition of conjunction on contracts is provided in [17], but this is for a simplified contract framework, as witnessed by the definition of parallel composition on contracts.

Outline. In Section 2 we summarise the compositional specification theory of [4], which serves as a basis for our AG reasoning framework. Section 3 introduces the main definitions of the AG framework, and presents a number of sound and complete compositional rules for the operators of the specification theory. An application of our framework is illustrated in Section 4, while Section 5 concludes our work and suggests possible extensions. Proofs of our results are made available as the technical report [18].

2 Compositional Specification Theory

In this section, we briefly survey the essential features of our compositional specification theory presented in [4]. In that paper, we present two notations for modelling components: a trace-based formalism and an operational representation. Here we focus on the trace-based models, since operational models can be mapped to semantically equivalent trace-based ones.

A component comes equipped with an interface, together with a set of behaviours over the interface. The interface is represented by a set of input actions and a set of output actions, which are necessarily disjoint, while the behaviour is characterised by sets of traces.

Definition 1 (Components). *A* component $\mathcal{P}$ *is a tuple* $\langle \mathcal{A}^I_\mathcal{P}, \mathcal{A}^O_\mathcal{P}, T_\mathcal{P}, F_\mathcal{P} \rangle$ *in which* $\mathcal{A}^I_\mathcal{P}$ *and* $\mathcal{A}^O_\mathcal{P}$ *are disjoint sets referred to as inputs and outputs respectively (the union of which is denoted by* $\mathcal{A}_\mathcal{P}$*),* $T_\mathcal{P} \subseteq \mathcal{A}^*_\mathcal{P}$ *is a non-empty set of permissible traces, and* $F_\mathcal{P} \subseteq \mathcal{A}^*_\mathcal{P}$ *is a set of inconsistent traces. The trace sets must satisfy the constraints:*

1. $F_\mathcal{P} \subseteq T_\mathcal{P}$
2. *If* $t \in T_\mathcal{P}$ *and* $i \in \mathcal{A}^I_\mathcal{P}$*, then* $ti \in T_\mathcal{P}$
3. $T_\mathcal{P}$ *is prefix closed*
4. *If* $t \in F_\mathcal{P}$ *and* $t' \in \mathcal{A}^*_\mathcal{P}$*, then* $tt' \in F_\mathcal{P}$*.*

The permissible traces contain all possible interaction sequences between the component and the environment; they are thus receptive to all inputs, as these are under the control of the environment. If on some interaction sequence an error arises in the component, or the environment issues a non-enabled input, the trace is said to be inconsistent. We adopt the convention that any inconsistent trace is suffix closed, meaning that, once the component becomes inconsistent, it behaves similarly to the process CHAOS in CSP.

From hereon let $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$ be components with signatures $\langle \mathcal{A}^I_\mathcal{P}, \mathcal{A}^O_\mathcal{P}, T_\mathcal{P}, F_\mathcal{P} \rangle$, $\langle \mathcal{A}^I_\mathcal{Q}, \mathcal{A}^O_\mathcal{Q}, T_\mathcal{Q}, F_\mathcal{Q} \rangle$ and $\langle \mathcal{A}^I_\mathcal{R}, \mathcal{A}^O_\mathcal{R}, T_\mathcal{R}, F_\mathcal{R} \rangle$ respectively.

Notation. Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be sets of actions. For a trace t, write $t \restriction \mathcal{A}$ for the projection of t onto $\mathcal{A}$. Now for $T \subseteq \mathcal{A}^*$, write $T \restriction \mathcal{B}$ for $\{t \restriction \mathcal{B} : t \in T\}$, $T \Uparrow \mathcal{B}$ for $\{t \in \mathcal{B}^* : t \restriction \mathcal{A} \in T\}$, $T \Uparrow\!\Uparrow \mathcal{B}$ for $\epsilon + (T \Uparrow \mathcal{B})(\epsilon + \mathcal{A}^I)$, $T \uparrow \mathcal{B}$ for $T(\mathcal{B})(\mathcal{A} \cup \mathcal{B})^*$, $T \uparrow_\epsilon \mathcal{B}$ for $T \cup (T \uparrow \mathcal{B})$, $\overline{T}$ for $\mathcal{A}^* \setminus T$, and $\mathsf{pre}(T)$ for the largest prefix-closed set contained in T.

Refinement. In the specification theory, refinement corresponds to safe-substitutivity. This means that $\mathcal{Q}$ is a refinement of $\mathcal{P}$ if $\mathcal{Q}$ can be used safely in any environment that is safe for $\mathcal{P}$. An environment is safe for a component if any interaction between the two cannot be extended by a sequence of output actions under the control of the component such that the resulting trace is inconsistent. We will thus need to consider the safe representation of a component, obtained by propagating inconsistencies backwards over outputs.

Definition 2 (Safe component). *Let $\mathcal{P}$ be a component. The most general safe representation for $\mathcal{P}$ is a component $\mathcal{E}(\mathcal{P}) = \langle \mathcal{A}^I_\mathcal{P}, \mathcal{A}^O_\mathcal{P}, T_{\mathcal{E}(\mathcal{P})}, F_{\mathcal{E}(\mathcal{P})} \rangle$, where $T_{\mathcal{E}(\mathcal{P})} = T_\mathcal{P} \cup F_{\mathcal{E}(\mathcal{P})}$ and $F_{\mathcal{E}(\mathcal{P})} = \{tt' \in \mathcal{A}^*_\mathcal{P} : t \in T_\mathcal{P}$ and $\exists t'' \in (\mathcal{A}^O_\mathcal{P})^* \cdot tt'' \in F_\mathcal{P}\}$.*

We can now give the formal definition of refinement. Intuitively, $\mathcal{Q}$ must be willing to accept any input that $\mathcal{P}$ can accept, but it must produce no more outputs than $\mathcal{P}$, otherwise we could not be certain how the environment would respond to these additional outputs.

Definition 3 (Refinement). *For components $\mathcal{P}$ and $\mathcal{Q}$, $\mathcal{Q}$ is said to be a refinement of $\mathcal{P}$, written $\mathcal{Q} \sqsubseteq_{imp} \mathcal{P}$, iff:*

1. $\mathcal{A}^I_\mathcal{P} \subseteq \mathcal{A}^I_\mathcal{Q}$
2. $\mathcal{A}^O_\mathcal{Q} \subseteq \mathcal{A}^O_\mathcal{P}$
3. $T_{\mathcal{E}(\mathcal{Q})} \subseteq T_{\mathcal{E}(\mathcal{P})} \cup T_{\mathcal{E}(\mathcal{P})} \uparrow (\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P})$
4. $F_{\mathcal{E}(\mathcal{Q})} \subseteq F_{\mathcal{E}(\mathcal{P})} \cup T_{\mathcal{E}(\mathcal{P})} \uparrow (\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P})$.

The set $T_{\mathcal{E}(\mathcal{P})} \uparrow (\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P})$ represents the extension of $\mathcal{P}$'s interface to include all inputs in $\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P}$. As these inputs are not ordinarily accepted by $\mathcal{P}$, they are treated as bad inputs, hence the suffix closure with arbitrary behaviour.

Parallel composition. The parallel composition of two components is obtained as the cross-product by synchronising on common actions and interleaving on independent actions. To support broadcasting, we make the assumption that inputs and outputs synchronise to produce outputs. Communication mismatches arising through non-input enabledness automatically appear as inconsistent traces in the product, on account of our component formulation. As the outputs of a component are controlled locally, we assume that the output actions of the components to be composed are disjoint.

