Compositional Assume-Guarantee Reasoning for Input/Output Component Theories

Chris Chilton[a], Bengt Jonsson[b], Marta Kwiatkowska[a]

[a] *Department of Computer Science, University of Oxford, UK*
[b] *Department of Information Technology, Uppsala University, Sweden*

Abstract

We formulate a sound and complete assume-guarantee framework for reasoning compositionally about components modelled as a variant of interface automata. The specification of a component, which expresses both safety and progress properties of input and output interactions with the environment, is characterised by finite traces. The framework supports dynamic reasoning about components and specifications, and includes rules for parallel composition, logical conjunction and disjunction corresponding to independent development, and quotient for incremental synthesis. Practical applicability of the framework is demonstrated through a link layer protocol case study.

Keywords: interfaces, assume-guarantee, contracts, safety, liveness, quiescence, compositionality, components, interface automata, refinement, substitutivity, conjunction, disjunction, quotient

1. Introduction

Component-based methodologies enable both design- and runtime assembly of software systems from heterogeneous components, 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 (Benveniste et al., 2008; Larsen et al., 2007; Doyen et al., 2008; Raclet et al., 2011). 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 behaviour. This allows for the use of compositional assume-guarantee (AG) reasoning, which enables the decomposition of the system into smaller components, each of which may be reasoned about in isolation during system development and verification.

In earlier work (Chen et al., 2012), we introduced a component-based specification theory, in which components communicate by synchronisation of input/output (I/O) actions, where inputs are controlled by the environment, while outputs (which are non-blocking) are controlled by the component. The component model is conceptually similar to the interface automata of de Alfaro and Henzinger (2001), except that we use a different underlying semantic model, which is based on classical sets of traces, rather than alternating simulation. This approach allows us to define the weakest refinement preorder that preserves substitutivity of components, which implies a full abstraction result. Distinguishing features of our theory, an extension of which is contained in (Chilton, 2013; Chilton et al., 2013a), are the inclusion of conjunction and quotient operators (which are more general than those of Doyen et al. (2008); Bhaduri and Ramesh (2008)), as well as logical disjunction and hiding, in addition to a progress-sensitive variant of refinement based on quiescence, whereby a refining component must make progress whenever the original can. The theory enjoys strong algebraic properties, with all the operators being compositional under refinement.

In (Chilton, 2013; de Alfaro and Henzinger, 2001), the assumptions and guarantees of components are merged into one behavioural representation. In many cases, this avoids duplication of common information, although it can be 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. Separation of assumptions from guarantees also supports 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 article, we present a specification theory for reasoning about AG specifications (or contracts) of components as modelled in (Chilton, 2013; Chilton et al., 2013a). The formalism is well suited to modelling components of distributed systems, such as communication protocols and mediators, to name but a few. A contract consists of an assumption, guarantee and liveness property, all of which are represented by sets of finite traces. This facilitates reasoning about safety and progress properties, and differs from (arguably) more complex approaches based on modal specifications and alternating simulation. Treating contracts as first-class citizens, we define the operators of parallel, conjunction, disjunction and quotient on contracts, and prove compositionality. This is the first work to present such an extensive collection of operators directly on contracts (to our knowledge, quotient has not previously been defined), which supports flexible development and verification of component-based systems using AG principles. In relating implementations (components) with contracts by means of satisfaction, a notion of refinement corresponding to implementation containment is defined on contracts. Based on this, we formulate a collection of sound and complete AG reasoning rules for the preservation of safety and progress properties under the operations and refinement preorder of the specification theory. The AG rule for parallel is inspired by the Compositionality Principle of Abadi and Lamport (1993); Abadi and Plotkin (1993), while the others admit novel treatment. The rules allow us to infer properties of compositions for both contracts and components, thus enabling designers to deduce whether it is safe to substitute a component, for example one synthesised at runtime by means of the quotient operator, with another. A preliminary version of this paper appeared as (Chilton et al., 2013b).

Related work. Compositional AG reasoning has been extensively studied in the literature. Traditionally, the work was concerned with compositional reasoning for processes, components and properties expressed in temporal logics (Pnueli, 1985; Clarke et al., 1989; Grumberg and Long, 1991). A variety of rule formats have been proposed, although Maier (2003) demonstrates through a set-theoretic setting that compositional circular AG rules (where compositionality is defined in a precise way) for parallel composition (corresponding to intersection) cannot both be sound and complete. In Namjoshi and Trefler (2010), a sound and complete circular rule is presented, which is non-compositional. We obtain soundness and completeness of our compositional rule by relying on the fact that the outputs of components to be composed are disjoint, which breaks circularity.

Abadi and Lamport (1993) consider compositional reasoning for contracts in the generic setting of state-based processes. They formulate a Compositionality Principle for parallel, which is sound for safety properties. A logical formulation of specifications is discussed by Abadi and Plotkin (1993), where intuitionistic and linear logic approaches are adopted. In contrast, our work considers an action-based component model and has a richer set of composition operators, including conjunction and quotient. Furthermore, we prove completeness, as remarked in the previous paragraph.

More recent proposals focus on compositional verification for component theories such as interface and I/O automata. Emmi et al. (2008) extend a learning-based compositional AG method to interface automata. Sound and complete rules are presented for the original operators defined by de Alfaro and Henzinger (2001), namely compatibility, parallel and refinement based on alternating simulation, but conjunction, disjunction and quotient are absent. Moreover, the rules are limited to being asymmetric in nature. Larsen et al. (2006) define an AG framework for I/O automata, where assumptions and guarantees are themselves specified as I/O automata. A parallel operator is defined on contracts, yielding the weakest specification respecting independent implementability, for which a sound and complete rule is presented. Our work differs by not requiring input-enabledness of components or guarantees, and allowing for specifications to have non-identical interfaces to their implementations. We also define conjunction, disjunction and quotient, and support progress properties, thus providing a significantly richer reasoning framework.

Raclet et al. (2011) have developed a compositional theory based on modal specifications, which includes the operations we consider in this article, but for systems without I/O distinction. Larsen et al. (2007) consider a cross between modal specifications and interface automata, where refinement is given in terms of alternating simulation/modal refinement (stronger than our trace containment), but conjunction and quotient are not defined. Both of these works use single models to encode assumptions and guarantees, whereas we adopt a contract-based approach.

Benveniste et al. (2008) present an abstract mathematical framework for contract-based design, based on set-theoretic operations on sets of behaviours. The framework does not give consideration to the specifics of the execution model, hence it is unclear whether the rules can be instantiated for any particular communication model.

Bauer et al. (2012) provide a generic construction for obtaining a contract framework 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. Delahaye et al. (2011) define conjunction on contracts, but this is for a simplified contract framework, as witnessed by the definition of parallel composition on contracts.

Outline. A summary of the compositional specification theory on which our AG reasoning framework is based is provided in Section 2. Section 3 introduces the AG framework for both safety and progress properties, and presents a number of sound and complete rules for the operators of the specification theory. An application of our framework to a case study involving a link layer protocol is demonstrated in Section 4, while Section 5 concludes and suggests future work.

2. Compositional Specification Theory

In this section, we briefly review the essential features of our compositional specification theory presented in (Chilton, 2013; Chilton et al., 2013a), an extended version of (Chen et al., 2012). The framework comprises two notations for modelling components: a trace-based formalism and an operational representation. Here we focus on the trace-based models, since the semantics of operational models can be defined in terms of sets of traces. For simplicity, it is assumed that components cannot diverge. A trace-based component comes equipped with an interface, together with a collection of behaviours characterised by three sets of traces.

Definition 1 (Component). *A* component $\mathcal{P}$ *is a tuple* $\langle \mathcal{A}^I_\mathcal{P}, \mathcal{A}^O_\mathcal{P}, T_\mathcal{P}, F_\mathcal{P}, K_\mathcal{P} \rangle$ *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* $\mathcal{A}_\mathcal{P}$*), and* $T_\mathcal{P}, F_\mathcal{P}, K_\mathcal{P} \subseteq \mathcal{A}^*_\mathcal{P}$ *are sets of observable, inconsistent and quiescent traces, satisfying the constraints:*

1. $F_\mathcal{P} \cup \{t \in T_\mathcal{P} : \nexists o \in \mathcal{A}^O_\mathcal{P} \cdot to \in T_\mathcal{P}\} \subseteq K_\mathcal{P} \subseteq T_\mathcal{P}$
2. $T_\mathcal{P}$ *is prefix closed*
3. *If* $t \in T_\mathcal{P}$ *and* $t' \in (\mathcal{A}^I_\mathcal{P})^*$*, then* $tt' \in T_\mathcal{P}$
4. *If* $t \in F_\mathcal{P}$ *and* $t' \in \mathcal{A}^*_\mathcal{P}$*, then* $tt' \in F_\mathcal{P}$.

If $\epsilon \in T_\mathcal{P}$*, we say that* $\mathcal{P}$ *is* realisable*, and is* unrealisable *otherwise.*

$T_\mathcal{P}$ consists of all observable interactions between the component and its environment. As inputs are controlled by the environment, $T_\mathcal{P}$ should be receptive to inputs (even if the component does not wish to see them). $F_\mathcal{P}$

encodes inconsistent behaviours (e.g., runtime errors, communication mismatches, undesirable inputs). On becoming inconsistent, the component exhibits chaotic behaviour, hence the extension closure of $F_{\mathcal{P}}$. $K_{\mathcal{P}}$ captures the quiescent and inconsistent behaviours of the component. A trace is quiescent if it is observable and results in a behaviour of the component that cannot immediately be extended by an output without additional stimulation from the environment. Since components can be nondeterministic, a quiescent trace may be extendable by an output on some executions; the point is that there must be at least one execution where this is not the case. Hence, $K_{\mathcal{P}}$ is not determined solely by $T_{\mathcal{P}}$ and $F_{\mathcal{P}}$.

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

Notation. Let $\mathcal{A}$ and $\mathcal{B}$ be sets of actions. For a trace t, write $t \upharpoonright \mathcal{A}$ for the projection of t onto $\mathcal{A}$. Now for $T \subseteq \mathcal{A}^*$, write $T \upharpoonright \mathcal{B}$ for $\{t \upharpoonright \mathcal{B} : t \in T\}$, $T \Uparrow \mathcal{B}$ for $\{t \in \mathcal{B}^* : t \upharpoonright \mathcal{A} \in T\}$, $T \uparrow \mathcal{B}$ for $T(\mathcal{B} \setminus \mathcal{A})(\mathcal{A} \cup \mathcal{B})^*$, and $\overline{T}$ for $\mathcal{A}^* \setminus T$.

Refinement. The specification theory comes equipped with a refinement preorder that corresponds to progress-sensitive substitutivity. $\mathcal{Q}$ is a substitutive refinement of $\mathcal{P}$ if the presence of a communication mismatch between $\mathcal{Q}$ and an arbitrary environment implies there is a communication mismatch between $\mathcal{P}$ and the environment. The progress-sensitive refinement additionally requires that $\mathcal{Q}$ must make observational progress whenever $\mathcal{P}$ can do so, meaning that, if $\mathcal{Q}$ is quiescent, then $\mathcal{P}$ must be quiescent (or inconsistent).

Since outputs are controlled locally by a component, a trace t, from which there is a sequence of output actions leading to an inconsistent trace, should be deemed as inconsistent, since the environment cannot prevent an inconsistency from arising if it allows the behaviour t. We therefore define the safe representation of a component, which is a component that becomes inconsistent immediately when the environment issues an input from which the original component can become inconsistent under its own control.

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

We now give the formal definition of refinement that respects the intuition mentioned previously.

Definition 3 (Refinement). *$\mathcal{Q}$ is a* progress-sensitive substitutive refinement *of $\mathcal{P}$, written $\mathcal{Q} \sqsubseteq^{l}_{imp} \mathcal{P}$, if:*

- $\mathcal{A}^I_{\mathcal{P}} \subseteq \mathcal{A}^I_{\mathcal{Q}}$
- $\mathcal{A}^O_{\mathcal{Q}} \subseteq \mathcal{A}^O_{\mathcal{P}}$
- $\mathcal{A}^I_{\mathcal{Q}} \cap \mathcal{A}^O_{\mathcal{P}} = \emptyset$
- $T_{\mathcal{E}(\mathcal{Q})} \subseteq T_{\mathcal{E}(\mathcal{P})} \cup (T_{\mathcal{E}(\mathcal{P})} \uparrow \mathcal{A}^I_{\mathcal{Q}})$
- $F_{\mathcal{E}(\mathcal{Q})} \subseteq F_{\mathcal{E}(\mathcal{P})} \cup (T_{\mathcal{E}(\mathcal{P})} \uparrow \mathcal{A}^I_{\mathcal{Q}})$
- $K_{\mathcal{E}(\mathcal{Q})} \subseteq K_{\mathcal{E}(\mathcal{P})} \cup (T_{\mathcal{E}(\mathcal{P})} \uparrow \mathcal{A}^I_{\mathcal{Q}})$.

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

In situations when progress-sensitivity is irrelevant, the refinement relation can be relaxed so that it is merely substitutive. This can be achieved by taking the quiescent traces of each component to be its set of observable traces. Under this assumption, the final condition of Definition 3 corresponding to the quiescent trace containment becomes redundant. In (Chilton, 2013), substitutive and progress-sensitive treatments of components are provided, where the refinement relations are denoted by $\sqsubseteq_{imp}$ and $\sqsubseteq^{l}_{imp}$ respectively. It is shown that, subject to *compatibility* of interfaces, the refinement relations are preorders ($\mathcal{P}$ and $\mathcal{Q}$ are said to be compatible if they agree on the I/O type of actions, i.e., $\mathcal{A}^I_{\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{Q}} = \emptyset = \mathcal{A}^O_{\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{Q}}$). This article presents a reasoning framework for the progress-sensitive refinement, which is a generalisation of the substitutive variety.

Parallel composition. The parallel composition of two components is obtained 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 receptiveness of observable traces. 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 composable for parallel composition (i.e., $\mathcal{A}^O_\mathcal{P} \cap \mathcal{A}^O_\mathcal{Q} = \emptyset$). Then $\mathcal{P} \mid\mid \mathcal{Q}$ is the component $\langle \mathcal{A}^I_{\mathcal{P}||\mathcal{Q}}, \mathcal{A}^O_{\mathcal{P}||\mathcal{Q}}, T_{\mathcal{P}||\mathcal{Q}}, F_{\mathcal{P}||\mathcal{Q}}, K_{\mathcal{P}||\mathcal{Q}} \rangle$, where:*

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

Informally, a trace is inconsistent if its projection is inconsistent in one component and observable in the other, while a trace can only be quiescent if its projection is quiescent in both components (or is inconsistent). A trace is observable, if it has observable projections on both components (or again is inconsistent).

As the aim of this article is to reason about component-based systems, we do not present the design-time operations of conjunction, disjunction or quotient on components. Instead, we consider a development process that generates contracts rather than component models. Therefore, we define the compositional operators directly on contracts, which yields contracts characterising sets of component implementations. Parallel composition is an exception, since it is a runtime operator that shows how different components of the specification theory interact with one another.

3. Assume-Guarantee Reasoning Framework

To support component-based reasoning, we introduce the concept of a contract, which consists of two prefix-closed sets of traces referred to as the *assumption* and *guarantee*, along with a set of *liveness* traces. 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. Progress must be made from any trace designated as live, meaning that a component may not be quiescent

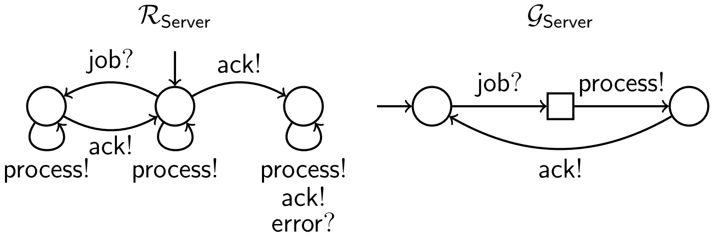

$\mathcal{R}_{\mathsf{Server}}$
$\mathcal{G}_{\mathsf{Server}}$
job?
ack!
ack!
process!
process!
process!
ack!
error?
job?
process!
ack!

Figure 1: Assumption and guarantee for Server

on a live trace. Recall that a trace is said to be quiescent in a component if it is observable and results in at least one execution that cannot immediately be extended by an output action. Quiescence has similarities with, although is not equivalent to, deadlock. In the case of the latter, a trace is said to be deadlocked just if there is some execution of the component over that trace, which cannot immediately be extended by an output or a non-inconsistent input. Therefore, deadlock implies quiescence.

Definition 5 (Contract). *A* contract $\mathcal{S}$ *is a tuple* $\langle \mathcal{A}^I_{\mathcal{S}}, \mathcal{A}^O_{\mathcal{S}}, \mathcal{R}_{\mathcal{S}}, \mathcal{G}_{\mathcal{S}}, \mathcal{L}_{\mathcal{S}} \rangle$*, in which* $\mathcal{A}^I_{\mathcal{S}}$ *and* $\mathcal{A}^O_{\mathcal{S}}$ *are disjoint sets (whose union is* $\mathcal{A}_{\mathcal{S}}$*), referred to as the inputs and outputs respectively,* $\mathcal{R}_{\mathcal{S}}$ *and* $\mathcal{G}_{\mathcal{S}}$ *are prefix closed subsets of* $\mathcal{A}^*_{\mathcal{S}}$*, 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* $tt' \in \mathcal{R}_{\mathcal{S}}$*, and* $\mathcal{L}_{\mathcal{S}} \subseteq \mathcal{R}_{\mathcal{S}} \cap \mathcal{G}_{\mathcal{S}}$ *is a (not necessarily prefix-closed) set of liveness traces.*

Since outputs are controlled by the 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. This contrasts with the work of Larsen et al. (2006), in which guarantees must be closed under input-extensions; one of our contributions is to show that this is not necessary, thus allowing significantly more flexibility when formulating contracts. Note that, by taking the set of liveness traces to be the empty set, the framework supports reasoning about safety properties, rather than safety and progress.