Definition 4 (Parallel composition). *Let $\mathcal{P}$ and $\mathcal{Q}$ be components such that $\mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^O_\mathcal{Q} = \emptyset$. Then $\mathcal{P} \,||\, \mathcal{Q}$ is the component $\langle \mathcal{A}^I_{\mathcal{P}||\mathcal{Q}}, \mathcal{A}^O_{\mathcal{P}||\mathcal{Q}}, T_{\mathcal{P}||\mathcal{Q}}, F_{\mathcal{P}||\mathcal{Q}} \rangle$, where:*

- $\mathcal{A}^I_{\mathcal{P}||\mathcal{Q}} = (\mathcal{A}^I_\mathcal{P} \cup \mathcal{A}^I_\mathcal{Q}) \setminus (\mathcal{A}^O_\mathcal{P} \cup \mathcal{A}^O_\mathcal{Q})$
- $\mathcal{A}^O_{\mathcal{P}||\mathcal{Q}} = \mathcal{A}^O_\mathcal{P} \cup \mathcal{A}^O_\mathcal{Q}$
- $T_{\mathcal{P}||\mathcal{Q}} = [(T_\mathcal{P} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}}) \cap (T_\mathcal{Q} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}})] \cup F_{\mathcal{P}||\mathcal{Q}}$
- $F_{\mathcal{P}||\mathcal{Q}} = [(T_\mathcal{P} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}}) \cap (F_\mathcal{Q} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}})]\mathcal{A}^*_{\mathcal{P}||\mathcal{Q}} \cup$
  $[(F_\mathcal{P} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}}) \cap (T_\mathcal{Q} \Uparrow \mathcal{A}_{\mathcal{P}||\mathcal{Q}})]\mathcal{A}^*_{\mathcal{P}||\mathcal{Q}}$.

Informally, a trace is permissible in $\mathcal{P} \parallel \mathcal{Q}$ if its projection onto $\mathcal{A}_\mathcal{P}$ is a trace of $\mathcal{P}$ and its projection onto $\mathcal{A}_\mathcal{Q}$ is a trace of $\mathcal{Q}$. A trace is inconsistent if it has a prefix whose projection onto the alphabet of one of the components is inconsistent and the projection onto the alphabet of the other component is a permissible trace of that component.

Conjunction. The conjunction of components $\mathcal{P}$ and $\mathcal{Q}$ is the coarsest component that will work safely in any environment that $\mathcal{P}$ or $\mathcal{Q}$ can work safely in. It can be thought of as finding a common implementation for a number of specifications. Thus, conjunction is essentially the meet operator on the refinement preorder. Consequently, the conjunction of two components is only defined when the union of their inputs is disjoint from the union of their outputs.

Definition 5 (Conjunction). *Let $\mathcal{P}$ and $\mathcal{Q}$ be components such that $\mathcal{A}^I_\mathcal{P} \cup \mathcal{A}^I_\mathcal{Q}$ and $\mathcal{A}^O_\mathcal{P} \cup \mathcal{A}^O_\mathcal{Q}$ are disjoint. Then $\mathcal{P} \wedge \mathcal{Q}$ is the component $\langle \mathcal{A}^I_{\mathcal{P}\wedge\mathcal{Q}}, \mathcal{A}^O_{\mathcal{P}\wedge\mathcal{Q}}, T_{\mathcal{P}\wedge\mathcal{Q}}, F_{\mathcal{P}\wedge\mathcal{Q}}\rangle$, where:*

- $\mathcal{A}^I_{\mathcal{P}\wedge\mathcal{Q}} = \mathcal{A}^I_\mathcal{P} \cup \mathcal{A}^I_\mathcal{Q}$
- $\mathcal{A}^O_{\mathcal{P}\wedge\mathcal{Q}} = \mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^O_\mathcal{Q}$
- $T_{\mathcal{P}\wedge\mathcal{Q}} = [(T_\mathcal{P} \cup T_\mathcal{P} \uparrow (\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P})) \cap (T_\mathcal{Q} \cup T_\mathcal{Q} \uparrow (\mathcal{A}^I_\mathcal{P} \setminus \mathcal{A}^I_\mathcal{Q}))] \cap \mathcal{A}^*_{\mathcal{P}\wedge\mathcal{Q}}$
- $F_{\mathcal{P}\wedge\mathcal{Q}} = [(F_\mathcal{P} \cup T_\mathcal{P} \uparrow (\mathcal{A}^I_\mathcal{Q} \setminus \mathcal{A}^I_\mathcal{P})) \cap (F_\mathcal{Q} \cup T_\mathcal{Q} \uparrow (\mathcal{A}^I_\mathcal{P} \setminus \mathcal{A}^I_\mathcal{Q}))] \cap \mathcal{A}^*_{\mathcal{P}\wedge\mathcal{Q}}$.

Intuitively, after any trace of $\mathcal{P} \wedge \mathcal{Q}$, the conjunction must accept any input offered by either $\mathcal{P}$ or $\mathcal{Q}$, but can only issue an output if both $\mathcal{P}$ and $\mathcal{Q}$ are willing to offer it. Once $\mathcal{P}$ becomes inconsistent, or an input is seen that is not an input of $\mathcal{P}$, the conjunction behaves like $\mathcal{Q}$ (and vice-versa).

Quotient. In [4], we introduced a quotient operator acting on components. Given a component $\mathcal{R}$, together with a component $\mathcal{P}$ implementing part of $\mathcal{R}$, the quotient $\mathcal{R}/\mathcal{P}$ yields the coarsest component for the remaining part of $\mathcal{R}$ to be implemented. Thus, the quotient satisfies the property: there exists $\mathcal{Q}$ such that $\mathcal{P} \parallel \mathcal{Q} \sqsubseteq_{imp} \mathcal{R}$ iff $\mathcal{P} \parallel (\mathcal{R}/\mathcal{P}) \sqsubseteq_{imp} \mathcal{R}$ and $\mathcal{Q} \sqsubseteq_{imp} (\mathcal{R}/\mathcal{P})$. Whether the quotient exists depends on the extent to which $\mathcal{P}$ is a sub-component of $\mathcal{R}$.

For the development in this paper, we will not use quotient on components, and refer to [4]. Instead, we will define a quotient operator that acts on AG specifications. Thus, the quotient of two AG specifications yields an AG specification characterising a set of component implementations.

3 Assume-Guarantee Framework for Safety Properties

To support reasoning about components, we introduce the concept of an AG specification, which consists of two prefix-closed sets of traces referred to as the *assumption* and *guarantee*. The assumption specifies the environment's allowable interaction sequences, while the guarantee is a constraint on the component's

behaviour. As assumptions and guarantees are prefix-closed, our theory ensures that components preserve (not necessarily regular) safety properties[3].

Definition 6 (AG specification). *An* AG specification $\mathcal{S}$ *is a tuple* $\langle \mathcal{A}^I_\mathcal{S}, \mathcal{A}^O_\mathcal{S}, \mathcal{R}_\mathcal{S}, \mathcal{G}_\mathcal{S} \rangle$*, in which* $\mathcal{A}^I_\mathcal{S}$ *and* $\mathcal{A}^O_\mathcal{S}$ *are disjoint sets, referred to as the inputs and outputs respectively, and* $\mathcal{R}_\mathcal{S}$ *and* $\mathcal{G}_\mathcal{S}$ *are prefix closed subsets of* $(\mathcal{A}^I_\mathcal{S} \cup \mathcal{A}^O_\mathcal{S})^*$*, referred to as the assumption and guarantee respectively, such that* $t \in \mathcal{R}_\mathcal{S}$ *and* $t' \in (\mathcal{A}^O_\mathcal{S})^*$ *implies* $tt' \in \mathcal{R}_\mathcal{S}$*.*

Since outputs are under the control of a component, we insist that assumptions are closed under output-extensions. On the other hand, we need not insist that the guarantee is closed under input-extensions, since the assumption can select inputs under which the guarantee is given.

Given an AG specification $\mathcal{S}$, we want to be able to say whether a component $\mathcal{P}$ satisfies $\mathcal{S}$. Informally, $\mathcal{P}$ satisfies $\mathcal{S}$ if for any interaction between $\mathcal{P}$ and the environment characterised by a trace t, if $t \in \mathcal{R}_\mathcal{S}$, then $t \in \mathcal{G}_\mathcal{S}$ and t cannot become inconsistent in $\mathcal{P}$ without further stimulation from the environment. Components can thus be thought of as implementations of AG specifications.

Before defining satisfaction, we need to introduce a notion of compatibility between AG specifications and components, meaning that they do not disagree on what are inputs or outputs.

Definition 7 (Compatibility). *Let* $\mathcal{P}$ *be a component, and let* $\mathcal{S}$ *and* $\mathcal{T}$ *be AG-specifications. Then* $\mathcal{P}$ *is compatible with* $\mathcal{S}$*, written* $\mathcal{P} \sim \mathcal{S}$*, iff* $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{S} = \emptyset = \mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^I_\mathcal{S}$*. Similarly,* $\mathcal{S}$ *is compatible with* $\mathcal{T}$*, written* $\mathcal{S} \sim \mathcal{T}$*, iff* $\mathcal{A}^I_\mathcal{S} \cap \mathcal{A}^O_\mathcal{T} = \emptyset = \mathcal{A}^O_\mathcal{S} \cap \mathcal{A}^I_\mathcal{T}$*.*

We can now give the formal definition for satisfaction of an AG specification by a component.

Definition 8 (AG satisfaction). *A component* $\mathcal{P}$ *satisfies the AG specification* $\mathcal{S}$*, written* $\mathcal{P} \models \mathcal{S}$*, iff:*

S1. $\mathcal{P} \sim \mathcal{S}$
S2. $\mathcal{A}^I_\mathcal{S} \subseteq \mathcal{A}^I_\mathcal{P}$
S3. $\mathcal{A}^O_\mathcal{P} \subseteq \mathcal{A}^O_\mathcal{S}$
S4. $\mathcal{R}_\mathcal{S} \cap T_\mathcal{P} \subseteq \mathcal{G}_\mathcal{S} \cap \overline{F_\mathcal{P}}$*.*

By output-extension closure of assumptions, condition S4 is equivalent to checking $\mathcal{R}_\mathcal{S} \cap T_\mathcal{P} \subseteq \mathcal{G}_\mathcal{S} \cap \overline{F_{\mathcal{E}(\mathcal{P})}}$, which involves the most general safe representation $\mathcal{E}(\mathcal{P})$ of $\mathcal{P}$ (see Definition 2). The following lemma shows that this definition of satisfaction is preserved under the component-based refinement corresponding to safe-substitutivity, subject to compatibility.

Lemma 1. *Let* $\mathcal{P}$ *and* $\mathcal{Q}$ *be components, and let* $\mathcal{S}$ *be an AG specification. If* $\mathcal{P} \models \mathcal{S}$*,* $\mathcal{Q} \sqsubseteq_{imp} \mathcal{P}$ *and* $\mathcal{Q} \sim \mathcal{S}$*, then* $\mathcal{Q} \models \mathcal{S}$*.*

[3] Model-checking components against AG specifications would force us to restrict the properties we can encode and check. In this setting, we would naturally restrict to the regular safety properties, which can be encoded by finite-state automata.

3.1 Refinement

There is a natural hierarchy on AG specifications respecting the satisfaction rule defined in Definition 8. From this we can define a refinement relation on AG specifications that corresponds to implementation containment. But first, we introduce the shorthand: $\mathsf{violations}(X) \triangleq \{t \in \mathcal{A}_X^* : t(\mathcal{A}_X^I)^* \in \mathcal{R}_X \cap \overline{\mathcal{G}_X}\}\mathcal{A}_X^*$.

Definition 9 (AG refinement). *Let $\mathcal{S}$ and $\mathcal{T}$ be AG specifications. $\mathcal{S}$ is said to be a refinement of $\mathcal{T}$, written $\mathcal{S} \sqsubseteq \mathcal{T}$, iff:*

R1. $\mathcal{S} \sim \mathcal{T}$
R2. $\mathcal{A}_{\mathcal{T}}^I \subseteq \mathcal{A}_{\mathcal{S}}^I$
R3. $\mathcal{A}_{\mathcal{S}}^O \subseteq \mathcal{A}_{\mathcal{T}}^O$
R4. $\mathsf{violations}(\mathcal{T}) \cap \mathcal{A}_{\mathcal{S}}^* \subseteq \mathsf{violations}(\mathcal{S})$
R5. $\mathcal{R}_{\mathcal{T}} \cap \mathcal{A}_{\mathcal{S}}^* \subseteq \mathcal{R}_{\mathcal{S}} \cup \mathsf{violations}(\mathcal{S})$.

It is our intention that $\mathcal{S} \sqsubseteq \mathcal{T}$ iff the implementations of $\mathcal{S}$ are contained within the implementations of $\mathcal{T}$ (subject to compatibility). Conditions R1-R3 are the bare minimum to uphold this principle. For condition R4, any component having a trace $t \in \mathsf{violations}(\mathcal{T}) \cap \mathcal{A}_{\mathcal{S}}^*$ cannot be an implementation of $\mathcal{T}$, so it should not be an implementation of $\mathcal{S}$. For this to be the case, the component must violate the guarantee on $\mathcal{S}$, i.e., $t \in \mathsf{violations}(\mathcal{S})$. Condition R5 deals with inconsistent traces. If a component has an inconsistent trace $t \in \mathcal{R}_{\mathcal{T}} \cap \mathcal{A}_{\mathcal{S}}^*$, then this cannot be an implementation of $\mathcal{T}$. Consequently, the component must not be an implementation of $\mathcal{S}$, so either t must violate the guarantee of $\mathcal{S}$, i.e., $t \in \mathsf{violations}(\mathcal{S})$, or t must be in $\mathcal{R}_{\mathcal{S}}$, so that the component cannot satisfy $\mathcal{S}$.

Lemma 2. *Refinement respects implementation containment:*

$$\mathcal{S} \sqsubseteq \mathcal{T} \iff \{\mathcal{P} : \mathcal{P} \models \mathcal{S} \text{ and } \mathcal{P} \sim \mathcal{T}\} \subseteq \{\mathcal{P} : \mathcal{P} \models \mathcal{T}\}.$$

In [15], Larsen *et al.* give a sound and complete characterisation of their refinement relation (which corresponds to implementation containment, as for us) by means of conformance tests. The definition assumes equality of interfaces, so does not need to deal with issues of compatibility or the complexities of both covariant and contravariant inclusion of inputs and outputs respectively (i.e., conditions R1-R3). Thus, their definition largely corresponds to condition R4. Condition R5 is not necessary in that setting, as implementation models are required to be input-enabled.

Refinement can be shown to be a preorder, provided that we add the minor technical condition that compatibility of components is maintained, as the next lemma shows.