Example 1. *Figure 1 presents a contract for a* Server*, which can receive* job*s,* process *jobs,* ack*nowledge the processing of a job, and be placed in* error *mode. The interface is given by all the actions appearing in the diagram, with the convention that actions followed by* ? *(resp.* !*) are inputs (resp. outputs).*

We adopt the convention that a square node in a figure indicates that the contract must make progress, while a circular node has no such requirement. The assumption leaves process *unconstrained, but ensures that* error *will never be sent providing* job *and* ack *alternate in that order. The guarantee requires that any* job *received can only be* ack*nowledged after having been* process*ed, a new* job *can only arrive after the previous one has been* ack*nowledged, and whenever a* job *is received it must be* process*ed (the progress condition).*

Given a contract $\mathcal{S}$, 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 $t \in \mathcal{G}_\mathcal{S}$, t cannot become inconsistent in $\mathcal{P}$ without further stimulation from the environment, and if $t \in \mathcal{L}_\mathcal{S}$ then the component is not permitted to be quiescent. Components can thus be thought of as implementations of contracts.

Definition 6 (Satisfaction). *A component $\mathcal{P}$* satisfies *the contract $\mathcal{S}$, written $\mathcal{P} \models \mathcal{S}$, iff:*

S1. $\mathcal{A}^I_\mathcal{S} \subseteq \mathcal{A}^I_\mathcal{P}$

S2. $\mathcal{A}^O_\mathcal{P} \subseteq \mathcal{A}^O_\mathcal{S}$

S3. $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{S} = \emptyset$

S4. $\mathcal{R}_\mathcal{S} \cap T_\mathcal{P} \subseteq \mathcal{G}_\mathcal{S} \cap \overline{F_\mathcal{P}}$

S5. $\mathcal{L}_\mathcal{S} \cap T_\mathcal{P} \subseteq \overline{K_\mathcal{P}}$.

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 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, subject to compatibility.

Lemma 1. *Let $\mathcal{P}$ and $\mathcal{Q}$ be components, and let $\mathcal{S}$ be a contract. If $\mathcal{P} \models \mathcal{S}$, $\mathcal{Q} \sqsubseteq^l_{imp} \mathcal{P}$ and $\mathcal{A}^I_\mathcal{Q} \cap \mathcal{A}^O_\mathcal{S} = \emptyset$, then $\mathcal{Q} \models \mathcal{S}$.*

Therefore, any implementation $\mathcal{P}$ of $\mathcal{S}$ must not be allowed to become inconsistent under its own control when offered inputs in the assumption, and any trace of $\mathcal{P}$ that is contained in $\mathcal{L}_\mathcal{S}$ must make observational progress.

Based on this result, a contract can be characterised by its most general satisfying component, which is the minimal satisfying component under the progress-sensitive substitutive refinement preorder. Note that every contract has at least one satisfying component, although it may not be realisable. In the case that a contract has a realisable satisfying component, the contract is said to be implementable, and such a component is said to be an implementation. In order to construct such a component, it is necessary to determine the set of erroneous traces of the contract. These are traces that cannot be in any satisfying component, because they will violate the guarantee or progress condition.

Definition 7. *Let $\mathcal{S}$ be a contract. Then:*

- $\mathsf{violations}(\mathcal{S})$ *is defined as* $\{t \in \mathcal{A}_{\mathcal{S}}^* : \exists t' \in (\mathcal{A}_{\mathcal{S}}^I)^* \cdot tt' \in \mathcal{R}_{\mathcal{S}} \cap \overline{\mathcal{G}_{\mathcal{S}}}\} \cdot \mathcal{A}_{\mathcal{S}}^*$
- $\mathsf{error}(\mathcal{S})$ *is defined as the smallest set containing both* $\mathsf{violations}(\mathcal{S})$ *and* $\{t \in \mathcal{A}_{\mathcal{S}}^* : \exists t' \in (\mathcal{A}_{\mathcal{S}}^I)^* \cdot tt' \in \mathcal{L}_{\mathcal{S}}$ *and* $\forall o \in \mathcal{A}_{\mathcal{S}}^O \cdot tt'o \in \mathsf{error}(\mathcal{S})\} \cdot \mathcal{A}_{\mathcal{S}}^*$*.*

Clearly, if $t \in \mathcal{R}_{\mathcal{S}} \cap \overline{\mathcal{G}_{\mathcal{S}}}$, then t cannot be a trace of any implementation of $\mathcal{S}$. Moreover, if there is a trace that can be extended by a sequence of inputs to become t, then this also cannot be in a satisfying component, due to input-receptiveness of components. Therefore $\mathsf{violations}(\mathcal{S})$ consists of all traces from which the environment can, under its own control, violate the guarantee. On the other hand, $\mathsf{error}(\mathcal{S})$ consists of all traces that are not in any satisfying component of $\mathcal{S}$. Therefore, $\mathsf{error}(\mathcal{S})$ consists of all traces in $\mathsf{violations}(\mathcal{S})$, along with any trace that is required to be live, but cannot be so due to all output successors violating a safety or progress error. By reducing the allowed behaviours of satisfying components, further progress errors can be introduced, which is why $\mathsf{error}(\mathcal{S})$ is defined recursively. Note that, in the safety setting when $\mathcal{L}_{\mathcal{S}} = \emptyset$, it holds that $\mathsf{error}(\mathcal{S}) = \mathsf{violations}(\mathcal{S})$.

Naturally, $\mathsf{error}(\mathcal{S})$ can be defined as the least fixed point of the defining equation above. Therefore, $\mathsf{error}(\mathcal{S}) = \cup_{i \in \mathbb{N}} X_i$, where $X_0 = \emptyset$ and $X_{i+1} \triangleq \mathsf{violations}(\mathcal{S}) \cup \{t \in \mathcal{A}_{\mathcal{S}}^* : \exists t' \in (\mathcal{A}_{\mathcal{S}}^I)^* \cdot tt' \in \mathcal{L}_{\mathcal{S}}$ and $\forall o \in \mathcal{A}_{\mathcal{S}}^O \cdot tt'o \in X_i\} \cdot \mathcal{A}_{\mathcal{S}}^*$.

The least refined component satisfying a contract can now be defined in a straightforward manner.

Definition 8. *Let $\mathcal{S}$ be a contract. Then the* least refined component satisfying $\mathcal{S}$ *is the component* $\mathcal{I}(\mathcal{S}) = \langle \mathcal{A}_{\mathcal{S}}^I, \mathcal{A}_{\mathcal{S}}^O, T_{\mathcal{I}(\mathcal{S})}, F_{\mathcal{I}(\mathcal{S})}, K_{\mathcal{I}(\mathcal{S})} \rangle$*, where:*

- $T_{\mathcal{I}(\mathcal{S})} = \overline{\mathsf{error}(\mathcal{S})}$

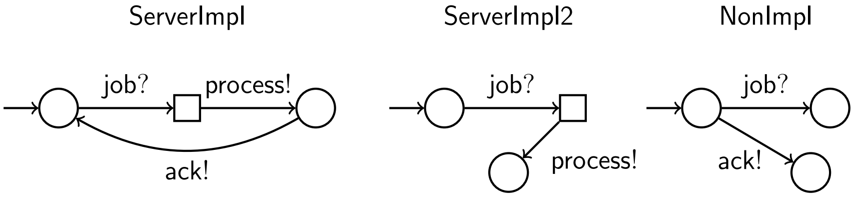


Figure 2: Implementations and non-implementation of Server

- $F_{\mathcal{I}(\mathcal{S})} = \overline{\mathsf{error}(\mathcal{S})} \cap \overline{\mathcal{R}_{\mathcal{S}}}$
- $K_{\mathcal{I}(\mathcal{S})} = \overline{\mathsf{error}(\mathcal{S})} \cap \overline{\mathcal{L}_{\mathcal{S}}}$.

The traces of $\mathcal{I}(\mathcal{S})$ are simply the behaviours that will never violate the contract. This means that, if a trace of $\mathcal{I}(\mathcal{S})$ is in the assumption, then it must also be in the guarantee, which ensures that $\mathcal{I}(\mathcal{S})$ satisfies the safety constraints of $\mathcal{S}$. In addition, if t is a trace of $\mathcal{I}(\mathcal{S})$, then no extension of t can be allowed to violate the progress conditions. Therefore, $K_{\mathcal{I}(\mathcal{S})}$ allows the component to be quiescent whenever it is not required to be live.

We now state the properties of the least refined satisfying component.

Lemma 2. *Let $\mathcal{S}$ be a contract and $\mathcal{P}$ be a component. Then:*

- *$\mathcal{I}(\mathcal{S})$ is non-realisable implies $\mathcal{S}$ is non-implementable;*
- *$\mathcal{I}(\mathcal{S}) \models \mathcal{S}$; and*
- *$\mathcal{P} \models \mathcal{S}$ iff $\mathcal{P} \sqsubseteq^{l}_{imp} \mathcal{I}(\mathcal{S})$.*

Example 2. ServerImpl *in Figure 2 is the least refined component satisfying the contract* Server *of Figure 1, where circular nodes represent quiescent, and square nodes represent non-quiescent, behaviours. As a convention, we omit input transitions to inconsistent states when drawing components (consequently, there are implicit inconsistent* job *transitions from the middle and last states and implicit inconsistent* error *transitions from all states). As* ServerImpl2 $\sqsubseteq^{l}_{imp}$ ServerImpl, ServerImpl2 *is also an implementation of* Server, *even though no* ack*nowledgement is performed (since the* Server *contract does not require progress after* process*ing).* NonImpl *is not an implementation for two reasons. First, progress is not made after receiving a* job, *and second* $\langle \mathsf{ack} \rangle \in \mathsf{violations}(\mathsf{Server})$, *since* $\langle \mathsf{ack}, \mathsf{error} \rangle \in \mathcal{R}_{\mathsf{Server}} \cap T_{\mathsf{NonImpl}}$, *while*

$\langle \mathsf{ack}, \mathsf{error} \rangle \notin \mathcal{G}_{\mathsf{Server}} \cap \overline{F_{\mathsf{NonImpl}}}$. *Note that, if the square node in* ServerImpl2 *was circular, this component would also not be an implementation of* Server*, since by non-determinism there could be a behaviour of the component that does not perform* process *after receiving a* job*.*

3.1. Refinement

Satisfaction of a contract by a component allows us to define a natural hierarchy on contracts corresponding to implementation containment. A constructive definition for this refinement relation follows.

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

R1. $\mathcal{A}^I_{\mathcal{T}} \subseteq \mathcal{A}^I_{\mathcal{S}}$

R2. $\mathcal{A}^O_{\mathcal{S}} \subseteq \mathcal{A}^O_{\mathcal{T}}$

R3. $\mathcal{A}^I_{\mathcal{S}} \cap \mathcal{A}^O_{\mathcal{T}} = \emptyset$

R4. $\mathsf{error}(\mathcal{T}) \cap \mathcal{A}^*_{\mathcal{S}} \subseteq \mathsf{error}(\mathcal{S})$

R5. $\mathcal{R}_{\mathcal{T}} \cap \mathcal{A}^*_{\mathcal{S}} \subseteq \mathcal{R}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$

R6. $\mathcal{L}_{\mathcal{T}} \cap \mathcal{A}^*_{\mathcal{S}} \subseteq \mathcal{L}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$.

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 impose necessary conditions on the alphabets to uphold this principle. For condition R4, any component having a trace $t \in \mathsf{error}(\mathcal{T}) \cap \mathcal{A}^*_{\mathcal{S}}$ 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 or progress condition on $\mathcal{S}$, i.e., $t \in \mathsf{error}(\mathcal{S})$. 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 $t \in \mathsf{error}(\mathcal{S})$ or t must be in $\mathcal{R}_{\mathcal{S}}$, so that the component cannot satisfy $\mathcal{S}$. Condition R6 forces implementations of $\mathcal{S}$ to be live on a trace t whenever t is required to be live on $\mathcal{T}$, unless a safety or progress violation is inevitable, in which case the implementation would have suppressed an output in the assumption at an earlier stage. This requirement guarantees implementation containment, and also that refinement is a preorder (subject to compatibility).

Definition 9 gives a sound and complete characterisation of refinement, as proven in Lemma 3. In related work, one often sees sound and incomplete characterisations, which may be more intuitive. One possibility is to replace conditions R4 and R5 by $\mathcal{R}_\mathcal{T} \subseteq \mathcal{R}_\mathcal{S}$ and $(\mathcal{G}_\mathcal{S} \cap \mathcal{R}_\mathcal{T}) \subseteq \mathcal{G}_\mathcal{T}$ (assuming identical interfaces of $\mathcal{S}$ and $\mathcal{T}$). This defines an equivalence on contracts with the same assumptions, where the guarantees differ only outside the assumptions. More formally, $\mathcal{S}$ and $\mathcal{T}$ are equivalent if $\mathcal{R}_\mathcal{S} = \mathcal{R}_\mathcal{T}$ and $(\mathcal{G}_\mathcal{S} \cap \mathcal{R}_\mathcal{S}) = (\mathcal{G}_\mathcal{T} \cap \mathcal{R}_\mathcal{T})$.

Lemma 3. *Refinement captures implementation containment:*

$$\mathcal{S} \sqsubseteq \mathcal{T} \iff \{\mathcal{P} : \mathcal{P} \models \mathcal{S} \text{ and } \mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{T} = \emptyset\} \subseteq \{\mathcal{P} : \mathcal{P} \models \mathcal{T}\}.$$

Larsen et al. (2006) give a sound and complete characterisation of their refinement relation (which corresponds to implementation containment, as in this article) 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, and Condition R6 is not necessary, since they only consider safety properties, rather than safety and progress.

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

Lemma 4 (Weak transitivity). *Let $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{U}$ be contracts such that $\mathcal{A}^I_\mathcal{S} \cap \mathcal{A}^O_\mathcal{U} = \emptyset$. If $\mathcal{S} \sqsubseteq \mathcal{T}$ and $\mathcal{T} \sqsubseteq \mathcal{U}$, then $\mathcal{S} \sqsubseteq \mathcal{U}$.*

Example 3. *A new contract* $\mathsf{Server2}$ *(with assumption $\mathcal{R}_{\mathsf{Server}}$ and guarantee obtained from $\mathcal{G}_{\mathsf{Server}}$ by removing the* ack *transition) is a refinement of* Server*, since it has fewer implementations. In particular,* $\mathcal{I}(\mathsf{Server2}) = \mathsf{ServerImpl2}$. $\mathsf{ServerImpl}$ *is not an implementation of* $\mathsf{Server2}$ *because* $\langle \mathsf{job}, \mathsf{process}, \mathsf{ack} \rangle \in \mathsf{violations}(\mathsf{Server2})$.

As we can represent a contract by its most general satisfying component, we can also do the reverse and represent a component by its most general contract. This can be found by examining the component's safe traces.

Definition 10. *The* characteristic contract *for component $\mathcal{P}$ is a contract $\mathcal{AG}(\mathcal{P}) = \langle \mathcal{A}^I_{\mathcal{P}}, \mathcal{A}^O_{\mathcal{P}}, \mathcal{R}_{\mathcal{AG}(\mathcal{P})}, \mathcal{G}_{\mathcal{AG}(\mathcal{P})}, \mathcal{L}_{\mathcal{AG}(\mathcal{P})} \rangle$, where $\mathcal{R}_{\mathcal{AG}(\mathcal{P})} = \mathcal{A}^*_{\mathcal{P}} \setminus F_{\mathcal{E}(\mathcal{P})}$, $\mathcal{G}_{\mathcal{AG}(\mathcal{P})} = T_{\mathcal{P}} \setminus F_{\mathcal{E}(\mathcal{P})}$ and $\mathcal{L}_{\mathcal{AG}(\mathcal{P})} = T_{\mathcal{P}} \setminus K_{\mathcal{E}(\mathcal{P})}$.*

The largest assumption safe for component $\mathcal{P}$ is the set of all traces that cannot become inconsistent under $\mathcal{P}$'s own control, while the guarantee is this same set of traces constrained to the behaviour of $\mathcal{P}$. The set of liveness traces $\mathcal{L}_{\mathcal{AG}(\mathcal{P})}$ contains the non-inconsistent traces of $\mathcal{P}$ that are not quiescent.

The following lemma shows the properties of the characteristic contract.

Lemma 5. *Let $\mathcal{P}$ be a component and $\mathcal{S}$ be a contract. 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 contract by a component is equivalent to checking whether the characteristic contract of the component is a refinement of the contract. This means that implementability of contracts, built up compositionally, follows immediately from compositionality results on contracts.

In the subsequent sections, we define the compositional operators of the specification theory directly on contracts. The operators are only defined when the contracts to be composed are *composable* (the conditions being specified as part of the definitions). We also present a number of sound and complete AG rules for inferring properties of composite systems from the properties of their subcomponents.

3.2. Parallel Composition

The parallel composition of contracts is defined as the least-refined contract satisfying independent implementability. Therefore, $\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}}$ is the smallest contract having $\mathcal{P} \mid\mid \mathcal{Q}$ as an implementation whenever $\mathcal{P} \models \mathcal{S}_{\mathcal{P}}$ and $\mathcal{Q} \models \mathcal{S}_{\mathcal{Q}}$. A constructive definition of contract composition is based on the well-established theorem of Abadi and Lamport (1993), which has appeared in several forms (Collette, 1993; Abadi and Lamport, 1995; Jonsson and Yih-Kuen, 1996). The composed contract has the largest assumption that prevents any implementation (say $\mathcal{P}$) of one contract ($\mathcal{S}_{\mathcal{P}}$) producing behaviour observable by the other contract ($\mathcal{S}_{\mathcal{Q}}$) that is outside of its assumption ($\mathcal{R}_{\mathcal{S}_{\mathcal{Q}}}$). The guarantee of the composition, on the other hand, is constrained to what can be guaranteed by both contracts to be composed. The liveness condition requires that a trace in the composition must make progress if at least one of the contracts requires this, since the parallel composition cannot suppress the output behaviour of implementing components.

Definition 11. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be contracts composable for parallel composition (i.e., $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} = \emptyset$). Then $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q}$ is a contract $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}, \mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \rangle$, where:*

- $\mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = (\mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}) \setminus (\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}})$
- $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$
- *$\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ is the largest prefix closed set such that $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}(\mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})^*$ is contained within the union of:*
  - $(\mathcal{R}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}) \cap (\mathcal{R}_{\mathcal{S}_\mathcal{Q}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})$
  - $\mathsf{error}(\mathcal{S}_\mathcal{P}) \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$
  - $\mathsf{error}(\mathcal{S}_\mathcal{Q}) \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$
- $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap (\overline{\mathsf{error}(\mathcal{S}_\mathcal{P}) \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}}) \cap (\overline{\mathsf{error}(\mathcal{S}_\mathcal{Q}) \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}})$
- $\mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap [(\mathcal{L}_{\mathcal{S}_\mathcal{P}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}) \cup (\mathcal{L}_{\mathcal{S}_\mathcal{Q}} \Uparrow \mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})]$.

The assumption $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ captures all behaviours whose projections onto $\mathcal{A}_{\mathcal{S}_\mathcal{P}}$ and $\mathcal{A}_{\mathcal{S}_\mathcal{Q}}$ are either contained within the assumptions $\mathcal{R}_{\mathcal{S}_\mathcal{P}}$ and $\mathcal{R}_{\mathcal{S}_\mathcal{Q}}$, or have violated at least one of the contracts. This rules out a trace t that has not violated either of the contracts, but is no longer within both assumptions (say $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}}$). For such a trace, no guarantee can be given, since $\mathcal{S}_\mathcal{P}$ can have an implementation with the inconsistent trace $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}}$, while $\mathcal{S}_\mathcal{Q}$ can have an implementation with the trace $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}}$. The parallel composition of these two components would thus be inconsistent on t, and so would not satisfy $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q}$ if $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$.

The guarantee $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ is constrained to the traces in $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ that do not violate either $\mathcal{S}_\mathcal{P}$ or $\mathcal{S}_\mathcal{Q}$. Any trace in an implementation of a contract must not be allowed to violate the contract, meaning that it must suppress an output before a violation can occur. Consequently, the parallel composition of such an implementation with an implementation of the other contract cannot proceed beyond this suppressed output, so $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ need not guarantee

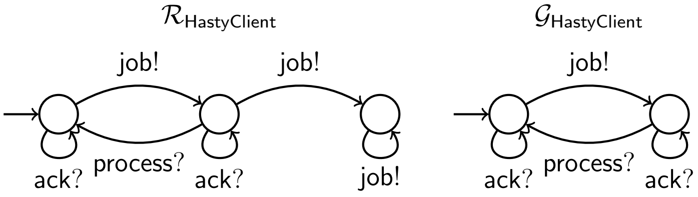

$\mathcal{R}_{\mathsf{HastyClient}}$
job!
job!
ack?
process?
ack?
job!
$\mathcal{G}_{\mathsf{HastyClient}}$
job!
ack?
process?
ack?

Figure 3: Assumption and guarantee for HastyClient

anything beyond that output. Thus, $\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ contains only traces reachable by the composition of any two implementations of the respective contracts that are in the assumption $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$.

By the definition of $\mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$, we know that $\mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \subseteq \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ as required, and any trace in $\mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ requires that at least one of $\mathcal{S}_\mathcal{P}$ or $\mathcal{S}_\mathcal{Q}$ is live. Therefore, the parallel composition of any pair of implementations of $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ must be live on this trace.

Example 4. *Figure 3 presents a contract* HastyClient *that can send a* job *to a server whenever the last job has been* process*ed, regardless of whether it has been* ack*nowledged or not. The composition of* HastyClient *with* Server *is a contract for which nothing can be assumed or guaranteed, since the output sequence* $\langle \mathsf{job!}, \mathsf{process!}, \mathsf{job!} \rangle$ *is not in* $\mathsf{error}(\mathsf{HastyClient})$*, but is also not in* $\mathcal{R}_{\mathsf{Server}}$ *or* $\mathsf{error}(\mathsf{Server})$*. This is problematic because* $\langle \mathsf{job!}, \mathsf{process?}, \mathsf{job!} \rangle$ *can be a trace in an implementation of* HastyClient*, while* $\langle \mathsf{job?}, \mathsf{process!}, \mathsf{job?} \rangle$ *can be an inconsistent trace in an implementation of* Server *(providing* $\langle \mathsf{job?}, \mathsf{process!} \rangle$ *is consistent, since* $\langle \mathsf{job?}, \mathsf{process!}, \mathsf{job?} \rangle \notin \mathcal{R}_{\mathsf{Server}}$*). Note that* $\langle \mathsf{job!}, \mathsf{process!}, \mathsf{job!} \rangle$ *is an inconsistent trace in the parallel composition of the two implementations, which explains why the assumption must be empty.*

Example 5. *In contrast to* HastyClient*, the composition of* RestrainedClient *(Figure 4) and* Server *is a contract with a completely open assumption (anything may be assumed), since the allowed behaviours of each contract cannot violate, or fall outside the assumption of, the other contract. The guarantee is equivalent to* $\mathcal{G}_{\mathsf{Server}}$*, having converted all actions to outputs.*

Subject to suitable constraints on the interfaces of contracts, it can be shown that parallel composition is monotonic under refinement.

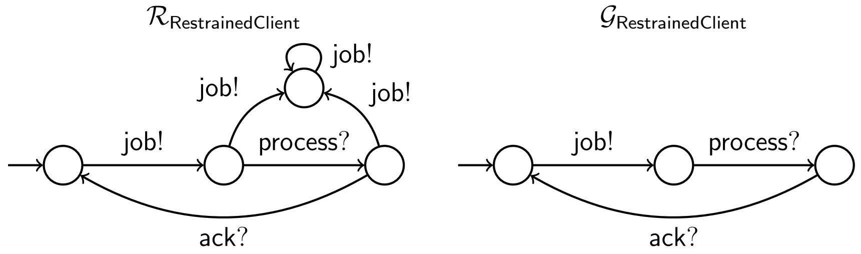

$\mathcal{R}_{\mathsf{RestrainedClient}}$
job!
job!
job!
job!
process?
ack?
$\mathcal{G}_{\mathsf{RestrainedClient}}$
job!
process?
ack?

Figure 4: Assumption and guarantee for RestrainedClient

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

In this theorem, the condition $\mathcal{A}^I_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} = \emptyset$ ensures compatibility of $\mathcal{S}'_\mathcal{P} \mid\mid \mathcal{S}'_\mathcal{Q}$ and $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q}$, which does not necessarily follow from $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}'_\mathcal{P}$, along with $\mathcal{S}_\mathcal{Q}$ and $\mathcal{S}'_\mathcal{Q}$, agreeing. The remaining condition is standard for compositionality of parallel composition (cf. de Alfaro and Henzinger (2001)), and ensures that, for any trace $t \in (\mathcal{A}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{A}_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}})^*$, $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} = t \upharpoonright \mathcal{A}_{\mathcal{S}'_\mathcal{P}}$ and $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = t \upharpoonright \mathcal{A}_{\mathcal{S}'_\mathcal{Q}}$ (that is, the projections onto $\mathcal{A}_{\mathcal{S}_\mathcal{P}}$ and $\mathcal{A}_{\mathcal{S}'_\mathcal{P}}$, and $\mathcal{A}_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{A}_{\mathcal{S}'_\mathcal{Q}}$, must match). Based on the monotonicity result, a sound and complete AG rule can be formulated for parallel composition.

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

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

Abadi and Lamport (1993) prove soundness of their parallel composition rule. Maier (2003) demonstrate that compositional circular AG rules are not both sound and complete. This seems at odds with our rule, but in our setting circularity is broken, since 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.

3.3. Conjunction

In this section, we define a conjunctive operator on contracts for combining independently developed requirements. From this, we show that the operator is compositional and corresponds to the meet operation on the refinement relation. This allows us to conclude that implementations of a conjunctive contract must be implementations of both contracts to be conjoined. Based on this, we formulate a sound and complete AG rule for conjunction.

Definition 12. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be contracts composable for conjunction (i.e., $\mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$ and $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$ are disjoint). Then $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ is a contract $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}, \mathcal{A}^O_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}, \mathcal{R}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}, \mathcal{G}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}, \mathcal{L}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}} \rangle$ defined by:*

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

The assumption $\mathcal{R}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}$ encompasses all of the assumptions made by either $\mathcal{S}_\mathcal{P}$ or $\mathcal{S}_\mathcal{Q}$, while the guarantee $\mathcal{G}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}$ is the largest subset of $\mathcal{R}_{\mathcal{S}_\mathcal{P}\wedge\mathcal{S}_\mathcal{Q}}$ that cannot violate the guarantees of $\mathcal{S}_\mathcal{P}$ or $\mathcal{S}_\mathcal{Q}$. Progress, on the other hand, must be made when at least one of the contracts can make progress, and the other contract has not violated its guarantee.

The next theorem shows 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 $\mathcal{S}_\mathcal{Q}$, which means that $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ is only implementable providing $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ share a common implementation. The fact that $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ may not have an implementation when both $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ do is a consequence of the conflicting nature of safety and progress.

Theorem 3. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$, and $\mathcal{S}'_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q}$ be contracts composable for conjunction. Then:*

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

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

Theorem 4. *Let $\mathcal{P}$ be a component, and let $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}$ be contracts such that $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{S} = \emptyset$. Then the following AG rule is both sound and complete:*

$$\textsc{Conjunction}\,\frac{\mathcal{P} \models \mathcal{S}_1 \qquad \mathcal{P} \models \mathcal{S}_2 \qquad \mathcal{S}_1 \wedge \mathcal{S}_2 \sqsubseteq \mathcal{S}}{\mathcal{P} \models \mathcal{S}}.$$

Example 6. *A* Client *is assumed to have an interface that can send* job*s to, and await* ack*nowledgements from, a server, can* login *once instructed by a user, and can* logout *when it pleases. Thus,* job *and* logout *are outputs, whereas* login *and* ack *are inputs. The combined effect of* Client *and* Server *should satisfy the properties:*

- Spec1: *If the observed behaviour over* login *and* logout *is always a prefix of* $(\mathsf{login}, \mathsf{logout})^*$*, then* login *and* process *should alternate.*
- Spec2: *If the observed behaviour over* login *and* logout *is always a prefix of* $(\mathsf{login}, \mathsf{logout})^*$*, then* process *and* logout *should alternate, and progress must be made whenever a job has been* process*ed and before a* logout *request is seen.*

Spec1 *and* Spec2 *are represented by the contracts* $\langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec1}} \rangle$ *and* $\langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec2}} \rangle$ *respectively, as depicted in Figure 5. The combined effect of these properties is given by the conjunctive contract* $\mathsf{Spec1} \wedge \mathsf{Spec2} = \langle \mathcal{R}_{\mathsf{Spec}}, \mathcal{G}_{\mathsf{Spec1} \wedge \mathsf{Spec2}} \rangle$*, the guarantee of which is presented in Figure 6. As* Spec1 *and* Spec2 *have the same interface, the guarantee of the conjunction is obtained as the intersection of* $\mathcal{G}_{\mathsf{Spec1}}$ *and* $\mathcal{G}_{\mathsf{Spec2}}$*. The liveness requirement of* Spec2 *manifests itself as a liveness requirement in the conjunction after* process *and before* logout*, indicated by the square node in Figure 6.*

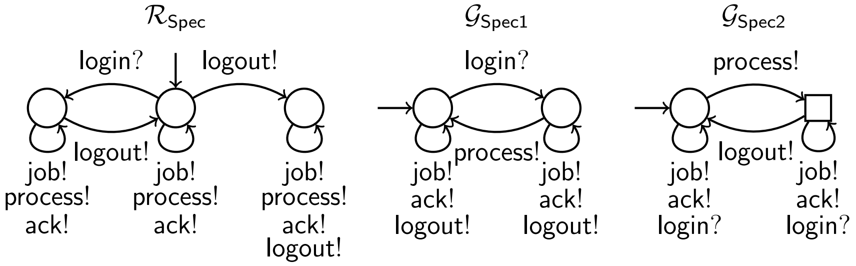


Figure 5: Assumption and guarantees for Spec1 and Spec2

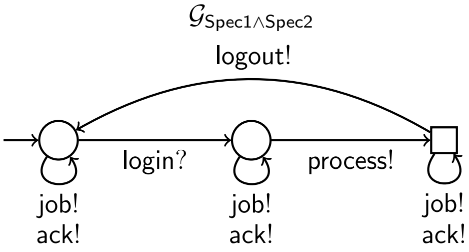


Figure 6: Guarantee for the conjunction of Spec1 and Spec2

3.4. Disjunction

In this section, we formulate a disjunctive operator on contracts. Whereas conjunction combines requirements in the sense that it strengthens guarantees, disjunction strengthens the assumptions on the environment to the extent that the implementations of the disjunction contains the union of the implementations of the contracts to be composed. Being the dual of conjunction, we show that disjunction is the join operator on the refinement preorder, and provide a sound and complete assume-guarantee rule.

Definition 13. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$ be contracts composable for disjunction (i.e., the same conditions as for conjunction). Then $\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}$ is a contract $\langle \mathcal{A}^I_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}$, $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}, \mathcal{R}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}, \mathcal{G}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}, \mathcal{L}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}\rangle$, where:*

- *$\mathcal{R}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}}$ is the intersection of the following sets:*
  - $\mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P}) \cup ((\mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P})) \uparrow \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}})$
  - $\mathcal{R}_{\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{Q}) \cup ((\mathcal{R}_{\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{Q})) \uparrow \mathcal{A}^O_{\mathcal{S}_\mathcal{P}})$
- $\mathcal{G}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}} = \mathcal{R}_{\mathcal{S}_\mathcal{P}\vee\mathcal{S}_\mathcal{Q}} \cap (\overline{\mathsf{error}(\mathcal{S}_\mathcal{P})} \cup \overline{\mathsf{error}(\mathcal{S}_\mathcal{Q})})$

- $\mathcal{L}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ *is the intersection of the following sets:*
  - $\mathcal{G}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$
  - $\mathcal{L}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P}) \cup ((\mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P})) \uparrow \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}})$
  - $\mathcal{L}_{\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{Q}) \cup ((\mathcal{R}_{\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{Q})) \uparrow \mathcal{A}^O_{\mathcal{S}_\mathcal{P}})$.

This definition of disjunction satisfies properties similar to those for conjunction, and hence is the join operator on the refinement preorder.

Theorem 5. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{Q}$, and $\mathcal{S}'_\mathcal{P}$ and $\mathcal{S}'_\mathcal{Q}$ be contracts composable for disjunction. Then:*

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

Based on the algebraic properties of disjunction, we can formulate a sound and complete AG rule. This demonstrates that a disjunctive contract contains the union of the implementations of the contracts to be composed, although there may be additional implementations that are not implementations of either contract.

Theorem 6. *Let $\mathcal{P}$ be a component, and let $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}$ be contracts such that $\mathcal{S}_1$ and $\mathcal{S}_2$ are composable for disjunction, and $\mathcal{A}^I_\mathcal{P} \cap \mathcal{A}^O_\mathcal{S} = \emptyset$. Then the following AG rule is both sound and complete:*

$$\textsc{Disjunction}\,\frac{\mathcal{P} \models \mathcal{S}_1 \quad \textit{or} \quad \mathcal{P} \models \mathcal{S}_2 \qquad \mathcal{S}_1 \vee \mathcal{S}_2 \sqsubseteq \mathcal{S}}{\mathcal{P} \models \mathcal{S}}.$$

The disjunction $\mathcal{S}_1 \vee \mathcal{S}_2$ is the strongest contract containing the union of the implementations for $\mathcal{S}_1$ and $\mathcal{S}_2$. In contrast to conjunction, which precisely characterises the intersection of the implementation sets, there may be implementations of the disjunction that are not implementations of either $\mathcal{S}_1$ or $\mathcal{S}_2$. The Hasse diagram of Figure 7 makes this relationship clear by depicting the least refined implementations of the contracts $\mathcal{S}_1$ and $\mathcal{S}_2$, along with their conjunction and disjunction. The implementations of a contract $\mathcal{S}$ are simply those implementations that appear above (i.e., can be reached from) $\mathcal{I}(\mathcal{S})$.