Lemma 3 (Weak transitivity). *For AG specifications $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{U}$, if $\mathcal{S} \sqsubseteq \mathcal{T}$, $\mathcal{T} \sqsubseteq \mathcal{U}$ and $\mathcal{S} \sim \mathcal{U}$, then $\mathcal{S} \sqsubseteq \mathcal{U}$.*

As an aside, component-based refinement $\sqsubseteq_{imp}$ is a preorder because, in refining a component $\mathcal{P}$ to a component $\mathcal{Q}$, it is possible to transform some of $\mathcal{P}$'s outputs into inputs of $\mathcal{Q}$, as this preserves safe-substitutivity. However, this transformation of action types does not make sense with AG specifications, which talk explicitly about the behaviour of the environment.

3.2 Inferring Components from AG Specifications

Given a specification for a component, we require a way for developers to construct an actual component that satisfies the requirements of the specification. In the following definition, we show how to infer the least refined component that satisfies a given specification.

Definition 10 (Inferred component). *Let $\mathcal{S}$ be an AG specification. Then the least refined implementation of $\mathcal{S}$ is the component $\mathcal{I}(\mathcal{S}) = \langle \mathcal{A}^I_{\mathcal{S}}, \mathcal{A}^O_{\mathcal{S}}, T_{\mathcal{I}(\mathcal{S})}, F_{\mathcal{I}(\mathcal{S})} \rangle$, defined only when $\epsilon \in T_{\mathcal{I}(\mathcal{S})}$, where:*

- $T_{\mathcal{I}(\mathcal{S})} = \mathsf{pre}(\{t \in \mathcal{R}_{\mathcal{S}} \cap \mathcal{G}_{\mathcal{S}} : \forall t' \in (\mathcal{A}^I_{\mathcal{S}})^* \cdot tt' \in \overline{\mathcal{R}_{\mathcal{S}}} \cup \mathcal{G}_{\mathcal{S}}\}) \cup F_{\mathcal{I}(\mathcal{S})}$
- $F_{\mathcal{I}(\mathcal{S})} = \{tit' : t \in \mathcal{R}_{\mathcal{S}} \cap \mathcal{G}_{\mathcal{S}},\ i \in \mathcal{A}^I_{\mathcal{S}} \text{ and } ti \notin \mathcal{R}_{\mathcal{S}}\} \cup \{t \in \mathcal{A}^*_{\mathcal{S}} : \epsilon \notin \mathcal{R}_{\mathcal{S}}\}$.

The following lemma shows that the obtained component model really is least refined with respect to the refinement preorder $\sqsubseteq_{imp}$ on implementations.

Lemma 4. *Let $\mathcal{S}$ be an AG specification, and let $\mathcal{P}$ be a component. Then:*

- *$\epsilon \notin T_{\mathcal{I}(\mathcal{S})}$ implies $\mathcal{S}$ is non-implementable;*
- *$\epsilon \in T_{\mathcal{I}(\mathcal{S})}$ implies $\mathcal{I}(\mathcal{S}) \models \mathcal{S}$; and*
- *$\mathcal{P} \models \mathcal{S}$ iff $\mathcal{P} \sqsubseteq_{imp} \mathcal{I}(\mathcal{S})$.*

3.3 Characteristic AG Specification of a Component

One may be interested in the most general AG specification that satisfies a component, which we refer to as the characteristic AG specification of the component. This can be found by examining the component's safe traces.

Definition 11 (Characteristic AG specification). *The* characteristic AG specification *for the component $\mathcal{P}$ is an AG specification $\mathcal{AG}(\mathcal{P}) = \langle \mathcal{A}^I_{\mathcal{P}}, \mathcal{A}^O_{\mathcal{P}}, \mathcal{R}_{\mathcal{AG}(\mathcal{P})}, \mathcal{G}_{\mathcal{AG}(\mathcal{P})} \rangle$, where $\mathcal{R}_{\mathcal{AG}(\mathcal{P})} = \mathcal{A}^*_{\mathcal{P}} \setminus F_{\mathcal{E}(\mathcal{P})}$ and $\mathcal{G}_{\mathcal{AG}(\mathcal{P})} = T_{\mathcal{P}} \setminus F_{\mathcal{E}(\mathcal{P})}$.*

The largest assumption safe for component $\mathcal{P}$ is the set of all non-inconsistent traces, while the guarantee is the set of traces of $\mathcal{E}(\mathcal{P})$ that are non-inconsistent. As the following lemma demonstrates, the characteristic AG specification satisfies the desired properties.

Lemma 5. *Let $\mathcal{P}$ be a component and let $\mathcal{S}$ be an AG specification. Then:*

- *$\mathcal{P} \models \mathcal{AG}(\mathcal{P})$; and*
- *$\mathcal{P} \models \mathcal{S}$ iff $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}$.*

The final point in the previous lemma shows that satisfaction of a specification by a component is equivalent to checking whether the characteristic AG specification of the component is a refinement of the specification. This means that implementability of specifications built up compositionally follows immediately from compositionality results on AG specifications, as we will see in the subsequent sections.

We are now in a position to present sound and complete AG rules for inferring properties of composite systems from the properties of their sub-components.

3.4 Parallel Composition

The AG rule for parallel composition is based on the well-established theorem of Abadi and Lamport [7], which has appeared in several forms [19,20,21]. Intuitively, the guarantee of any component must not be allowed to violate the assumptions of the other components. Such reasoning seems circular, but the circularity can be broken up in our setting as a safety property cannot be simultaneously violated by two or more components. This is due to an output being under the control of at most one component.

Notation. To assist in our definition, we introduce the following shorthands:

- $\mathcal{R}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \triangleq (\mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}) \cap (\mathcal{R}_{\mathcal{S}_\mathcal{Q}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})$
- $\mathcal{G}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \triangleq (\mathcal{G}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}) \cap (\mathcal{G}_{\mathcal{S}_\mathcal{Q}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})$
- $\mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \triangleq (\mathcal{G}_{\mathcal{S}_\mathcal{P}} \Uparrow\Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}) \cap (\mathcal{G}_{\mathcal{S}_\mathcal{Q}} \Uparrow\Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})$.

Definition 12. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be AG specifications such that $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} = \emptyset$. If $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ are both implementable, then $\mathcal{S}_\mathcal{P} \,||\, \mathcal{S}_\mathcal{Q}$ is an AG specification $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \rangle$ defined by:*

- $\mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = (\mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}) \setminus (\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}})$
- $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$
- $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \subseteq \mathcal{A}^*_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ *is the largest prefix closed set satisfying* $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}(\mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})^* \cap \mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \subseteq \mathcal{R}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$
- $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{G}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$.

If at least one of $\mathcal{S}_\mathcal{P}$ or $\mathcal{S}_\mathcal{Q}$ is non-implementable, then $\mathcal{S}_\mathcal{P} \,||\, \mathcal{S}_\mathcal{Q} = \langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{A}^*_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \emptyset \rangle$

$\mathcal{S}_\mathcal{P} \,||\, \mathcal{S}_\mathcal{Q}$ yields the strongest specification satisfiable by the parallel composition of any two components that satisfy $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$. The specification only guarantees what can be assured by both $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$, thus it is the strongest composition. The assumption is the largest collection of environmental behaviours that cannot violate either of the guarantees $\mathcal{G}_{\mathcal{S}_\mathcal{P}}$ or $\mathcal{G}_{\mathcal{S}_\mathcal{Q}}$, and moreover does not permit a component implementing one of the specifications to violate the other specification's assumption. Ignoring differences in alphabets, this can loosely be phrased as $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{G}_{\mathcal{S}_\mathcal{P}} \subseteq \mathcal{R}_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{G}_{\mathcal{S}_\mathcal{Q}} \subseteq \mathcal{R}_{\mathcal{S}_\mathcal{P}}$, which is akin to the presentation in [7]. However, as implementations are not required to be input-enabled, this must be reformulated as $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \subseteq \mathcal{R}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$.