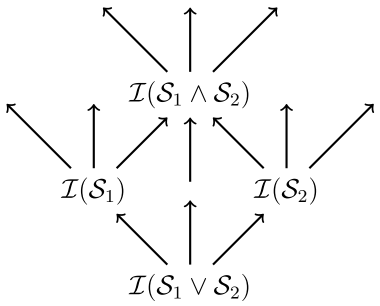

$\mathcal{I}(\mathcal{S}_1 \wedge \mathcal{S}_2)$
$\mathcal{I}(\mathcal{S}_1)$
$\mathcal{I}(\mathcal{S}_2)$
$\mathcal{I}(\mathcal{S}_1 \vee \mathcal{S}_2)$

Figure 7: Implementations of contracts $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_1 \wedge \mathcal{S}_2$ and $\mathcal{S}_1 \vee \mathcal{S}_2$

3.5. Quotient

The AG rule for parallel composition in Theorem 2 makes use of the composition $\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}}$. 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 contracts. Then the quotient $\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}$ is a contract $\langle \mathcal{A}^I_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}, \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}, \mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}, \mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}, \mathcal{L}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \rangle$, defined only when $\mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}} \subseteq \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}}$, where:*

- $\mathcal{A}^I_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} = \mathcal{A}^I_{\mathcal{S}_{\mathcal{W}}} \setminus \mathcal{A}^I_{\mathcal{S}_{\mathcal{P}}}$
- $\mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} = \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}} \setminus \mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}}$
- $\mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} = [\mathcal{R}_{\mathcal{S}_{\mathcal{W}}} \cap (\overline{\mathsf{error}(\mathcal{S}_{\mathcal{P}}) \Uparrow \mathcal{A}_{\mathcal{S}_{\mathcal{W}}}})] \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$
- $\mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$ *is the largest subset of* $\mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$ *disjoint from* $[\mathcal{R}_{\mathcal{S}_{\mathcal{W}}} \cap (\overline{\mathsf{error}(\mathcal{S}_{\mathcal{P}}) \Uparrow \mathcal{A}_{\mathcal{S}_{\mathcal{W}}}}) \cap (\mathsf{error}(\mathcal{S}_{\mathcal{W}}) \cup (\overline{\mathcal{R}_{\mathcal{S}_{\mathcal{P}}} \Uparrow \mathcal{A}_{\mathcal{S}_{\mathcal{W}}}}))] \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$
- $\mathcal{L}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} = \mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \cap [\mathcal{L}_{\mathcal{S}_{\mathcal{W}}} \cap (\overline{\mathsf{error}(\mathcal{S}_{\mathcal{P}}) \Uparrow \mathcal{A}_{\mathcal{S}_{\mathcal{W}}}}) \cap (\overline{\mathcal{L}_{\mathcal{S}_{\mathcal{P}}} \Uparrow \mathcal{A}_{\mathcal{S}_{\mathcal{W}}}})] \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}.$

Although not immediately obvious from the formulation of the previous definition, the assumption is closed under output-extensions, the assumption and guarantee are both prefix-closed, and the liveness set is contained within both the assumption and guarantee. Therefore, the quotient is a well-formed contract. Before explaining the intuition behind the definition, we introduce the following theorem, which shows that the quotient operator on contracts yields the weakest decomposition of the parallel composition.

Theorem 7. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be contracts. Then there exists a contract $\mathcal{S}_\mathcal{Q}$ such that $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$ iff the following properties hold:*

- *The quotient $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is defined*
- $\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}) \sqsubseteq \mathcal{S}_\mathcal{W}$
- $\mathcal{A}^I_{\mathcal{S}_\mathcal{Q}} = \mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ *implies* $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$.

In explaining the intuition behind the definition of quotient, it is necessary to consider the properties of Theorem 7 along with the formulation of refinement and parallel composition (Definitions 9 and 11). To obtain the least refined solution $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ for $\mathcal{S}_\mathcal{P} \mid\mid X \sqsubseteq \mathcal{S}_\mathcal{W}$, it is essential that the quotient roughly[1] satisfies the following properties for $t \in \mathcal{A}^*_{\mathcal{S}_\mathcal{W}}$:

- If $t \in \mathsf{error}(\mathcal{S}_\mathcal{W})$ and:
  - $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$, then $t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$, so there is no need for $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$
  - $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$, then it must hold that $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$ (i.e., take $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}}$ so that $t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$.
- If $t \in \mathcal{R}_{\mathcal{S}_\mathcal{W}} \backslash \mathsf{error}(\mathcal{S}_\mathcal{W})$, then first attempt to ensure that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}\mid\mid(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})} \backslash \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$ holds, and failing that ensure $t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$:
  - If $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$, then $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}\mid\mid(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})}$, so there is no need for $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$.
  - If $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$ and $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{P}}$, simply take $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$, so that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}\mid\mid(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})} \backslash \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$.
  - If $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$ and $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}}$, then we require $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$, so take $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}}$.
- If $t \in \mathcal{L}_{\mathcal{S}_\mathcal{W}} \setminus \mathsf{error}(\mathcal{S}_\mathcal{W})$ and $t \notin \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}))$, then we require $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P}\mid\mid(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})}$. If $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, then it need not hold that $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$. If instead $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, then it must hold that $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$.

[1] Exceptions need to be made since the conditions are not mutually exclusive, and properties like prefix closure and output-extendability must be maintained.

Note that, in the definition of $\mathcal{G}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$, the set required to be disjoint from $\mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ essentially characterises a subset of traces that must be in $\mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$. Furthermore, in the definition of quotient, the set of inputs $\mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ is taken to be the smallest set such that $\mathcal{A}^I_{\mathcal{S}_\mathcal{W}} \subseteq \mathcal{A}^I_{\mathcal{S}_\mathcal{P}||(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})}$, the latter being a necessary condition for $\mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}) \sqsubseteq \mathcal{S}_\mathcal{W}$. Yet, in fact, the set of inputs for quotient can be parameterised without affecting the results of Theorem 7. This is useful, since enlarging the set of inputs allows for the possibility of the quotient to observe the behaviour of $\mathcal{S}_\mathcal{P}$, which yields a contract with more specific behaviour. Such a contract cannot be obtained through refinement alone, as $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ does not imply $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$ in general, since monotonicity only holds on a restricted set of interfaces (cf. Theorem 1).

Remark 1 (Parameterised quotient). *As justified above, the set of inputs $\mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ in Definition 14 can be replaced by any set X such that $\mathcal{A}^I_{\mathcal{S}_\mathcal{W}} \setminus \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \subseteq X \subseteq \overline{\mathcal{A}^O_{\mathcal{S}_\mathcal{W}}}$.*

We now present a sound and complete AG rule for quotient on contracts.

Theorem 8. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be contracts such that $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is defined, let $\mathcal{P}$ range over components having the same interface as $\mathcal{S}_\mathcal{P}$, and let $\mathcal{Q}$ be a component having the same interface as $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ (where the quotient is parameterised on the set $\mathcal{A}^I_\mathcal{Q}$). Then the following AG rule is both sound and complete:*

$$\textsc{Quotient}\frac{\forall \mathcal{P} \cdot \mathcal{P} \models \mathcal{S}_\mathcal{P} \textit{ implies } \mathcal{P} \mid\mid \mathcal{Q} \models \mathcal{S}_\mathcal{W}}{\mathcal{Q} \models \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}.$$

We insist that the components $\mathcal{P}$ and $\mathcal{Q}$ must have the same interfaces as their respective contracts, since parallel composition is only monotonic when restrictions are placed on the interfaces of the contracts to be composed (cf. Theorem 1). The proof of the rule hints that the universal quantification over all components $\mathcal{P}$ can be replaced by the single component $\mathcal{I}(\mathcal{S}_\mathcal{P})$, meaning that it is not necessary to quantify over an infinite number of components in order to satisfy the premise.

Corollary 1. *Let $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{W}$ be contracts such that $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ is defined, and let $\mathcal{Q}$ be a component having the same interface as $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ (where the quotient* *is parameterised on the set $\mathcal{A}^I_{\mathcal{Q}}$). Then the following AG rule is both sound and complete:*

$$\textsc{Quotient-Revised}\frac{\mathcal{I}(\mathcal{S}_{\mathcal{P}}) \mid\mid \mathcal{Q} \models \mathcal{S}_{\mathcal{W}}}{\mathcal{Q} \models \mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}.$$

Example 7. *We now derive a* Client *contract (having an interface as described in Example 6) that can interact with* Server *(Figure 1), whilst satisfying the requirements of* $\mathsf{Spec1} \wedge \mathsf{Spec2}$ *(Figures 5 and 6). This is obtained as* $(\mathsf{Spec1} \wedge \mathsf{Spec2})/\mathsf{Server}$*, where the quotient operator is parameterised on the set of inputs* $\{\mathsf{login}, \mathsf{ack}\}$*. The resulting contract is shown in Figure 8. The guarantee is obtained from the assumption by pruning any trace whose corresponding projections are in* $\mathsf{error}(\mathsf{Spec1} \wedge \mathsf{Spec2})$ *or not in* $\mathcal{R}_{\mathsf{Server}}$*. Note that the bottom right node of* $\mathcal{G}_{\mathsf{Client}}$ *is required to be live in Figure 8, since* $\mathsf{Spec1} \wedge \mathsf{Spec2}$ *requires liveness after the trace* $\langle \mathsf{login},\ \mathsf{job},\ \mathsf{process}\rangle$*, while the projection of this trace onto* Server *does not guarantee liveness. As all output extensions of the trace* $\langle \mathsf{login}, \mathsf{job}\rangle$ *in* Client *are contained within* $\mathsf{error}(\mathsf{Client})$*, it follows that* $\langle \mathsf{login}, \mathsf{job}\rangle \in \mathsf{error}(\mathsf{Client})$*. Consequently, every implementation of the* Client *contract is unable to issue a* job *after a successful* login*, because, if it were to do so, there would be no guarantee that the* Server *will* ack*nowledge the processing, meaning that a liveness violation can arise. If, however, the liveness requirement was dropped from* $\mathsf{Spec2}$*, then no state of* $\mathcal{G}_{\mathsf{Client}}$ *would need to be live, and so an implementation of the* Client *contract could send a* job *after having received a* login *request.*

3.6. 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 contracts. This is useful for system development, as we will often have the specification of a whole system, rather than the specifications of the subsystems to be composed.

Corollary 2. *Let $\mathcal{P}$ and $\mathcal{Q}$ be components, and let $\mathcal{S}_{\mathcal{P}}$, $\mathcal{S}_{\mathcal{Q}}$ and $\mathcal{S}$ be contracts such that $\mathcal{A}_{\mathcal{P}} \cap \mathcal{A}_{\mathcal{Q}} \cap \mathcal{A}_{\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}}} \subseteq \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \cap \mathcal{A}_{\mathcal{S}_{\mathcal{Q}}}$ and $\mathcal{A}^I_{\mathcal{P} \mid\mid \mathcal{Q}} \cap \mathcal{A}^O_{\mathcal{S}} = \emptyset$. When the quotient is parameterised on $\mathcal{A}^I_{\mathcal{S}_{\mathcal{Q}}}$, the following rule is both sound and complete:*

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

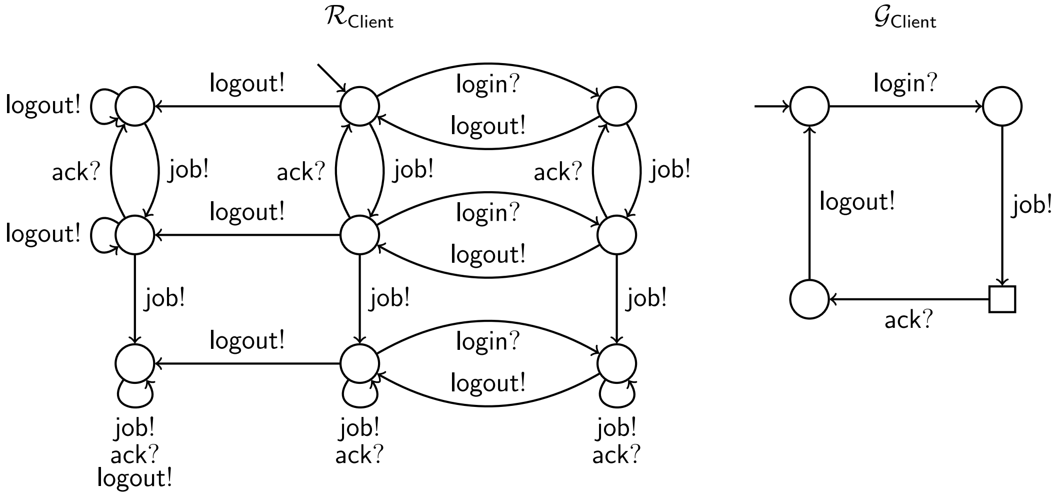
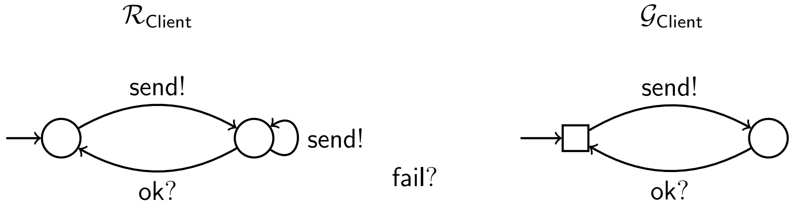


Figure 8: Assumption and guarantee for Client



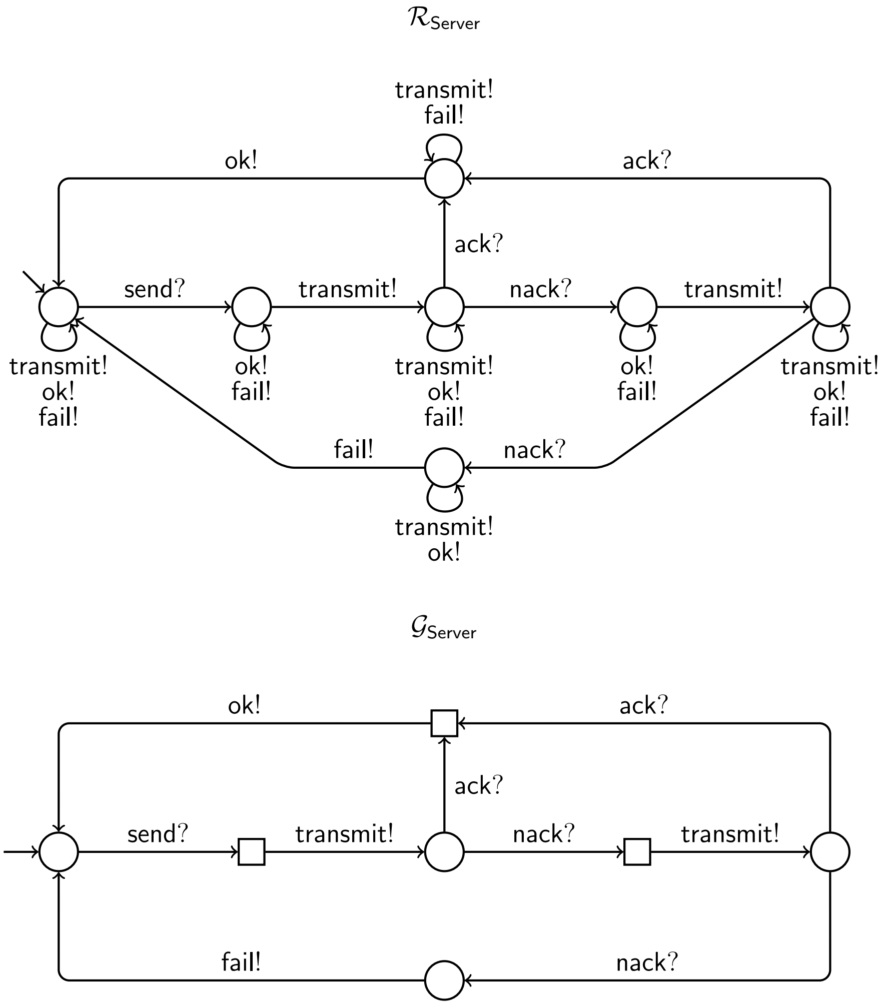
$\mathcal{R}_{\mathsf{Client}}$
$\mathcal{G}_{\mathsf{Client}}$
send!
send!
ok?
fail?
send!
ok?

Figure 9: Assumption and guarantee of a Client that additionally has fail? in its interface

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 $\mathcal{S}_\mathcal{P} \parallel \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}$. Note that this substitution requires no change to the constraints on the contracts and components. The rule is useful for scenarios when the contract $\mathcal{S}$ is supplied along with a subcontract $\mathcal{S}_\mathcal{P}$ (or for when a subcontract $\mathcal{S}_\mathcal{P}$ can easily be inferred). In such circumstances, the missing contract $\mathcal{S}_\mathcal{Q}$ can be taken as any refinement of $\mathcal{S}/\mathcal{S}_\mathcal{P}$.

4. Case Study

To demonstrate our assume-guarantee framework with its range of operators, and to relate it to previously proposed frameworks, we consider a case study that is a variant of the running example used by Larsen et al. (2006). The case study considers a client needing to send data over a potentially unreliable communication link. Between the client and the link sits a



$\mathcal{R}_{\mathsf{Server}}$

$\mathcal{G}_{\mathsf{Server}}$

Figure 10: Assumption and guarantee of a Server

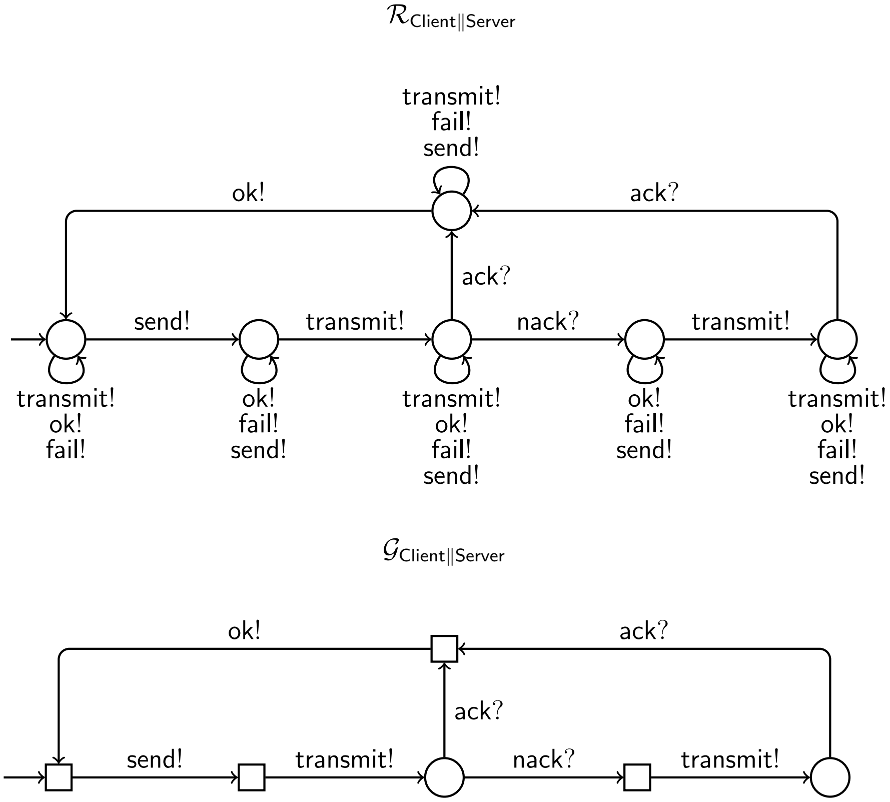
server which, to a limited extent, recovers from failures in the communication link and acknowledges successful transmissions to the client. Larsen et al. (2006) used this example to illustrate parallel composition and to show how different assumptions about the reliability of the communication link can be exchanged in the specification of the server. In this section, we not only illustrate parallel composition, but also show:

- how our quotient operation can be used to derive the weakest contract for the communication link, given a specification of the server; and
- how the conjunction operation can be used to combine several specifications of the communication link.

More precisely, we first form the composition of the client and the server, from which we derive the weakest contract for the communication link that guarantees the entire system will not encounter runtime errors (including communication mismatches). Thereafter, we refine this contract by conjoining it with a contract representing a protocol for the communication link.

Describing the operation of the participating components, a Client (see Figure 9) can communicate with a Server (Figure 10) by sending data, and can observe whether the transmission was ok or whether it failed. The Server, on the other hand, is an intermediary between the Client and a communication link. It receives data from the Client via the send interaction, and then transmits it to the communication link, after which it waits for positive or negative confirmation that the data was successfully delivered, in the form of ack and nack signals, respectively. In the case that the transmission is acknowledged, the Server indicates to the Client that all is ok. Otherwise, if nack is received from the link, the Server attempts to retransmit, and if nack is received for a second time in succession, the Server will signify to the Client that a failure has occurred. The models of the Client and Server are taken from Larsen et al. (2006) (where they are referred to as *Client* and *TryTwice* respectively), so we may highlight the differentiating features of our work.

The combined behaviour of the Client and Server (i.e., Client || Server) is shown in Figure 11. To understand intuitively how the composition is derived, note that fail appears in the static interface of Client (Figure 9), yet Client assumes that fail will never be issued by the environment. It follows that Client || Server can never guarantee that there is a safe behaviour containing fail. Therefore, to prevent such a behaviour arising, the environment



$\mathcal{R}_{\mathsf{Client}||\mathsf{Server}}$

$\mathcal{G}_{\mathsf{Client}||\mathsf{Server}}$

Figure 11: Assumption and guarantee of Client||Server

must never issue the preceding nack, which will in turn prevent an implementation of Server from issuing fail to an implementation of the Client.

When contrasting Figure 11 with the parallel composition of Client and Server in Larsen et al. (2006) (where our Server corresponds to *TryTwice*), after accounting for the difference in parallel composition (whereby we do not automatically hide the actions that are shared between components, i.e, send, ok, and fail), one observes that our guarantee can be expressed in a simpler manner, given that it need not be input-enabled.

We now wish to construct a contract representing the behaviour of the communication link that transmits information from the Server. As a point of departure, we assume that the operation of the communication link is governed by a protocol, which is represented by the contract LinkLayer1 in Figure 12. The protocol awaits a transmission request (transmit), after which it attempts to deliver the data to the intended recipient. Suc-

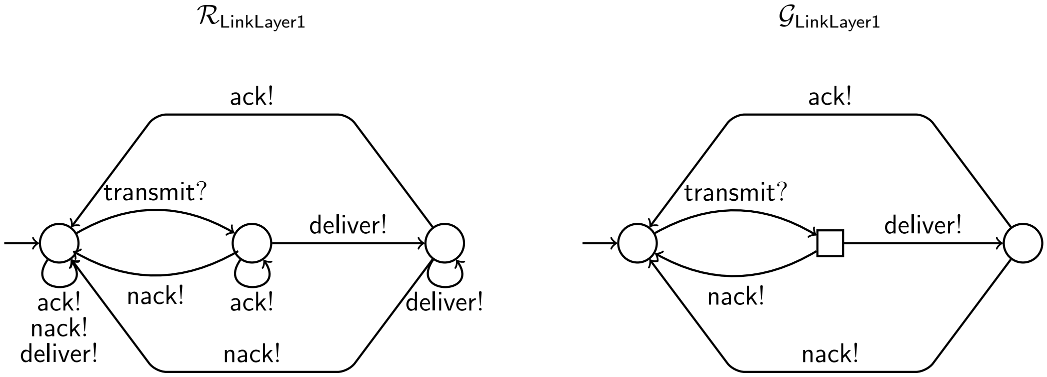


Figure 12: Contract for LinkLayer1

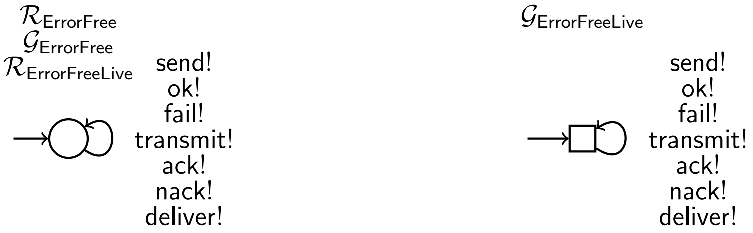
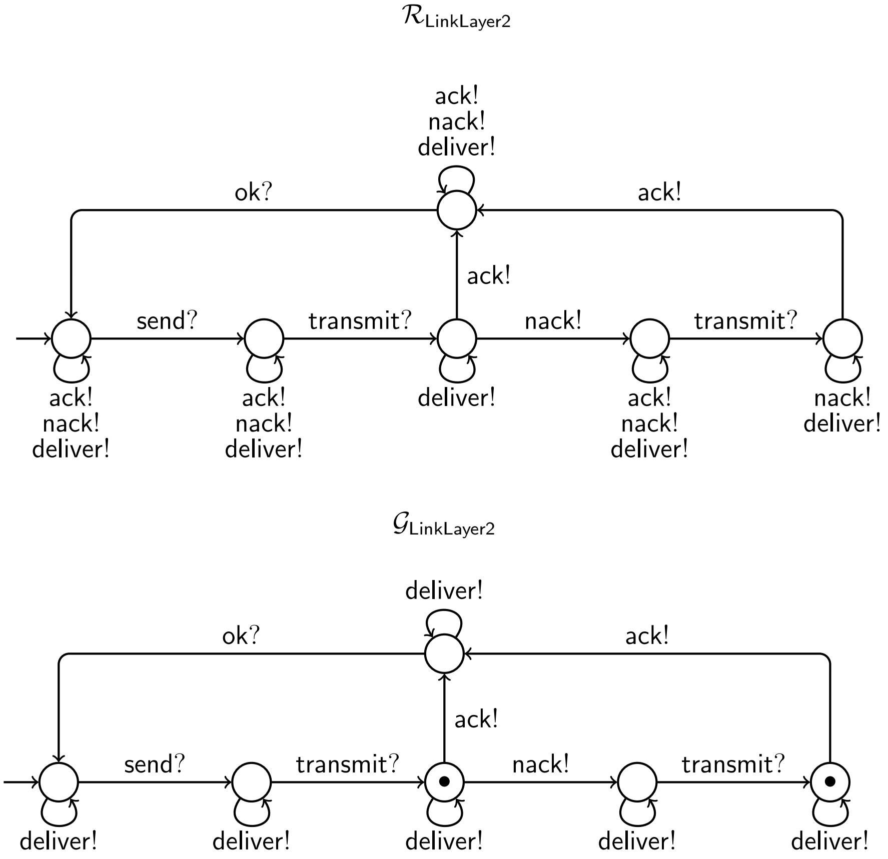

$\mathcal{R}_{\mathsf{ErrorFree}}$
$\mathcal{G}_{\mathsf{ErrorFree}}$
$\mathcal{R}_{\mathsf{ErrorFreeLive}}$
send!
ok!
fail!
transmit!
ack!
nack!
deliver!
$\mathcal{G}_{\mathsf{ErrorFreeLive}}$
send!
ok!
fail!
transmit!
ack!
nack!
deliver!

Figure 13: Assumption and guarantee for contracts ErrorFree and ErrorFreeLive

cessful delivery of the data results in a positive acknowledgment, while a nack occurs if deliver does not complete successfully, or if deliver cannot be performed for some reason. Unfortunately, the parallel composition of LinkLayer1 with Client || Server is a contract for which nothing can be assumed, meaning that no safety or progress properties can be inferred. To see why, note that the assumption of the composition must be empty because ⟨send, transmit, nack, transmit, nack⟩ is a trace over outputs whose projections onto Server||Client and LinkLayer1, respectively, are not contained in both assumptions, while they are also not in the respective error sets (cf. Definition 11 for parallel).

In order to formulate a stronger contract for the communication link, we use our theory to derive the weakest restrictions on the communication medium that allows all three of the Client, Server and link to communicate. This can be formulated as the quotient ErrorFree/(Client || Server), where ErrorFree is the component having a single chaotic state labelled by all actions, which should be treated as outputs (Figure 13, left-hand side). The only


$\mathcal{R}_{\mathsf{LinkLayer2}}$

$\mathcal{G}_{\mathsf{LinkLayer2}}$

Figure 14: Assumption and guarantee of LinkLayer2 with full interface

significant constraint imposed by ErrorFree is that no unspecified receptions will occur. Note that the state is not required to be live, and hence the assumption is the same as the guarantee.

The resulting contract, referred to as LinkLayer2, is depicted in Figure 14 when the set of input actions is taken to be $\{\mathsf{send}, \mathsf{transmit}, \mathsf{ok}\}$, whereas Figure 15 is the corresponding contract synthesised by the quotient operation when the set of inputs is taken to be $\{\mathsf{transmit}\}$. Recall from Remark 1 that the set of input actions to the quotient operator can be parameterised.

LinkLayer2 (parameterised on $\{\mathsf{transmit}\}$) is thus a contract that will allow Server and Client to interact with one another, but it may not respect the protocol of LinkLayer1, meaning that it may not meaningfully interact with the communication link. Therefore, we define $\mathsf{LinkLayer1} \wedge \mathsf{LinkLayer2}$ as the

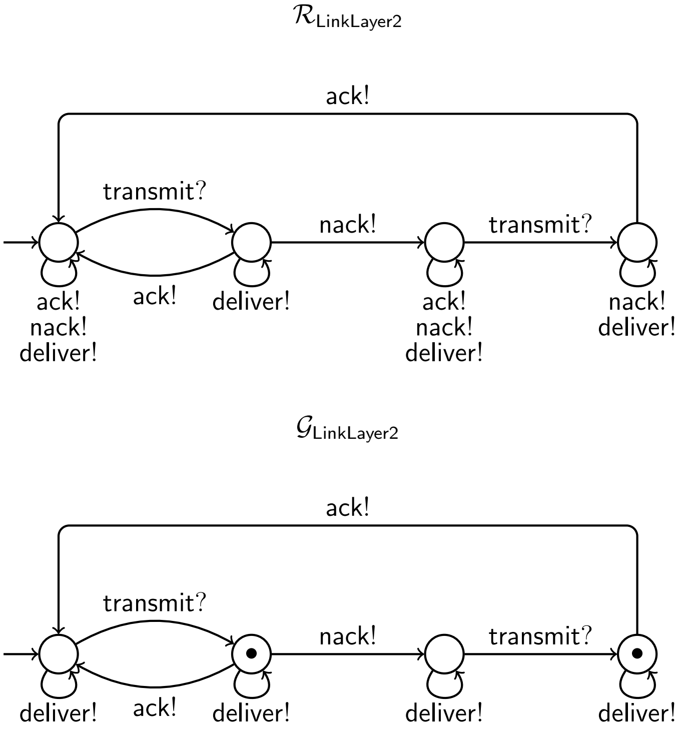


Figure 15: Assumption and guarantee of LinkLayer2 with restricted interface

contract for implementations that should communicate with the link (shown in Figure 16). Any implementation of this contract must never nack two transmissions in succession.

We now consider the impact of liveness, in addition to safety. Let ErrorFreeLive be the ErrorFree contract, but with the requirement that the sole state must be live (also shown in Figure 13, right-hand side). Then ErrorFreeLive/(Client || Server) is the contract in Figures 14 and 15, but with states containing • treated as though they are live (i.e., they should be squares). Similarly, LinkLayer1 ∧ LinkLayer2 is as depicted in Figure 16, but with the • filled nodes converted to squares. In the liveness setting, note that if, for some reason, the state following nack in Figure 10 is not live, then the specification in Figure 16 would not allow any nack at all, since it would lead to a state from which progress would not be guaranteed, thus conflicting with the requirements imposed by ErrorFreeLive.

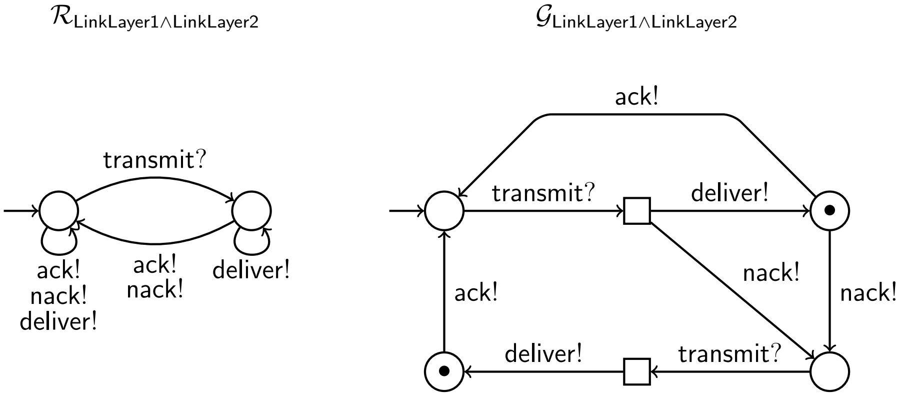

$\mathcal{R}_{\mathsf{LinkLayer1}\wedge\mathsf{LinkLayer2}}$
transmit?
ack!
nack!
deliver!
ack!
nack!
deliver!
$\mathcal{G}_{\mathsf{LinkLayer1}\wedge\mathsf{LinkLayer2}}$
ack!
transmit?
deliver!
nack!
ack!
nack!
deliver!
transmit?

Figure 16: Assumption and guarantee of LinkLayer1∧LinkLayer2

To summarise, this case study demonstrates how our framework adds significant flexibility over previous frameworks, such as the one by Larsen et al. (2006). Specifically, we provide a simpler formalism that does not require input-enabledness of guarantees, while supporting compositional reasoning not only for safety, but also liveness properties. A rich collection of operators are defined beyond those in Larsen et al. (2006). Our quotient operator facilitates the automated incremental construction of contracts for missing components, while conjunction combines independently developed requirements represented by multiple contracts. These features provide a range of additional checks for the validity of derived contracts, and support a truly contract-based design methodology.