The set $\mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$ extends $\mathcal{G}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$ by a single input on each of $\mathcal{G}_{\mathcal{S}_\mathcal{P}}$ and $\mathcal{G}_{\mathcal{S}_\mathcal{Q}}$, and also includes ϵ. This has the effect of ensuring that, if $t \in \mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q}) \cap \mathcal{R}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$ and $ta \notin \mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{Q})$, then whatever the action type of a, wlog $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}}}$ or $ta \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}}}$. Thus, any implementation of $\mathcal{S}_\mathcal{P}$ must have suppressed an output at some stage along the trace $ta \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}}$, implying the parallel composition of any two implementations of $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ will suppress an output along ta. Thus, $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ contains only traces within $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ and traces not reachable by any pair of implementations of $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$.

Subject to suitable constraints on the alphabets of AG specifications, it can be shown that the parallel composition operator on AG specifications is compositional under the AG refinement relation, as the following theorem demonstrates.

Theorem 1. *Let $\mathcal{S}_\mathcal{P}$, $\mathcal{S}'_\mathcal{P}$, $\mathcal{S}_\mathcal{Q}$ and $\mathcal{S}'_\mathcal{Q}$ be AG specifications such that $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} = \emptyset$, $\mathcal{S}'_\mathcal{P} \parallel \mathcal{S}'_\mathcal{Q} \sim \mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q}$, $\mathcal{A}^I_{\mathcal{S}'_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}'_\mathcal{Q}} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$, $\mathcal{A}^O_{\mathcal{S}'_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}'_\mathcal{Q}} \subseteq \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{A}^I_{\mathcal{S}'_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}'_\mathcal{Q}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{P} \| \mathcal{S}_\mathcal{Q}} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$. If $\mathcal{S}'_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$, then $\mathcal{S}'_\mathcal{P} \parallel \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q}$.*

The condition $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} = \emptyset$ ensures that the parallel composition of the AG specifications is defined, while $\mathcal{S}'_\mathcal{P} \parallel \mathcal{S}'_\mathcal{Q} \sim \mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q}$ means $\mathcal{S}'_\mathcal{P} \parallel \mathcal{S}'_\mathcal{Q}$ and $\mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q}$ are comparable under refinement. The remaining three conditions are standard for compositionality of parallel composition. From this compositionality result, it is easy to give a sound and complete AG rule.

Theorem 2. *Let $\mathcal{P}$ and $\mathcal{Q}$ be components, and let $\mathcal{S}_\mathcal{P}$, $\mathcal{S}_\mathcal{Q}$ and $\mathcal{S}$ be AG specifications such that $\mathcal{P} \parallel \mathcal{Q} \sim \mathcal{S}$, $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{Q} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$, $\mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^I_\mathcal{Q} \subseteq \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^I_\mathcal{Q} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{P} \| \mathcal{S}_\mathcal{Q}} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$. Then the following AG rule is both sound and complete:*

$$\textsc{Parallel}\ \frac{\mathcal{P} \models \mathcal{S}_\mathcal{P} \qquad \mathcal{Q} \models \mathcal{S}_\mathcal{Q} \qquad \mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}}{\mathcal{P} \parallel \mathcal{Q} \models \mathcal{S}}.$$

3.5 Conjunction

In this section we define a conjunctive operator on AG specifications for combining independently developed requirements. From this we show that the operator is both compositional and corresponds to the meet operation on the refinement relation. This allows us to formulate a sound and complete AG rule.

The conjunction of AG specifications $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ is only defined when $\mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$ is disjoint from $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$, in which case we say $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ are composable. The composability constraint is necessary, as otherwise it is not possible to find an interface that can refine both $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$.

Definition 13. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be AG specifications composable for conjunction. Then $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ is an AG specification $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}, \mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}, \mathcal{G}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \rangle$ defined by:*

- $\mathcal{A}^I_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} = \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$
- $\mathcal{A}^O_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} = \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$
- $\mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} = (\mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathcal{R}_{\mathcal{S}_\mathcal{Q}}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$
- $\mathcal{G}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ *is the intersection of the following sets:*
  - $\mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \cap (\mathcal{G}_{\mathcal{S}_\mathcal{P}} \cup \mathcal{G}_{\mathcal{S}_\mathcal{Q}})$
  - $\mathsf{pre}(\overline{\mathcal{R}_{\mathcal{S}_\mathcal{P}}} \cup \mathcal{G}_{\mathcal{S}_\mathcal{P}}) \uparrow_\epsilon (\mathcal{A}^I_{\mathcal{S}_\mathcal{Q}} \setminus \mathcal{A}^I_{\mathcal{S}_\mathcal{P}})$
  - $\mathsf{pre}(\overline{\mathcal{R}_{\mathcal{S}_\mathcal{Q}}} \cup \mathcal{G}_{\mathcal{S}_\mathcal{Q}}) \uparrow_\epsilon (\mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \setminus \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}})$.

The assumption $\mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ is constrained to be within at least one of $\mathcal{R}_{\mathcal{S}_\mathcal{P}}$ or $\mathcal{R}_{\mathcal{S}_\mathcal{Q}}$. On the other hand, the guarantee $\mathcal{G}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ must be within at least one of $\mathcal{G}_{\mathcal{S}_\mathcal{P}}$ or $\mathcal{G}_{\mathcal{S}_\mathcal{Q}}$, and must ensure that, if the assumption for one of the specifications is satisfied, then the corresponding guarantee cannot have been violated.

The next two theorems show that our definition of conjunction corresponds to the meet operator on the refinement relation, and is compositional under refinement. Consequently, the set of implementations for $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ is the intersection of the implementation sets for $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$.

Theorem 3. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be AG specifications such that $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ are composable for conjunction. Then:*

- $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P}$
- $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$
- $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P}$ *and* $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{Q}$ *implies* $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$.

Theorem 4. *Let $\mathcal{S}_\mathcal{P}$, $\mathcal{S}_\mathcal{Q}$, $\mathcal{S}'_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q}$ be AG specifications such that $\mathcal{S}'_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q}$ are composable for conjunction, $\mathcal{S}'_\mathcal{P} \sim \mathcal{S}_\mathcal{Q}$ and $\mathcal{S}'_\mathcal{Q} \sim \mathcal{S}_\mathcal{P}$. If $\mathcal{S}'_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$, then $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$.*

From these strong algebraic properties, we can formulate an AG rule for conjunction that is both sound and complete.

Theorem 5. *Let $\mathcal{P}$ and $\mathcal{Q}$ be components composable for conjunction, and let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be AG specifications such that $\mathcal{P} \sim \mathcal{S}_\mathcal{Q}$, $\mathcal{Q} \sim \mathcal{S}_\mathcal{P}$ and $\mathcal{P} \wedge \mathcal{Q} \sim \mathcal{S}$. Then the following AG rule is both sound and complete:*

$$\textsc{Conjunction}\ \frac{\mathcal{P} \models \mathcal{S}_\mathcal{P} \qquad \mathcal{Q} \models \mathcal{S}_\mathcal{Q} \qquad \mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}}{\mathcal{P} \wedge \mathcal{Q} \models \mathcal{S}}.$$

3.6 Quotient

The AG rule for parallel composition in Theorem 2 makes use of the composition $\mathcal{S}_\mathcal{P} \,||\, \mathcal{S}_\mathcal{Q}$. To support incremental development, we need a way of decomposing the composition to find $\mathcal{S}_\mathcal{Q}$ given $\mathcal{S}_\mathcal{P}$. We can do this using a quotient operator.