5. Conclusion

We have presented a compositional specification theory for reasoning about safety and progress properties of component behaviours, where we explicitly separate the assumptions made on the environment's behaviour from the guarantees provided by the component. Our theory supports refinement based on traces, which relates specifications by implementation containment. We define the compositional operations of parallel composition, as well as – for the first time in this setting – conjunction, disjunction and quotient, directly on contracts. Sound and complete AG reasoning rules are provided for the four operators, preserving both safety and progress properties, which facilitates reasoning about, e.g., substitutivity of components synthesised at runtime. The theory can be extended with hiding, providing a proper treatment of divergence is given for components, as reported in (Chilton, 2013). Allowing divergence necessitates the extension of the contract framework to include sets of traces that must not diverge, in addition to the traces that must make progress. This is in contrast to works such as (Jonsson, 1994), which assume that a diverging process makes progress. We take a more pragmatic view in requiring that progress is observable. The AG rules can be fully automated, when restricting to regular properties (which can be represented by finite-state automata), as they are based on simple set-theoretic operations and do not require the learning of assumptions. The composition operations are polynomial-time constructions on finite automata, and the refinement relation can also be checked in polynomial-time, when the participating specifications are deterministic finite-state automata.

Acknowledgments. The authors are supported by EU FP7 project CONNECT, the ERC Advanced Grant VERIWARE, and the UPMARC centre of excellence.

References

Abadi, M., Lamport, L., 1993. Composing specifications. ACM Trans. on Programming Languages and Systems 15, 73–132.

Abadi, M., Lamport, L., 1995. Conjoining specifications. ACM Trans. on Programming Languages and Systems 17, 507–534.

Abadi, M., Plotkin, G., 1993. A logical view of composition. Theoretical Computer Science 114, 3–30.

de Alfaro, L., Henzinger, T.A., 2001. Interface automata. SIGSOFT Softw. Eng. Notes 26, 109–120.

Bauer, S., David, A., Hennicker, R., Larsen, K., Legay, A., Nyman, U., Wasowski, A., 2012. Moving from specifications to contracts in component-based design, in: Lara, J., Zisman, A. (Eds.), FASE'12. Springer. volume 7212 of *Lecture Notes in Computer Science*, pp. 43–58.

Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C., 2008. Multiple viewpoint contract-based specification and design, in: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.P. (Eds.), Proc. FMCO 2007, 6th Int. Symp. Formal Methods for Components and Objects, Amsterdam, The Netherlands, Springer. pp. 200–225.

Bhaduri, P., Ramesh, S., 2008. Interface synthesis and protocol conversion. Form. Asp. Comput. 20, 205–224.

Chen, T., Chilton, C., Jonsson, B., Kwiatkowska, M., 2012. A Compositional Specification Theory for Component Behaviours, in: Seidl, H. (Ed.), Programming Languages and Systems, Proc. 21st European Symposium on Programming (ESOP'12), Springer-Verlag. pp. 148–168.

Chilton, C., 2013. An Algebraic Theory of Componentised Interaction. Ph.D. thesis. Department of Computer Science, University of Oxford.

Chilton, C., Jonsson, B., Kwiatkowska, M., 2013a. An Algebraic Theory of Interface Automata. Technical Report CS-RR-13-02. Department of Computer Science, University of Oxford. Http://www.cs.ox.ac.uk/chris.chilton/publications/cs-rr-13-02.pdf.

Chilton, C., Jonsson, B., Kwiatkowska, M., 2013b. Assume-guarantee reasoning for safe component behaviours, in: Pasareanu, C., Salaün, G. (Eds.), Proc. 9th International Symposium on Formal Aspects of Component Software (FACS'12), Springer. pp. 92–109.

Clarke, E., Long, D., McMillan, K., 1989. Compositional model checking, in: Proc. 4th Annual Symposium on Logic in computer science, IEEE Press. pp. 353–362.

Collette, P., 1993. Application of the composition principle to Unity-like specifications, in: TAPSOFT'93, LNCS 668, Springer-Verlag. pp. 230–242.

Delahaye, B., Caillaud, B., Legay, A., 2011. Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects. FMSD 38, 1–32.

Doyen, L., Henzinger, T.A., Jobstmann, B., Petrov, T., 2008. Interface theories with component reuse, in: Proc. 8th ACM international conference on Embedded software, ACM. pp. 79–88.

Emmi, M., Giannakopoulou, D., Păsăreanu, C., 2008. Assume-Guarantee Verification for Interface Automata, in: Cuellar, J., Maibaum, T., Sere,

K. (Eds.), FM 2008: Formal Methods. Springer. volume 5014 of *Lecture Notes in Computer Science*, pp. 116–131.

Grumberg, O., Long, D.E., 1991. Model checking and modular verification. ACM Transactions on Programming Languages and Systems 16.

Jonsson, B., 1994. Compositional specification and verification of distributed systems. ACM Trans. on Programming Languages and Systems 16, 259–303.

Jonsson, B., Yih-Kuen, T., 1996. Assumption/guarantee specifications in linear-time temporal logic. Theoretical Computer Science 167, 47–72.

Larsen, K.G., Nyman, U., Wasowski, A., 2006. Interface input/output automata, in: FM 2006, Springer. pp. 82–97.

Larsen, K.G., Nyman, U., Wasowski, A., 2007. Modal I/O automata for interface and product line theories, in: Nicola, R.D. (Ed.), ESOP, Springer. pp. 64–79.

Maier, P., 2003. Compositional circular assume-guarantee rules cannot be sound and complete, in: Proc. 6th International conference on Foundations of Software Science and Computation Structures and joint European conference on Theory and practice of software, Springer. pp. 343–357.

Namjoshi, K.S., Trefler, R.J., 2010. On the completeness of compositional reasoning methods. ACM Trans. Comput. Logic 11, 16:1–16:22.

Pnueli, A., 1985. In transition from global to modular temporal reasoning about programs, in: Apt, K.R. (Ed.), Logics and models of concurrent systems. Springer, pp. 123–144.

Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R., 2011. A modal interface theory for component-based design. Fundam. Inform. 108, 119–149.

Appendix A. Proofs

Proof of Lemma 1

We show that $\mathcal{R}_{\mathcal{S}} \cap T_{\mathcal{Q}} \subseteq \mathcal{G}_{\mathcal{S}} \cap \overline{F_{\mathcal{Q}}}$. Let $t \in \mathcal{R}_{\mathcal{S}} \cap T_{\mathcal{Q}}$. From $\mathcal{Q} \sqsubseteq^{l}_{imp} \mathcal{P}$ it follows that $t \in T_{\mathcal{E}(\mathcal{P})} \cup (T_{\mathcal{E}(\mathcal{P})} \uparrow \mathcal{A}^{I}_{\mathcal{Q}})$. But, in fact, $t \in T_{\mathcal{E}(\mathcal{P})}$ as $\mathcal{A}^{I}_{\mathcal{S}} \subseteq \mathcal{A}^{I}_{\mathcal{P}} \subseteq$

$\mathcal{A}^I_\mathcal{Q}$, $t \in \mathcal{A}^*_\mathcal{S}$ and $\mathcal{A}^I_\mathcal{Q} \cap \mathcal{A}^O_\mathcal{S} = \emptyset$. Therefore, either $t \in T_\mathcal{P}$ or $t \in F_{\mathcal{E}(\mathcal{P})}$. For the former, $t \in \mathcal{R}_\mathcal{S} \cap T_\mathcal{P}$ implies $t \in \mathcal{G}_\mathcal{S} \cap \overline{F_{\mathcal{E}(\mathcal{P})}}$. As $t \notin F_{\mathcal{E}(\mathcal{P})}$ (and moreover $t \notin T_{\mathcal{E}(\mathcal{P})} \uparrow \mathcal{A}^I_\mathcal{Q}$) it follows that $t \notin F_{\mathcal{E}(\mathcal{Q})}$ since $\mathcal{Q} \sqsubseteq^l_{imp} \mathcal{P}$. Hence $t \in \mathcal{G}_\mathcal{S} \cap \overline{F_\mathcal{Q}}$ as required. If instead $t \in F_{\mathcal{E}(\mathcal{P})}$, then either $t \equiv \epsilon$, or there is some prefix $t'i$ of t with $i \in \mathcal{A}^I_\mathcal{P}$ such that $t' \notin F_{\mathcal{E}(\mathcal{P})}$ while $t'i \in F_{\mathcal{E}(\mathcal{P})}$. For both cases $\mathcal{P} \not\models \mathcal{S}$, which is contradictory (the latter because $t'i \in T_\mathcal{P}$).

Now suppose that $t \in \mathcal{L}_\mathcal{S} \cap T_\mathcal{Q}$. Then $t \in \mathcal{A}^*_\mathcal{P}$, so, from $\mathcal{Q} \sqsubseteq^l_{imp} \mathcal{P}$, we have $t \in T_{\mathcal{E}(\mathcal{P})}$. If $t \in T_\mathcal{P} \setminus F_{\mathcal{E}(\mathcal{P})}$, then from $\mathcal{P} \models \mathcal{S}$ we derive $t \in \overline{K_\mathcal{P}}$, thus $t \in \overline{K_\mathcal{Q}}$ from $\mathcal{Q} \sqsubseteq^l_{imp} \mathcal{P}$. If instead $t \in F_{\mathcal{E}(\mathcal{P})}$, then, by the same reasoning as previously, we see that $\mathcal{P} \not\models \mathcal{S}$.

Proof of Lemma 2

For the first claim, we show that $t \in X_i$ implies t is not a trace in any implementation of $\mathcal{S}$ for each $i \in \mathbb{N}$, where X_i is the i-th iteration of defining $\mathsf{error}(\mathcal{S})$ as a least fixed point. For $i = 0$, the result holds trivially as $X_0 = \emptyset$. So suppose that the result holds for $i = k$. Now $t \in X_{k+1}$ implies that $t \in \mathsf{violations}(\mathcal{S})$ or there is $t' \in (\mathcal{A}^I_\mathcal{S})^*$ such that $tt' \in \mathcal{L}_\mathcal{S}$ and $\forall o \in \mathcal{A}^O_\mathcal{S} \cdot tt'o \in X_k$. If $t \in \mathsf{violations}(\mathcal{S})$, then clearly t cannot be a trace of any implementation of $\mathcal{S}$, since condition S4 will not be satisfied. If instead t satisfies the second property, then it follows by the induction hypothesis that tt' is a quiescent trace, which contradicts $tt' \in \mathcal{L}_\mathcal{S}$. Therefore, tt' cannot be a trace of any implementation of $\mathcal{S}$, and so t also cannot be a trace, by input receptiveness of components. Taking $t \equiv \epsilon$, it follows that $\mathcal{S}$ is non-implementable.

For the second claim, suppose $t \in \mathcal{R}_\mathcal{S} \cap T_{\mathcal{I}(\mathcal{S})}$. Then $t \in \mathcal{R}_\mathcal{S} \cap \overline{\mathsf{error}(\mathcal{S})}$, which implies $t \in \mathcal{G}_\mathcal{S}$. Moreover, as $t \in \mathcal{R}_\mathcal{S}$, it follows that $t \in \overline{F_{\mathcal{I}(\mathcal{S})}}$. Hence S4 is satisfied. Now suppose that $t \in \mathcal{L}_\mathcal{S} \cap T_{\mathcal{I}(\mathcal{S})}$. Then clearly $t \notin K_{\mathcal{I}(\mathcal{S})}$ by definition, so S5 is satisfied.

For the third claim, the if direction follows by the previous claim and Lemma 1. For the only if direction, we need to show that $T_{\mathcal{E}(\mathcal{P})} \subseteq T_{\mathcal{I}(\mathcal{S})} \cup (T_{\mathcal{I}(\mathcal{S})} \uparrow \mathcal{A}^I_\mathcal{P})$, $F_{\mathcal{E}(\mathcal{P})} \subseteq F_{\mathcal{I}(\mathcal{S})} \cup (T_{\mathcal{I}(\mathcal{S})} \uparrow \mathcal{A}^I_\mathcal{P})$ and $K_{\mathcal{E}(\mathcal{P})} \subseteq K_{\mathcal{I}(\mathcal{S})} \cup (T_{\mathcal{I}(\mathcal{S})} \uparrow \mathcal{A}^I_\mathcal{P})$. If $t \in T_{\mathcal{E}(\mathcal{P})}$ and $t \notin \mathcal{A}^*_\mathcal{S}$, then there is a prefix $t'a$ of t such that $t' \in \mathcal{A}^*_\mathcal{S}$ and $a \in \mathcal{A}^I_\mathcal{P} \setminus \mathcal{A}_\mathcal{S}$, which by an inductive argument that assumes the result holds for all strict prefixes allows us to derive $t \in T_{\mathcal{I}(\mathcal{S})} \uparrow \mathcal{A}^I_\mathcal{P}$. So suppose that $t \in T_\mathcal{P} \cap \mathcal{A}^*_\mathcal{S}$. Then by the first claim, since $\mathcal{P} \models \mathcal{S}$, it follows $t \notin \mathsf{error}(\mathcal{S})$. Hence $t \in T_{\mathcal{I}(\mathcal{S})}$. Now suppose that $t \in F_{\mathcal{E}(\mathcal{P})} \cap \mathcal{A}^*_\mathcal{S}$. Then as $\mathcal{P} \models \mathcal{S}$, it follows that $t \notin \mathsf{error}(\mathcal{S})$ and $t \notin \mathcal{R}_\mathcal{S}$. Consequently, $t \in F_{\mathcal{I}(\mathcal{S})}$. Finally, suppose that $t \in K_\mathcal{P} \cap \mathcal{A}^*_\mathcal{S}$. Then as $t \in T_\mathcal{P}$ it follows that $t \notin \mathcal{L}_\mathcal{S}$, since $\mathcal{P} \models \mathcal{S}$. Hence, $t \in K_{\mathcal{I}(\mathcal{S})}$.

Proof of Lemma 3

For the only if direction, suppose $\mathcal{P} \models \mathcal{S}$ and $\mathcal{A}^I_{\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{T}} = \emptyset$. We first show that $\mathcal{P} \models \mathcal{T}$, so suppose $t \in \mathcal{R}_{\mathcal{T}} \cap T_{\mathcal{P}}$. Then, by the definition of $\sqsubseteq$, it follows that $t \in \mathcal{R}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$. If $t \in \mathsf{error}(\mathcal{S})$, then $t \notin T_{\mathcal{P}}$, since $\mathcal{P} \models \mathcal{S}$, which is contradictory. Therefore, $t \in \mathcal{R}_{\mathcal{S}}$, which from $\mathcal{P} \models \mathcal{S}$ implies $t \in \mathcal{G}_{\mathcal{S}} \cap \overline{F_{\mathcal{P}}}$. But as $t \notin \mathsf{error}(\mathcal{S})$, it follows that $t \notin \mathsf{error}(\mathcal{T})$ and so $t \in \mathcal{G}_{\mathcal{T}}$. Hence $t \in \mathcal{G}_{\mathcal{T}} \cap \overline{F_{\mathcal{P}}}$ as required. Now suppose that $t \in \mathcal{L}_{\mathcal{T}} \cap T_{\mathcal{P}}$. Then from $\mathcal{S} \sqsubseteq \mathcal{T}$ it follows that $t \in \mathcal{L}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$. If $t \in \mathcal{L}_{\mathcal{S}}$, then $t \in \overline{K_{\mathcal{P}}}$, since $\mathcal{P} \models \mathcal{S}$. If instead $t \in \mathsf{error}(\mathcal{S})$, then $\mathcal{P} \not\models \mathcal{S}$, which is contradictory. Hence, $\mathcal{P} \models \mathcal{T}$ as required.

For the if direction, Lemmas 1 and 2 allow us to conclude that $\mathcal{I}(\mathcal{S}) \sqsubseteq^l_{imp} \mathcal{I}(\mathcal{T})$. Suppose that $t \in \mathsf{error}(\mathcal{T}) \cap \mathcal{A}^*_{\mathcal{S}}$. Then $t \notin T_{\mathcal{I}(\mathcal{T})}$, hence $t \notin T_{\mathcal{I}(\mathcal{S})}$, meaning $t \in \mathsf{error}(\mathcal{S})$. Now suppose that $t \in \mathcal{R}_{\mathcal{T}} \cap \mathcal{A}^*_{\mathcal{S}}$. Then $t \notin F_{\mathcal{I}(\mathcal{T})}$, which implies $t \notin F_{\mathcal{I}(\mathcal{S})}$, hence $t \notin \overline{\mathcal{R}_{\mathcal{S}}} \cap \overline{\mathsf{error}(\mathcal{S})}$ i.e., $t \in \mathcal{R}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$. Finally, suppose that $t \in \mathcal{L}_{\mathcal{T}} \cap \mathcal{A}^*_{\mathcal{S}}$. Then $t \notin K_{\mathcal{I}(\mathcal{T})}$, hence $t \notin K_{\mathcal{I}(\mathcal{S})}$. Thus $t \notin \overline{\mathcal{L}_{\mathcal{S}}} \cap \overline{\mathsf{error}(\mathcal{S})}$, and so $t \in \mathcal{L}_{\mathcal{S}} \cup \mathsf{error}(\mathcal{S})$ as required.

Proof of Lemma 4

Essentially follows from transitivity of $\subseteq$.

Proof of Lemma 5

For the first claim, let $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})} \cap T_{\mathcal{P}}$. Then, as $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})}$, it follows $t \in \overline{F_{\mathcal{E}(\mathcal{P})}}$. Given $t \in T_{\mathcal{P}}$, it thus follows $t \in \mathcal{G}_{\mathcal{AG}(\mathcal{P})} \cap \overline{F_{\mathcal{E}(\mathcal{P})}}$ as required. Furthermore, if $t \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})} \cap T_{\mathcal{P}}$, then $t \notin K_{\mathcal{E}(\mathcal{P})}$, hence $t \notin K_{\mathcal{P}}$.

For the second claim, the if direction follows by the previous claim and Lemma 3. For the only if direction, suppose that $t \in \mathsf{error}(\mathcal{S}) \cap \mathcal{A}^*_{\mathcal{P}}$. Hence $t \notin T_{\mathcal{P}} \cup F_{\mathcal{E}(\mathcal{P})}$ as $\mathcal{P} \models \mathcal{S}$, which implies $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})} \cap \overline{\mathcal{G}_{\mathcal{AG}(\mathcal{P})}}$. Hence, $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$. Now suppose that $t \in \mathcal{R}_{\mathcal{S}} \cap \mathcal{A}^*_{\mathcal{P}}$. Then $\mathcal{P} \models \mathcal{S}$ implies $t \notin T_{\mathcal{P}}$ or $t \notin F_{\mathcal{E}(\mathcal{P})}$. Note that $t \notin T_{\mathcal{P}}$ implies $t \notin F_{\mathcal{E}(\mathcal{P})}$ (consider a prefix in $T_{\mathcal{P}} \cap F_{\mathcal{E}(\mathcal{P})}$). Hence $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})}$. Finally, suppose that $t \in \mathcal{L}_{\mathcal{S}} \cap \mathcal{A}^*_{\mathcal{P}}$. Then from $\mathcal{P} \models \mathcal{S}$, it follows that $t \notin T_{\mathcal{P}}$ or $t \notin K_{\mathcal{P}}$. In the case of the former, $t \notin F_{\mathcal{E}(\mathcal{P})}$ as $\mathcal{P} \models \mathcal{S}$, so $t \in \mathsf{violations}(\mathcal{AG}(\mathcal{P}))$, which implies $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$. For the latter, if $t \notin \mathcal{L}_{\mathcal{AG}(\mathcal{P})}$, then $t \notin T_{\mathcal{P}}$ or $t \in T_{\mathcal{P}} \cap F_{\mathcal{E}(\mathcal{P})}$, both of which imply $t \in \mathsf{violations}(\mathcal{AG}(\mathcal{P}))$, and so $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$.

Appendix B. Parallel Composition

The following lemma is useful to the proof of parallel compositionality and for establishing the properties satisfied by the quotient operator (which is the adjoint of parallel). It essentially states that an error in the parallel composition of two contracts must manifest itself as an error in at least one of the contracts to be composed.

Lemma 6. *$t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$ implies $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ or $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$.*

Proof. Show that $t \in X_i$ implies $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ or $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$, where X_i is the i-th iteration of $\mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$ defined as a least fixed point. When $i = 0$, the result hold trivially, since $X_0 = \emptyset$. So suppose that $t \in X_{k+1}$. Then $t \in \mathsf{violations}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, or there exists $t' \in (\mathcal{A}^I_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})^*$ such that $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ and $\forall o \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cdot tt'o \in X_k$. If $t \in \mathsf{violations}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, then there exists a prefix and input extension $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}}$. So, without loss of generality, $t' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ by the definition of $||$, from which it follows $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$. For the latter case, without loss of generality suppose that $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$. If $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$, then it follows that there exists $o' \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cdot tt'o' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$. As $tt'o' \in X_k$, it follows that $tt'o' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. Moreover, as $o' \notin \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$, it follows that $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$ as required. □

Proof of Theorem 1

Note that the alphabet constraints are satisfied, so first show $\mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}} \subseteq \mathcal{R}_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}'_\mathcal{P} \mid\mid \mathcal{S}'_\mathcal{Q})$. Suppose $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}}$, and all strict prefixes of t are in $\mathcal{R}_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}} \cap \overline{\mathsf{error}(\mathcal{S}'_\mathcal{P} \mid\mid \mathcal{S}'_\mathcal{Q})}$. If $t \notin \mathcal{R}_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}}$, then there exists $t' \in (\mathcal{A}^O_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}})^*$ such that, wlog, $tt' \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}'_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}'_\mathcal{P})$ and $tt' \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{Q}} \notin \mathsf{error}(\mathcal{S}'_\mathcal{Q})$. As $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} = tt' \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{P}}$ and $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = tt' \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{Q}}$, it follows that $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P})$ since $\mathcal{S}'_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P}$, and $tt' \restriction \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \notin \mathsf{error}(\mathcal{S}_\mathcal{Q})$ since $\mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$. Hence, $tt' \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$, which implies $t \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ as $t' \in (\mathcal{A}^O_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}})^*$, but this is contradictory.

Now suppose that $t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q}) \cap \mathcal{A}^*_{\mathcal{S}'_\mathcal{P}||\mathcal{S}'_\mathcal{Q}}$. Then by Lemma 6 it follows that, without loss of generality, $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathsf{error}(\mathcal{S}_\mathcal{P})$. Since $t \restriction \mathcal{A}_{\mathcal{S}_\mathcal{P}} = t \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{P}}$, it follows from $\mathcal{S}'_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P}$ that $t \restriction \mathcal{A}_{\mathcal{S}'_\mathcal{P}} \in \mathsf{error}(\mathcal{S}'_\mathcal{P})$. Now from the first part, we derive $t \in \mathcal{R}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}} \cup \mathsf{error}(\mathcal{S}'_{\mathcal{P}} \mid\mid \mathcal{S}'_{\mathcal{Q}})$, so it follows that $t \in \mathsf{error}(\mathcal{S}'_{\mathcal{P}} \mid\mid \mathcal{S}'_{\mathcal{Q}})$, since certainly $t \notin \mathcal{G}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$.

Finally, show $t \in \mathcal{L}_{\mathcal{S}_{\mathcal{P}}||\mathcal{S}_{\mathcal{Q}}} \cap \mathcal{A}^*_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$ implies $t \in \mathcal{L}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}} \cup \mathsf{error}(\mathcal{S}'_{\mathcal{P}} \mid\mid \mathcal{S}'_{\mathcal{Q}})$. Suppose that $t \notin \mathsf{error}(\mathcal{S}'_{\mathcal{P}} \mid\mid \mathcal{S}'_{\mathcal{Q}})$. Then by the first part, as $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}||\mathcal{S}_{\mathcal{Q}}} \cap \mathcal{A}^*_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$, it follows that $t \in \mathcal{R}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$, and so $t \in \mathcal{G}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$. Hence $t \upharpoonright \mathcal{A}_{\mathcal{S}'_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}'_{\mathcal{P}})$ and $t \upharpoonright \mathcal{A}_{\mathcal{S}'_{\mathcal{Q}}} \notin \mathsf{error}(\mathcal{S}'_{\mathcal{Q}})$. Now, without loss of generality, $t \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \in \mathcal{L}_{\mathcal{S}_{\mathcal{P}}}$, so from $\mathcal{S}'_{\mathcal{P}} \sqsubseteq \mathcal{S}_{\mathcal{P}}$, it follows that $t \upharpoonright \mathcal{A}_{\mathcal{S}'_{\mathcal{P}}} \in \mathcal{L}_{\mathcal{S}'_{\mathcal{P}}}$. Hence $t \in \mathcal{L}_{\mathcal{S}'_{\mathcal{P}}||\mathcal{S}'_{\mathcal{Q}}}$ as required.

Proof of Theorem 2

For soundness, we know $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}_{\mathcal{P}}$ and $\mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_{\mathcal{Q}}$. By the theorem conditions, the conditions for Theorem 1 are satisfied, so $\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}}$. From compatibility of $\mathcal{P} \mid\mid \mathcal{Q}$ and $\mathcal{S}$, we obtain $\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}$ by weak transitivity. Now by Lemma 7 (an ancillary result, following) we derive $\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}) \sqsubseteq \mathcal{S}$ by transitivity, given that the alphabets of $\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q})$ coincide with those of $\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q})$.

For completeness, take $\mathcal{S}_{\mathcal{P}} = \mathcal{AG}(\mathcal{P})$ and $\mathcal{S}_{\mathcal{Q}} = \mathcal{AG}(\mathcal{Q})$. Then by transitivity, the result follows from Lemma 7.

Lemma 7. *$\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}) \sqsubseteq \mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q})$.*

Proof. First suppose that $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$ and $t \notin \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$. Then $t \upharpoonright \mathcal{A}_{\mathcal{P}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})}$ and $t \upharpoonright \mathcal{A}_{\mathcal{Q}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{Q})}$, which implies that $t \upharpoonright \mathcal{A}_{\mathcal{P}} \notin F_{\mathcal{E}(\mathcal{P})}$ and $t \upharpoonright \mathcal{A}_{\mathcal{Q}} \notin F_{\mathcal{E}(\mathcal{Q})}$. Hence, $t \notin F_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}$, from which it follows that $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. For the other direction, suppose $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$ and $t \notin \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$. Then, $t \in \mathcal{G}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$, which implies $t \in T_{\mathcal{P}||\mathcal{Q}} \setminus F_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}$, which means that $t \upharpoonright \mathcal{A}_{\mathcal{P}} \notin F_{\mathcal{E}(\mathcal{P})}$ and $t \upharpoonright \mathcal{A}_{\mathcal{Q}} \notin F_{\mathcal{E}(\mathcal{Q})}$ i.e., $t \upharpoonright \mathcal{A}_{\mathcal{P}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})}$ and $t \upharpoonright \mathcal{A}_{\mathcal{Q}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{Q})}$. From this it follows that $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$, having noticed that no output extension of t can violate this constraint.

For the error set containments, suppose that $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$ and $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})} \cap \mathcal{R}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. We demonstrate that $X_i \subseteq \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$ for each $i \in \mathbb{N}$, where X_i is the i-th iteration of defining the least fixed point characterising $\mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$. The result holds trivially when $i = 0$, since $X_i = \emptyset$. For the inductive case, suppose $t \in X_{k+1}$. Then $t \in \mathsf{violations}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$, or there exists $t' \in (\mathcal{A}^I_{\mathcal{P}||\mathcal{Q}})^*$ such that $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$ and for each $o \in \mathcal{A}^O_{\mathcal{P}||\mathcal{Q}}$ it holds that $tt'o \in X_k$. For the former, it follows that there exists $t' \in (\mathcal{A}^I_{\mathcal{P}||\mathcal{Q}})^*$ such that $tt' \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})} \cap$

$\overline{\mathcal{G}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}}$. Consequently, wlog, $tt' \restriction \mathcal{A}_{\mathcal{P}} \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$, which implies $t \restriction \mathcal{A}_{\mathcal{P}} \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$. Suppose for a contradiction that $t \in \mathcal{G}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. Then $t \in T_{\mathcal{P}||\mathcal{Q}} \setminus F_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}$, which implies $t \restriction \mathcal{A}_{\mathcal{P}} \in T_{\mathcal{P}}$. But, as $t \restriction \mathcal{A}_{\mathcal{P}} \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$, it follows that $\mathcal{P} \not\models \mathcal{AG}(\mathcal{P})$, which is contradictory. Therefore, $t \notin \mathcal{G}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$ and so $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$. For the latter case, by the induction hypothesis we have that $tt'o \in \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$, which implies that $tt' \in \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$, given that $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$ implies $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. To see this last implication, from $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$ it holds wlog that $tt' \restriction \mathcal{A}_{\mathcal{P}} \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})}$ and $tt' \restriction \mathcal{A}_{\mathcal{Q}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{Q})}$, since $tt' \restriction \mathcal{A}_{\mathcal{Q}} \notin \mathsf{error}(\mathcal{AG}(\mathcal{Q}))$. Hence, $tt' \restriction \mathcal{A}_{\mathcal{P}} \in T_{\mathcal{P}} \setminus K_{\mathcal{E}(\mathcal{P})}$ and $tt' \restriction \mathcal{A}_{\mathcal{Q}} \in T_{\mathcal{Q}} \cap \overline{F_{\mathcal{E}(\mathcal{Q})}}$. Thus, $tt' \in T_{\mathcal{P}||\mathcal{Q}} \setminus K_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}$, implying $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. Consequently, $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$ as required.

For the other direction of the containment, suppose $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$ and $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})} \cap \mathcal{R}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$. Using a similar X_i argument it follows that $t \in \mathsf{violations}(\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}))$, or there exists $t' \in (\mathcal{A}^I_{\mathcal{P}||\mathcal{Q}})^*$ such that $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$ and for each $o \in \mathcal{A}^O_{\mathcal{P}||\mathcal{Q}}$, it holds that $tt'o \in \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$. For the former, suppose that there exists $t' \in (\mathcal{A}^I_{\mathcal{P}||\mathcal{Q}})^*$ such that $tt' \in \mathcal{R}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})} \cap \overline{\mathcal{G}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}}$. Then $tt' \notin \overline{T_{\mathcal{P}||\mathcal{Q}} \cup F_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}}$, which implies wlog that $tt' \restriction \mathcal{A}_{\mathcal{P}} \notin T_{\mathcal{P}} \cup F_{\mathcal{E}(\mathcal{P})}$. Hence, $tt' \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})} \cap \overline{\mathcal{G}_{\mathcal{AG}(\mathcal{P})}}$, which implies $tt' \restriction \mathcal{A}_{\mathcal{P}} \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$. Therefore, $t \restriction \mathcal{A}_{\mathcal{P}} \in \mathsf{error}(\mathcal{AG}(\mathcal{P}))$, which implies $t \notin \mathcal{G}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$. Consequently, $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$ as we are assuming that $t \in \mathcal{R}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})}$. For the latter, from $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P}||\mathcal{Q})}$, it follows that $tt' \in T_{\mathcal{P}||\mathcal{Q}} \setminus K_{\mathcal{E}(\mathcal{P}||\mathcal{Q})}$. Consequently, without loss of generality, $tt' \restriction \mathcal{A}_{\mathcal{P}} \in T_{\mathcal{P}} \setminus K_{\mathcal{E}(\mathcal{P})}$ and $tt' \restriction \mathcal{A}_{\mathcal{Q}} \in T_{\mathcal{Q}} \setminus F_{\mathcal{E}(\mathcal{Q})}$. This means that $tt' \restriction \mathcal{A}_{\mathcal{P}} \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})}$ and $tt' \restriction \mathcal{A}_{\mathcal{Q}} \in \mathcal{R}_{\mathcal{AG}(\mathcal{Q})}$. Thus $tt' \in \mathcal{L}_{\mathcal{AG}(\mathcal{P})||\mathcal{AG}(\mathcal{Q})} \cup \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$. Either way, we derive $t \in \mathsf{error}(\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}))$.

The reasoning for the liveness set containments can be extracted from the error set containments mentioned previously. □

Appendix C. Conjunction

Proof of Theorem 3

First show that $\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{P}}$. Suppose $t \in \mathsf{error}(\mathcal{S}_{\mathcal{P}}) \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$. Then there is a prefix t' of t such that $t' \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$ and $t' \in \mathsf{error}(\mathcal{S}_{\mathcal{P}})$. Therefore, $t' \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}} \cap \overline{\mathcal{G}_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}}$, implying $t \in \mathsf{error}(\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}})$. If $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$, then $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$ as required. Finally, suppose $t \in \mathcal{L}_{\mathcal{S}_{\mathcal{P}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$. As $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$, it follows that $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}}}$. Moreover, if $t \notin \mathsf{error}(\mathcal{S}_{\mathcal{P}} \wedge \mathcal{S}_{\mathcal{Q}})$, then $t \in \mathcal{G}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$. So from $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, it is easy to see that $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ as required. By similar reasoning $\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$.

For the second claim, we show $\mathsf{error}(\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{R}} \subseteq \mathsf{error}(\mathcal{S}_\mathcal{R})$ by demonstrating that $t \in X_i \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{R}}$ implies $t \in \mathsf{error}(\mathcal{S}_\mathcal{R})$ by induction on i, where X_i is the i-th iteration of defining $\mathsf{error}(\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q})$ as a least fixed point. When $i = 0$ the result holds trivially as $X_i = \emptyset$. Now suppose $i = k$ for $k > 0$. If $t \in \mathsf{violations}(\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q})$, then there is a prefix t' of t and input extension $t'' \in (\mathcal{A}^I_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}})^*$ such that $t't'' \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}}$. So without loss of generality, $t't'' \notin \overline{\mathsf{error}(\mathcal{S}_\mathcal{P})} \cup (\overline{\mathsf{error}(\mathcal{S}_\mathcal{P})} \uparrow \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}})$. This means that there is a prefix of $t't''$ contained in $\mathsf{error}(\mathcal{S}_\mathcal{P})$, which must also be in $\mathsf{error}(\mathcal{S}_\mathcal{R})$ since $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P}$. If instead there exists $t' \in (\mathcal{A}^I_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}})^*$ such that $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ and $\forall o \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \cdot tt'o \in X_{i-1}$, then $\forall o' \in \mathcal{A}^O_{\mathcal{S}_\mathcal{R}}$ it follows that $tt'o' \in \mathsf{error}(\mathcal{S}_\mathcal{R})$ by the induction hypothesis. Moreover, from $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$, it follows that without loss of generality, $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$. So from $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P}$ we derive $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{R}} \cup \mathsf{error}(\mathcal{S}_\mathcal{R})$. But $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{R}}$ also implies $tt' \in \mathsf{error}(\mathcal{S}_\mathcal{R})$, hence $t \in \mathsf{error}(\mathcal{S}_\mathcal{R})$ as required. Now suppose that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{R}}$. Then without loss of generality, $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{R}}$, so from $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P}$, we derive $t \in \mathcal{R}_{\mathcal{S}_\mathcal{R}} \cup \mathsf{error}(\mathcal{S}_\mathcal{R})$. Finally, suppose $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{R}}$. If $t \notin \mathsf{error}(\mathcal{S}_\mathcal{R})$, then we have $t \in \mathcal{R}_{\mathcal{S}_\mathcal{R}} \cap \mathcal{G}_{\mathcal{S}_\mathcal{R}}$, since $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$ implies $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}}$, which implies $t \in \mathcal{R}_{\mathcal{S}_\mathcal{R}}$. Without loss of generality, $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, so from $\mathcal{S}_\mathcal{R} \sqsubseteq \mathcal{S}_\mathcal{P}$ it follows that $t \in \mathcal{L}_{\mathcal{S}_\mathcal{R}}$ as required.