Definition 14. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be AG specifications. Then the quotient $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is an AG specification $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}, \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}, \mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \rangle$, defined only when $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \subseteq \mathcal{A}^O_{\mathcal{S}_\mathcal{W}}$, where $\mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \mathcal{A}^I_{\mathcal{S}_\mathcal{W}} \setminus \mathcal{A}^I_{\mathcal{S}_\mathcal{P}}$, $\mathcal{A}^O_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \mathcal{A}^O_{\mathcal{S}_\mathcal{W}} \setminus \mathcal{A}^O_{\mathcal{S}_\mathcal{P}}$ and:*

- *If $\mathcal{S}_\mathcal{P}$ is implementable, and $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ implies $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{P}}$, then:*
  - $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = [\mathcal{R}_{\mathcal{S}_\mathcal{W}} \cap (\mathcal{G}_{\mathcal{S}_\mathcal{P}} \uparrow\uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}})(\mathcal{A}^O_{\mathcal{S}_\mathcal{W}})^*] \restriction \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$
  - $\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap (X \restriction \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}})$, *where X is the largest prefix closed set satisfying* $X(\mathcal{A}^I_{\mathcal{S}_\mathcal{P}})^* \cap \mathcal{R}_{\mathcal{S}_\mathcal{W}} \subseteq \mathsf{pre}(\mathcal{G}_{\mathcal{S}_\mathcal{W}} \cup \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}}) \cap \mathsf{pre}((\mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}) \cup \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}} \uparrow\uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}})$.*
- *If $\mathcal{S}_\mathcal{P}$ is implementable and $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{W}} \cap \overline{\mathcal{R}_{\mathcal{S}_\mathcal{P}}}$, then $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \mathcal{A}^*_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ and $\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \emptyset$.*
- *If $\mathcal{S}_\mathcal{P}$ is non-implementable, then $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = \emptyset$.*

Although not immediately obvious, the assumption in the previous definition is closed under output-extensions. Before explaining the definition, we introduce the following theorem, which shows that the quotient operator on AG specifications yields the weakest decomposition of the parallel composition.

Theorem 6. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be AG specifications. Then there exists an AG specification $\mathcal{S}_\mathcal{Q}$ such that $\mathcal{S}_\mathcal{P} \,||\, \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$ iff the following properties hold:*

- *The quotient $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is defined*
- *$\mathcal{S}_\mathcal{P} \,||\, (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}) \sqsubseteq \mathcal{S}_\mathcal{W}$*
- *$\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$.*

To make sense of the definition for quotient (in the difficult case of $\mathcal{S}_\mathcal{P}$ being implementable and $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ implies $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{P}}$), it is necessary to consider the final two results in Theorem 6. For these, we need to show that: (i) $\mathcal{R}_{\mathcal{S}_\mathcal{W}} \subseteq \mathcal{R}_{\mathcal{S}_\mathcal{P}||(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})}$; and (ii) $\mathcal{R}_{\mathcal{S}_\mathcal{W}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{W}}} \subseteq \mathsf{violations}(\mathcal{S}_\mathcal{P} \,||\, (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$. Clause (i) amounts to showing $\mathcal{R}_{\mathcal{S}_\mathcal{W}} \cap \mathcal{G}^+(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}) \subseteq \mathcal{R}(\mathcal{S}_\mathcal{P}, \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$, i.e., the condition for parallel composition. Thus, the assumption $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ is the smallest output-closed set such that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ and $t \in \mathcal{G}_{\mathcal{S}_\mathcal{P}} \upuparrows \mathcal{A}_{\mathcal{S}_\mathcal{W}}$ implies $t \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}$. The cases of $t \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}$ or $t \notin \mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \upuparrows \mathcal{A}_{\mathcal{S}_\mathcal{W}}$ are handled by $\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$.

Considering the guarantee $\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$, it is obvious that it need only be contained within the assumption $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$. Moreover, it is safe to have $t \in \mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \upuparrows \mathcal{A}_{\mathcal{S}_\mathcal{W}}$ if $t \notin \mathcal{G}_{\mathcal{S}_\mathcal{P}} \upuparrows \mathcal{A}_{\mathcal{S}_\mathcal{W}}$ or $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow_{\mathcal{A}_{\mathcal{S}_\mathcal{P}}} \mathcal{A}_{\mathcal{S}_\mathcal{W}}$; this is equivalent to requiring $t \in \mathsf{pre}((\mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}) \cup \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}} \upuparrows \mathcal{A}_{\mathcal{S}_\mathcal{W}}})$. For requirement (ii), if $t \in \mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}$, then it must be the case that $t \notin \mathcal{G}_{\mathcal{S}_\mathcal{W}}$ implies $t \notin \mathcal{G}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}$. This is equivalent to requiring $t \in \mathsf{pre}(\mathcal{G}_{\mathcal{S}_\mathcal{W}} \cup \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{W}}})$. Piecing these conditions together yields a definition of quotient that is correct by construction.

Theorem 7. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be AG specifications such that $\mathcal{P}$ ranges over components having the same interface as $\mathcal{S}_\mathcal{P}$, and $\mathcal{Q}$ is a component having the same interface as $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$. If $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is defined (i.e., $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \subseteq \mathcal{A}^O_{\mathcal{S}_\mathcal{W}}$), then the following AG rule is sound and complete:*

$$\textsc{Quotient}\ \frac{\forall \mathcal{P} \cdot \mathcal{P} \models \mathcal{S}_\mathcal{P} \text{ implies } \mathcal{P} \,||\, \mathcal{Q} \models \mathcal{S}_\mathcal{W}}{\mathcal{Q} \models \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}.$$

The restriction on $\mathcal{P}$ and $\mathcal{S}_\mathcal{P}$ having the same interface, and $\mathcal{Q}$ and $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ having the same interface, is necessary, because the parallel operator is only compositional under certain restrictions on the interfaces (cf Theorem 1).

3.7 Decomposing Parallel Composition

The following corollary shows how we can revise the AG rule for parallel composition so that it makes use of quotient on AG specifications when we know the global specification $\mathcal{S}$. This is useful for system development, as we will often have the specification of a global system, rather than the specifications of the systems to be composed.

Corollary 1. *Let $\mathcal{P}$ and $\mathcal{Q}$ be components such that $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^I_\mathcal{Q} = \emptyset$, and let $\mathcal{S}_\mathcal{P}$, $\mathcal{S}_\mathcal{Q}$ and $\mathcal{S}$ be AG specifications. If $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} = \emptyset$, $\mathcal{P} \parallel \mathcal{Q} \sim \mathcal{S}$, $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{Q} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^I_\mathcal{Q} \subseteq \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$, then the following rule is both sound and complete:*

$$\textsc{Parallel-Decompose}\frac{\mathcal{P} \models \mathcal{S}_\mathcal{P} \qquad \mathcal{Q} \models \mathcal{S}_\mathcal{Q} \qquad \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}/\mathcal{S}_\mathcal{P}}{\mathcal{P} \parallel \mathcal{Q} \models \mathcal{S}}.$$

This rule, based on Theorem 2, differs in having the premise $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}/\mathcal{S}_\mathcal{P}$ in place of $\mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}$. This substitution is permitted by the results of Theorem 6. The condition $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^I_\mathcal{Q} = \emptyset$ is necessary in order to show that $\mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \parallel (\mathcal{S}/\mathcal{S}_\mathcal{P})$, given the constraints on parallel compositionality, and the fact that $\mathcal{A}^I_{\mathcal{S}_\mathcal{P}}$ and $\mathcal{A}^I_{\mathcal{S}/\mathcal{S}_\mathcal{P}}$ are always disjoint.

4 A Printing Example

We illustrate our assume-guarantee framework on a simple example of component-based design for a system concerned with printing a document. The system as a whole is composed of a job scheduler, a printer controller and the physical printer itself. Intuitively, the scheduler decides when a print job can *start*, and expects to be informed when the job has $finish$ed. The controller, on the other hand, waits for the *start* signal from the scheduler, after which it instructs the printer to *print* the document, and awaits confirmation from the printer that the document has *printed*. At this stage, the controller will signal to the scheduler that the job has $finish$ed. The printer accepts a *print* command, after which it will start to print the document, and will signify when the document is *printed*.

We iteratively derive a design by successively applying AG rules and constructions. We start by making use of two specifications for the combined effect of the scheduler and printer controller:

1. Spec1: If the number of jobs sent to *print* is equal to or one greater than the number of jobs *printed*, then the number of job *start*s must be equal to or one greater than the number of requests sent to *print*.
2. Spec2: If the number of jobs sent to *print* is equal to or one greater than the number of jobs *printed*, then the number of *printed* documents must be equal to or one greater than the number of jobs $finish$ed.

Spec1 and Spec2 can be represented by the AG specifications $\langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec1}} \rangle$ and $\langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec2}} \rangle$ respectively, where the assumptions and guarantees are depicted in Figure 1. For simplicity, we represent sets of traces by means of finite automata, and annotate states with an F to indicate that a trace becomes inconsistent. The combined effect of Spec1 and Spec2 is given by the conjunctive specification $\mathsf{Spec1} \wedge \mathsf{Spec2} = \langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec1} \wedge \mathsf{Spec2}} \rangle$, the guarantee of which is presented in Figure 2.

To demonstrate compositional AG reasoning, by Definition 10 we can find implementations $\mathcal{I}(\mathsf{Spec1})$ and $\mathcal{I}(\mathsf{Spec2})$ of Spec1 and Spec2 respectively, which

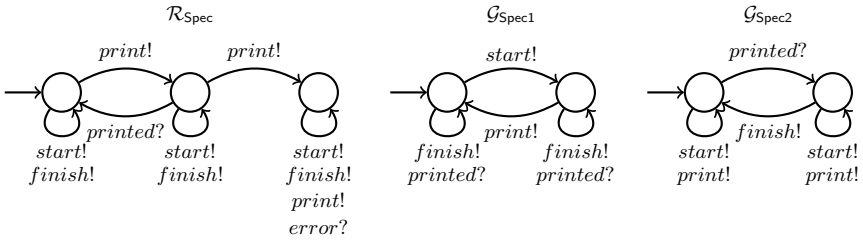

Fig. 1. Assumption and guarantees for Spec1 and Spec2.

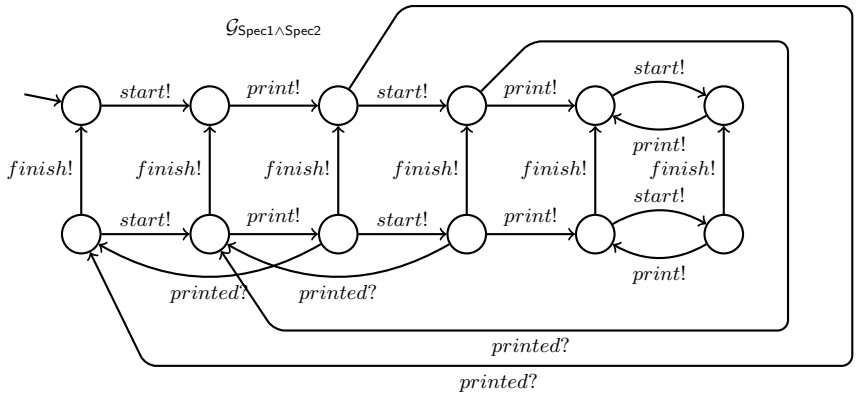

Fig. 2. The guarantee for Spec1 ∧ Spec2.

by Theorem 5 allows us to derive $\mathcal{I}(\mathsf{Spec1}) \wedge \mathcal{I}(\mathsf{Spec2}) \models \mathsf{Spec1} \wedge \mathsf{Spec2}$. According to Lemma 4, this means that $\mathcal{I}(\mathsf{Spec1}) \wedge \mathcal{I}(\mathsf{Spec2}) \sqsubseteq_{imp} \mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2})$. Now by Theorem 3, we know $\mathsf{Spec1} \wedge \mathsf{Spec2} \sqsubseteq \mathsf{Spec1}$, so from Lemma 2 we obtain $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2}) \models \mathsf{Spec1}$, and from Lemma 4 we derive $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2}) \sqsubseteq_{imp} \mathcal{I}(\mathsf{Spec1})$. By similar reasoning it can be shown that $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2}) \sqsubseteq_{imp} \mathcal{I}(\mathsf{Spec2})$, hence by Theorem 2 of [4] we acquire $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2}) \sqsubseteq_{imp} \mathcal{I}(\mathsf{Spec1}) \wedge \mathcal{I}(\mathsf{Spec2})$. Mutual refinement of components in our framework corresponds to equality of models, so $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2}) = \mathcal{I}(\mathsf{Spec1}) \wedge \mathcal{I}(\mathsf{Spec2})$. Such an implementation is shown in Figure 3. Note how this component is unwilling to *print* after encountering two *start* requests not separated by a job being *printed*. This is because $\mathcal{R}_{\mathsf{Spec}}$ can issue an *error* after such an occurrence, but this is not accepted by $\mathcal{G}_{\mathsf{Spec1}\wedge\mathsf{Spec2}}$. Moreover, this implementation is able to *start* and *print* an unbounded number of jobs without ever having to *finish* one of them.

We now propose an alternative derivation based on quotient, by making use of a constraint specification $\mathsf{Sched} = \langle \mathcal{R}_{\mathsf{Sched}}, \mathcal{G}_{\mathsf{Sched}} \rangle$ that requires *start* and *finish* to alternate (shown in Figure 4). We wish to find an implementation for the printer controller, let it be called Controller, such that Controller is an implementation of Spec1 ∧ Spec2 subject to the constraints imposed by Sched. This is equivalent to requiring $\mathsf{Controller} \models (\mathsf{Spec1} \wedge \mathsf{Spec2})/\mathsf{Sched}$. The specification

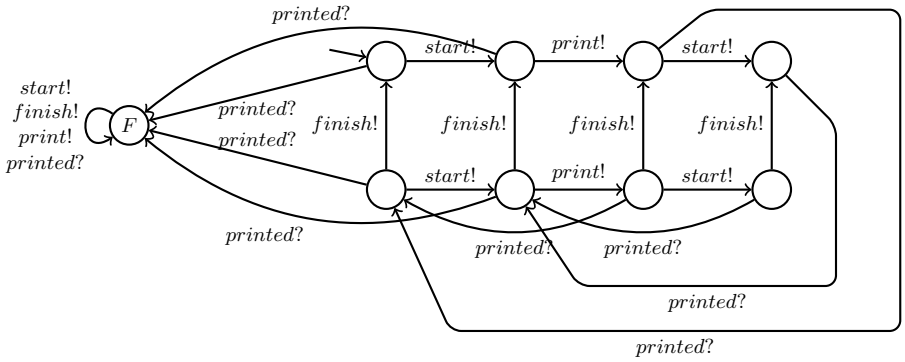

Fig. 3. The most general implementation of Spec1 ∧ Spec2.