For the third claim, by the first claim we have $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}'_\mathcal{P}$ and $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}'_\mathcal{Q}$. Now by transitivity, we see that $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P}$ and $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{Q}$ providing $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}'_\mathcal{Q}} = \emptyset$ and $\mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^I_{\mathcal{S}'_\mathcal{P}} = \emptyset$, so by the second claim, it follows that $\mathcal{S}'_\mathcal{P} \wedge \mathcal{S}'_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$ as required. If either of the compatibility conditions are not satisfied, we can obtain new contracts $\mathcal{S}''_\mathcal{P}$ for $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}''_\mathcal{Q}$ for $\mathcal{S}_\mathcal{Q}$ that have output set $\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}$ and contain all of the traces from the respective contracts, except for those with an output in $(\mathcal{A}^O_{\mathcal{S}_\mathcal{P}} \setminus \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}}) \cup (\mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} \setminus \mathcal{A}^O_{\mathcal{S}_\mathcal{P}})$ that has been removed from the interface. It is straightforward to show that $\mathcal{S}''_\mathcal{P} \wedge \mathcal{S}''_\mathcal{Q} = \mathcal{S}_\mathcal{P} \wedge \mathcal{S}_\mathcal{Q}$.

Proof of Theorem 4

For soundness, note by the second claim of Theorem 3 that $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}_1 \wedge \mathcal{S}_2$. Hence $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}$, as the compatibility constraint for weak transitivity is satisfied. For completeness, the result follows by idempotence of conjunction, having taken $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}$.

Appendix D. Disjunction

Proof of Theorem 5

For the first claim of $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}$, we first show that $\mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P}} \subseteq \mathsf{error}(\mathcal{S}_\mathcal{P})$. So let X_i be the i-th approximation of $\mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$ defined as a fixed point. Then by induction on i, we show that $X_i \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P}} \subseteq \mathsf{error}(\mathcal{S}_\mathcal{P})$. Suppose that $t \in X_{k+1} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$. If $t \in \mathsf{violations}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$, then there is a prefix t' of t such that $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}} \cap \overline{\mathcal{G}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}}$. Hence $t' \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ and so $t \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ as required. Otherwise, there is a trace $t' \in (\mathcal{A}^I_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}})^*$ such that $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ and for all $o \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ it holds that $tt'o \in X_k$. Consequently, as $t' \in (\mathcal{A}_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}_{\mathcal{S}_\mathcal{Q}})^*$, it follows that $tt' \in \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$, and so $tt' \in \mathcal{L}_{\mathcal{S}_\mathcal{P}}$. As a result, $tt' \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ since $tt'o' \in \mathsf{error}(\mathcal{S}_\mathcal{P})$ for each $o' \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P}}$ by the induction hypothesis. From this we derive $t \in \mathsf{error}(\mathcal{S}_\mathcal{P})$. Now suppose that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$. Then $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P})$ by definition. Similarly, if $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$, then $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P})$ as required. Showing $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}$ is similar.

For the second claim, suppose that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{R}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. If $t \equiv \epsilon$, then $\epsilon \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ trivially, while if $t \equiv t'o$ for $o \in \mathcal{A}^O_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$, then $t'o \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ by the induction hypothesis and output extendability of assumptions or extendability of $\mathsf{violations}/\mathsf{error}$. Instead, if $t \equiv t'i$ for $i \in \mathcal{A}^I_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$, then by the induction hypothesis in the difficult case we have $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cap \overline{\mathsf{error}(\mathcal{S}_\mathcal{P})}$ and $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{Q}} \cap \overline{\mathsf{error}(\mathcal{S}_\mathcal{Q})}$. As $i \in \mathcal{A}^I_{\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^I_{\mathcal{S}_\mathcal{Q}}$, it follows from $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{R}$ and $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{R}$ that $t'i \in \mathcal{R}_{\mathcal{S}_\mathcal{P}} \cap \mathcal{R}_{\mathcal{S}_\mathcal{Q}}$. Hence, $t'i \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$.

Now suppose that $t \in \mathsf{error}(\mathcal{S}_\mathcal{R}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. Then there exists a smallest prefix t' of t such that $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{R}} \cap \mathsf{error}(\mathcal{S}_\mathcal{R}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. Suppose all strict prefixes of t' are not in $\mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$. Then by the previous part, it follows that $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. If $t' \in \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$, then from $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{R}$ it follows that $t' \in \mathsf{error}(\mathcal{S}_\mathcal{P})$, and if $t' \in \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$, then from $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{R}$ it follows that $t' \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. Hence $t' \notin \mathcal{G}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ (noting $\mathcal{G}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}} \subseteq \mathcal{A}^*_{\mathcal{S}_\mathcal{P}} \cup \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$), which implies $t' \in \mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$. By extension closure of error, we have $t \in \mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$.

For the progress condition, suppose $t \in \mathcal{L}_{\mathcal{S}_\mathcal{R}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. Assuming $t \notin \mathsf{error}(\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q})$, we can infer that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}} \cap \mathcal{G}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. Suppose for a contradiction that $t \notin \mathcal{L}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$. Then since $t \in \mathcal{R}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$, it follows that $t \in \mathcal{A}^*_{\mathcal{S}_\mathcal{P}}$ and $t \notin \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, or $t \in \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$ and $t \notin \mathcal{L}_{\mathcal{S}_\mathcal{Q}}$. However, both of these contradict $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{R}$ and $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P}$. Hence $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}}$ as required.

For the third claim, by the first claim we have that $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}$ and $\mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{P} \vee \mathcal{S}_\mathcal{Q}$. Since the contracts under consideration are composable for disjunction, it follows from $\mathcal{S}'_{\mathcal{P}} \sqsubseteq \mathcal{S}_{\mathcal{P}}$ and $\mathcal{S}'_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{Q}}$, along with transitivity (compatibility holds), that $\mathcal{S}'_{\mathcal{P}} \sqsubseteq \mathcal{S}_{\mathcal{P}} \vee \mathcal{S}_{\mathcal{Q}}$ and $\mathcal{S}'_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{P}} \vee \mathcal{S}_{\mathcal{Q}}$. Now by the second claim it is straightforward to derive $\mathcal{S}'_{\mathcal{P}} \vee \mathcal{S}'_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{P}} \vee \mathcal{S}_{\mathcal{Q}}$.

Proof of Theorem 6

For soundness, assume $\mathcal{P} \models \mathcal{S}_1$. Then $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}_1$ and $\mathcal{S}_1 \sqsubseteq \mathcal{S}_1 \vee \mathcal{S}_2$ by Theorem 5. Since $\mathcal{A}^I_{\mathcal{P}} \cap \mathcal{A}^O_{\mathcal{S}} = \emptyset$, it follows that transitivity holds, and so $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}$, implying $\mathcal{P} \models \mathcal{S}$. For completeness, take $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}$. The result then holds by idempotence of $\vee$.

Appendix E. Quotient

Proof of Theorem 7

For the first claim, if $\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{W}}$, then $\mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}||\mathcal{S}_{\mathcal{Q}}} = \mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}} \cup \mathcal{A}^O_{\mathcal{S}_{\mathcal{Q}}} \subseteq \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}}$, which implies $\mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}} \subseteq \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}}$. Now suppose that $\mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}} \subseteq \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}}$. Then we construct a contract $\mathcal{S}_{\mathcal{Q}} = \langle \mathcal{A}^I_{\mathcal{S}_{\mathcal{W}}}, \mathcal{A}^O_{\mathcal{S}_{\mathcal{W}}} \setminus \mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}}, \mathcal{A}^*_{\mathcal{S}_{\mathcal{Q}}}, \emptyset, \emptyset \rangle$, which, having no implementations, implies $\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}}$ has no implementations. The constraints R1 to R3 are satisfied, so $\mathcal{S}_{\mathcal{P}} \mid\mid \mathcal{S}_{\mathcal{Q}} \sqsubseteq \mathcal{S}_{\mathcal{W}}$ as required.

For the second claim, suppose $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. If $t \notin \mathcal{R}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$, then there exists a prefix t' of t and $t'' \in (\mathcal{A}^O_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})})^*$ such that $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \notin \mathcal{R}_{\mathcal{S}_{\mathcal{P}}}$ or $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \notin \mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$, and $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{P}})$ and $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})$. It follows that $t't'' \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}}$, so $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$, which means $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \notin \mathcal{R}_{\mathcal{S}_{\mathcal{P}}}$. Therefore, $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \notin \mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$, which implies $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \in \mathsf{violations}(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})$. But this contradicts $t't'' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})$. Hence $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$.

Now suppose that $t \in \mathsf{error}(\mathcal{S}_{\mathcal{W}}) \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. Then, there exists a prefix t' of t such that $t' \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}} \cap \mathsf{error}(\mathcal{S}_{\mathcal{W}})$. By the previous part, it follows that $t' \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. Now suppose for a contradiction that $t' \in \mathcal{G}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. Then $t' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{P}})$ and $t' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})$. But it follows that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \in \mathsf{violations}(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})$, since $t' \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \cap \overline{\mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}}$. This contradicts $t' \in \mathcal{G}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. Hence $t' \in \mathsf{error}(\mathcal{S}_{\mathcal{P}} \mid\mid (\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}))$ and so $t \in \mathsf{error}(\mathcal{S}_{\mathcal{P}} \mid\mid (\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}))$.

Finally, suppose that $t \in \mathcal{L}_{\mathcal{S}_{\mathcal{W}}} \cap \mathcal{A}^*_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$, and $t \notin \mathsf{error}(\mathcal{S}_{\mathcal{P}} \mid\mid (\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}))$. Then by the previous part, $t \notin \mathsf{error}(\mathcal{S}_{\mathcal{W}})$, so $t \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})} \cap \mathcal{G}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$. Hence $t \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \in \mathcal{R}_{\mathcal{S}_{\mathcal{P}}} \cap \overline{\mathsf{error}(\mathcal{S}_{\mathcal{P}})}$ and $t \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \in \mathcal{R}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}} \cap \mathcal{G}_{\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}}}$. If $t \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \in \mathcal{L}_{\mathcal{S}_{\mathcal{P}}}$, then $t \in \mathcal{L}_{\mathcal{S}_{\mathcal{P}}||(\mathcal{S}_{\mathcal{W}}/\mathcal{S}_{\mathcal{P}})}$ as required, since $t \upharpoonright \mathcal{A}_{\mathcal{S}_{\mathcal{P}}} \notin \mathsf{error}(\mathcal{S}_{\mathcal{P}})$

implies $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \in \mathcal{G}_{\mathcal{S}_\mathcal{P}}$. If instead $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{L}_{\mathcal{S}_\mathcal{P}}$, then $t \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$, which implies $t \in \mathcal{L}_{\mathcal{S}_\mathcal{P}||(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})}$ as required.

For the third claim, suppose that $t \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$. Then there exists $t' \in \mathcal{A}^*_{\mathcal{S}_\mathcal{W}}$ such that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = t$ with $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ and $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$. From $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ we derive $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, given that $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$. If $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$, then it follows that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathcal{R}_{\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{Q})$. If instead $t' \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, then it follows that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$ by Lemma 6. Note that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = t$.

Now suppose that $t \in \mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}) \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$. Then there exists a prefix t' of t such that $t' \in \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$. We show that $X_i \cap \mathcal{R}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}} \subseteq \mathsf{error}(\mathcal{S}_\mathcal{Q})$ by induction on i, where X_i is the i-th approximation of $\mathsf{error}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$. The case of $i = 0$ is trivial, since $X_0 = \emptyset$. For the difficult case of $t' \in X_{k+1}$, either: (i) $t' \in \mathsf{violations}(\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$; or (ii) there exists $t'' \in (\mathcal{A}^I_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}})^*$ such that $t't'' \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ and $\forall o \in \mathcal{A}^O_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cdot t't''o \in X_k$. For (i), there is a prefix and input extension t'' of t' such that there exists $t_w \in \mathcal{R}_{\mathcal{S}_\mathcal{W}}$ with $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = t''$, $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$, and either $t_w \in \mathsf{error}(\mathcal{S}_\mathcal{W})$ or $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}}$. If $t_w \in \mathsf{error}(\mathcal{S}_\mathcal{W})$, then $t_w \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, since $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$. By Lemma 6, it follows that $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. Alternatively, if $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}}$, then if $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \notin \mathsf{error}(\mathcal{S}_\mathcal{Q})$ it follows that $t_w \notin \mathcal{R}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$. Since $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$, it must hold that $t_w \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, which again by Lemma 6 implies $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. Note that $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = t''$, so $t \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. For (ii), by the induction hypothesis we know that $\forall o' \in \mathcal{A}^O_{\mathcal{S}_\mathcal{Q}} \cdot t't''o' \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. To show that $t't'' \in \mathcal{L}_{\mathcal{S}_\mathcal{Q}}$, note from $t't'' \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}}$ that there exists $t_w \in \mathcal{L}_{\mathcal{S}_\mathcal{W}}$ with $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = t't''$ such that $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{L}_{\mathcal{S}_\mathcal{P}}$ and $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$. Since $\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q} \sqsubseteq \mathcal{S}_\mathcal{W}$, it follows that $t_w \in \mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$ or $t_w \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$. For the former, it follows that $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathcal{L}_{\mathcal{S}_\mathcal{Q}}$, while for the latter $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$ (Lemma 6). Either way, since $t_w \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = t't''$, it follows that $t't'' \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$, which in turn yields $t' \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$.

Finally, suppose that $t \in \mathcal{L}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} \cap \mathcal{A}^*_{\mathcal{S}_\mathcal{Q}}$. Then there exists $t' \in \mathcal{A}^*_{\mathcal{S}_\mathcal{W}}$ with $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}} = t$ such that $t' \in \mathcal{L}_{\mathcal{S}_\mathcal{W}}$, $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathcal{L}_{\mathcal{S}_\mathcal{P}}$ and $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{P}} \notin \mathsf{error}(\mathcal{S}_\mathcal{P})$. From $t' \in \mathcal{L}_{\mathcal{S}_\mathcal{W}}$ we derive $t' \in \mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}} \cup \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$. If $t' \in \mathcal{L}_{\mathcal{S}_\mathcal{P}||\mathcal{S}_\mathcal{Q}}$, then certainly $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathcal{L}_{\mathcal{S}_\mathcal{Q}}$. If instead $t' \in \mathsf{error}(\mathcal{S}_\mathcal{P} \mid\mid \mathcal{S}_\mathcal{Q})$, then by Lemma 6 $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} \in \mathsf{error}(\mathcal{S}_\mathcal{Q})$. It is easy to see that $t' \upharpoonright \mathcal{A}_{\mathcal{S}_\mathcal{Q}} = t$.

Proof of Theorem 8

For soundness, first note that $\mathcal{I}(\mathcal{S}_\mathcal{P}) \models \mathcal{S}_\mathcal{P}$, and so $\mathcal{I}(\mathcal{S}_\mathcal{P}) \mid\mid \mathcal{Q} \models \mathcal{S}_\mathcal{W}$. Consequently, $\mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P}) \mid\mid \mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$, and from the proof of Theorem 2 we know that $\mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P})) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$. Moreover, $\mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P})) \sqsubseteq \mathcal{S}_\mathcal{P}$, so by Theorem 7 it follows that $\mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ as required.

For completeness, by the interfaces of $\mathcal{P}$ and $\mathcal{S}_\mathcal{P}$, as well as $\mathcal{Q}$ and $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$, matching, it follows that if $\mathcal{AG}(\mathcal{P}) \sqsubseteq \mathcal{S}_\mathcal{P}$, then $\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{P} \mid\mid (\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P})$, since the conditions for monotonicity in Theorem 1 are satisfied. Now by transitivity (the conditions being trivially satisfied) and Theorem 7, we obtain $\mathcal{AG}(\mathcal{P}) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$. Hence $\mathcal{AG}(\mathcal{P} \mid\mid \mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$ by Lemma 7.

Proof of Corollary 1

For soundness, note that $\mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P}) \mid\mid \mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$, which by Lemma 7 yields $\mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P})) \mid\mid \mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}$. As $\mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P})) \sqsubseteq \mathcal{S}_\mathcal{P} \sqsubseteq \mathcal{AG}(\mathcal{I}(\mathcal{S}_\mathcal{P}))$, it follows by Theorem 7 that $\mathcal{AG}(\mathcal{Q}) \sqsubseteq \mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ given $\mathcal{AG}(\mathcal{Q})$ and $\mathcal{S}_\mathcal{W}/\mathcal{S}_\mathcal{P}$ have identical interfaces. Completeness follows by Theorem 8.

Appendix F. Decomposing Parallel Composition

Proof of Corollary 2

Follows immediately from Theorems 2 and 7.