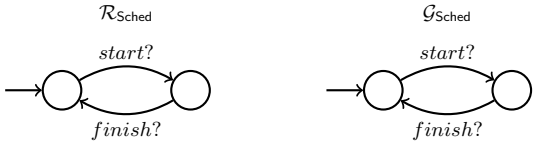

Fig. 4. Specification of a scheduling constraint Sched.

(Spec1 ∧ Spec2)/Sched is exhibited in Figure 5, and the most general implementation is obtained from $\mathcal{G}_{(\mathsf{Spec1}\wedge\mathsf{Spec2})/\mathsf{Sched}}$ by appending all non-enabled inputs as inconsistent traces. In contrast to $\mathcal{I}(\mathsf{Spec1} \wedge \mathsf{Spec2})$, the constraints imposed by Sched on Spec1 ∧ Spec2 means that any candidate implementation for Controller will ensure that there can be at most one outstanding job that has not $finish$ed.

5 Conclusion

We have presented a complete specification theory for reasoning about safety properties of component behaviours with an explicit separation of assumptions from guarantees. Our theory supports refinement based on traces, which relates specifications by implementation containment. We define compositional operations of parallel composition, as well as – for the first time in this setting – conjunction and quotient, directly on AG specifications. We give sound and complete AG reasoning rules for the three operators, which preserve safety and enable the reasoning about, e.g., safe substitutivity of components synthesised at run-time. The theory can be extended with disjunction and hiding, as well as liveness through the introduction of quiescence. The AG rules can also be fully automated, as they are based on simple set-theoretic operations and do not require the learning of assumptions. The refinement is linear-time, and hence amenable to automata-theoretic approaches.

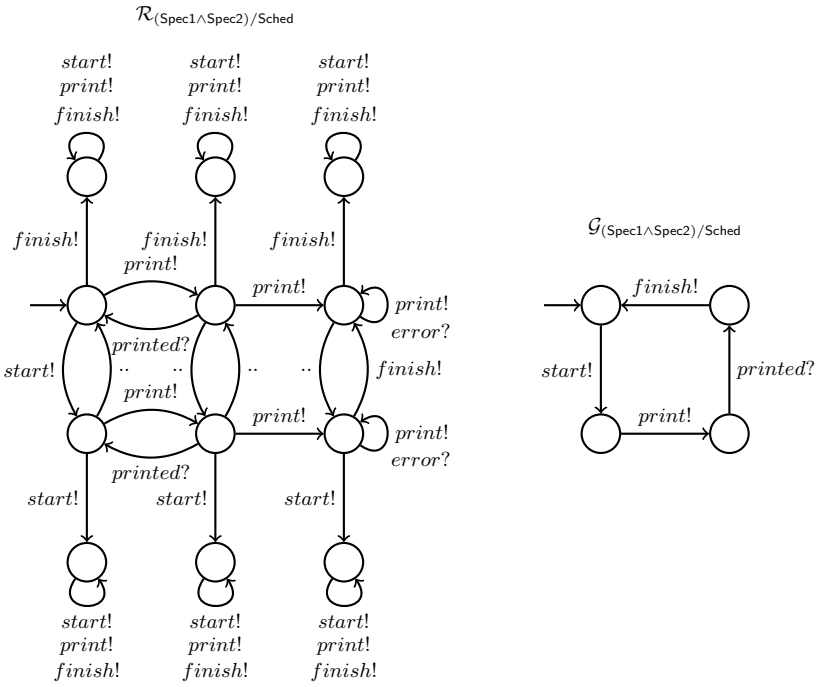

Fig. 5. Specification for $(\mathsf{Spec1} \wedge \mathsf{Spec2})/\mathsf{Sched}$.

Acknowledgments. The authors are supported by EU FP7 project CONNECT and ERC Advanced Grant VERIWARE.

References

1. Larsen, K.G., Nyman, U., Wasowski, A.: Modal I/O automata for interface and product line theories. In Nicola, R.D., ed.: ESOP. Volume 4421 of LNCS., Springer (2007) 64–79
2. Doyen, L., Henzinger, T.A., Jobstmann, B., Petrov, T.: Interface theories with component reuse. In: Proc. 8th ACM international conference on Embedded software. EMSOFT '08, ACM (2008) 79–88
3. Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. Inform. **108** (2011) 119–149
4. Chen, T., Chilton, C., Jonsson, B., Kwiatkowska, M.: A Compositional Specification Theory for Component Behaviours. In Seidl, H., ed.: ESOP'12. Volume 7211 of LNCS., Springer (2012) 145–165
5. de Alfaro, L., Henzinger, T.A.: Interface automata. SIGSOFT Softw. Eng. Notes **26** (2001) 109–120
6. Bhaduri, P., Ramesh, S.: Interface synthesis and protocol conversion. Form. Asp. Comput. **20** (2008) 205–224

7. Abadi, M., Lamport, L.: Composing specifications. ACM Transactions on Programming Languages and Systems **15** (1993) 73–132
8. Abadi, M., Plotkin, G.: A logical view of composition. Theoretical Computer Science **114** (1993) 3–30
9. Pnueli, A.: Logics and models of concurrent systems. Springer (1985) 123–144
10. Clarke, E., Long, D., McMillan, K.: Compositional model checking. In: Proc. 4th Annual Symposium on Logic in computer science, IEEE Press (1989) 353–362
11. Grumberg, O., Long, D.E.: Model checking and modular verification. ACM Transactions on Programming Languages and Systems **16** (1991)
12. Maier, P.: A set-theoretic framework for assume-guarantee reasoning. In Orejas, F., Spirakis, P.G., Leeuwen, J., eds.: ICALP'01, LNCS 2076. (2001) 821–834
13. Namjoshi, K.S., Trefler, R.J.: On the completeness of compositional reasoning methods. ACM Trans. Comput. Logic **11** (2010) 16:1–16:22
14. Emmi, M., Giannakopoulou, D., Păsăreanu, C.: Assume-Guarantee Verification for Interface Automata. In Cuellar, J., Maibaum, T., Sere, K., eds.: FM 2008: Formal Methods. Volume 5014 of LNCS. Springer (2008) 116–131
15. Larsen, K.G., Nyman, U., Wasowski, A.: Interface input/output automata. In: FM 2006. Volume 4085 of LNCS., Springer (2006) 82–97
16. Bauer, S., David, A., Hennicker, R., Larsen, K., Legay, A., Nyman, U., Wasowski, A.: Moving from specifications to contracts in component-based design. In Lara, J., Zisman, A., eds.: FASE'12. Volume 7212 of LNCS. Springer (2012) 43–58
17. Delahaye, B., Caillaud, B., Legay, A.: Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects. FMSD **38** (2011) 1–32
18. Chilton, C., Jonsson, B., Kwiatkowska, M.: Assume-Guarantee Reasoning for Safe Component Behaviours. Technical Report CS-RR-12-07, Department of Computer Science, University of Oxford (2012)
19. Collette, P.: Application of the composition principle to Unity-like specifications. In: TAPSOFT'93, LNCS 668, Springer-Verlag (1993) 230–242
20. Abadi, M., Lamport, L.: Conjoining specifications. ACM Transactions on Programming Languages and Systems **17** (1995) 507–534
21. Jonsson, B., Tsay, Y.K.: Assumption/guarantee specifications in linear-time temporal logic. Theoretical Computer Science **167** (1996) 47–72