Contract-Based Reasoning for Component Systems with Complex Interactions

Sophie Quinton, Susanne Graf and Roberto Passerone

Verimag Research Report n^o TR-2010-12

May 2010

Contract-Based Reasoning for Component Systems with Complex Interactions

Sophie Quinton, Susanne Graf and Roberto Passerone

May 2010

Abstract

This paper analyzes various notions of refinement used in contract and interface theories. We define a generic component algebra and then focus on possible verification strategies and discuss their compositional properties. We study the relationship between refinement of open systems and refinement under context, and show that we can obtain stronger compositional results, known as circular reasoning, if we loosen their connection. The theoretical results are then presented in the context of two models derived from the SPEEDS HRC specification language, and we show how to combine partial tool chains for both frameworks into a complete tool chain for our methodology.

Keywords: refinement, interface theories, circular reasoning

Reviewers: Susanne Graf

Notes:

How to cite this report:

```
@techreport { ,
title = { Contract-Based Reasoning for Component Systems with Complex Interactions},
author = { Sophie Quinton, Susanne Graf and Roberto Passerone},
institution = {  Verimag Research Report },
number = {TR-2010-12},
year = { },
note = { }
}
```

1 Introduction

Contract and interface frameworks are emerging as the formalism of choice for system designs that require large and dispersed teams, or where the supply chain is complex [3]. Contracts are usually expressed as pairs of *assumptions*, or properties that the environment must satisfy, and *guarantees*, the properties that must be satisfied by each particular component. This style of specification is typically employed in the context of component-based frameworks, which facilitate the design process by hierarchically decomposing the design. This process is particularly effective if one can deduce properties of the entire system from properties of its parts. In this case the framework is said to support *compositional reasoning*.

Of interest in system design is the relationship of refinement between a specification and an implementation. In contract frameworks, refinement takes different forms, depending on whether we are considering refinement with respect to a specification, refinement of contracts, or refinement of an implementation with respect to a contract. In this paper we present a methodology that makes use of these three forms of refinement to simplify the verification process in system design. In particular, we consider two specific notions: refinement between closed systems, and refinement under context, which is in turn used to define the relationship of satisfaction of a contract by a component, and refinement between contracts, which we call dominance. These relationships are introduced using a generic contract framework that uses abstract composition operators to encompass a variety of different interaction models.

We discuss some rules for compositional reasoning and the constraints they impose on the refinement relations. In particular, circular reasoning, which allows a component and its environment to be refined concurrently — each relying on the abstract description of its context — entails an interesting rule for proving dominance. But it imposes a relatively strong requirement on the refinement relation. We show how to relax this constraint when checking refinement between contracts. In addition, we impose a much looser relationship between refinement and refinement under context than usually assumed, which allows us in some cases to derive more powerful rules for reasoning about contracts.

This work has a practical motivation in the component-framework HRC — standing for heterogeneous rich components — defined in the SPEEDS IP project and used in the COMBEST project. The model defines component properties as extended transition systems and provides several composition models, ranging from low-level semantic composition to composition frameworks underlying the design tools used by system designers. We focus here on two composition – and corresponding contract — frameworks. We show that both can be seen as instances of our general contract framework and, based on the relationships between the underlying refinement relations and the properties they enjoy, we show how to combine partial tool chains for both frameworks into a complete tool chain for our methodology.

The proofs of all theorems are given in the appendix.

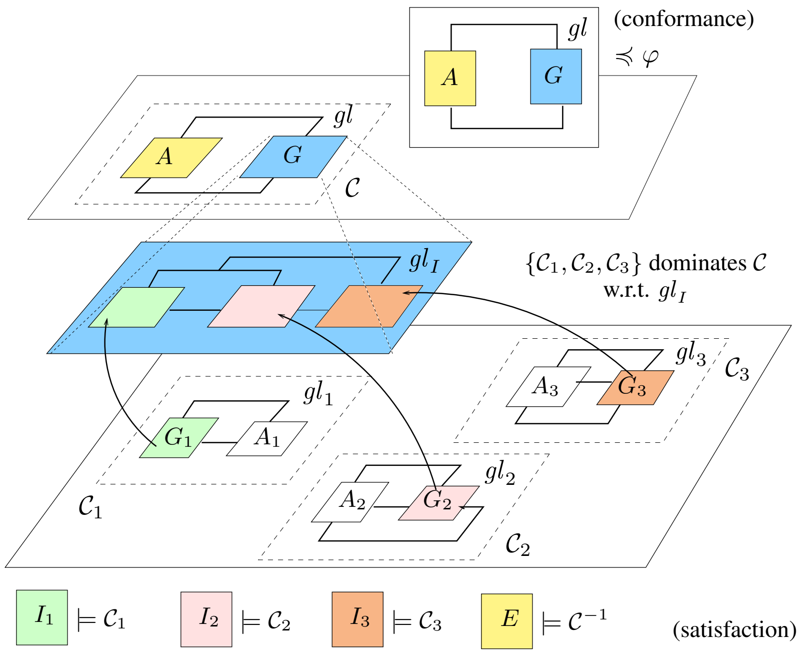
2 Contract-based design

2.1 Design methodology

From a macroscopic point of view, we adopt a top-down design and verification methodology in which high-level properties are pushed progressively from the level of the system to the level of atomic components — which we call implementations. As usual, this is just a convenient representation; in real life, we always achieve the final picture in several iterations alternatively going up and down the hierarchy.

Our methodology, illustrated in Figure 1, makes use of the notion of *contract* [2, 12]. More details are provided in [3]. For a component K, a contract describes 1) the interface of K 2) the interaction between K and its environment denoted E, and 3) an abstraction of the expected behavior of E and an abstraction of the promised behavior of K. The idea is that the system component is refined into a set of subcomponents assembled using a complex interaction layer. Contracts are associated to each of the subcomponents in such a way that if we can build implementations satisfying the contracts of the subcomponents, then their composition satisfies the system contract.

More precisely, we suppose given a global property φ which the system K under construction has to realize together with an environment on which we may have some knowledge, expressed by a property



gl
(conformance)
$\preccurlyeq \varphi$
A
G
gl
A
G
$\mathcal{C}$
gl_I
$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ dominates $\mathcal{C}$ w.r.t. gl_I
gl_3
$\mathcal{C}_3$
A_3
G_3
gl_1
G_1
A_1
$\mathcal{C}_1$
gl_2
A_2
G_2
$\mathcal{C}_2$
I_1 $\models \mathcal{C}_1$
I_2 $\models \mathcal{C}_2$
I_3 $\models \mathcal{C}_3$
E $\models \mathcal{C}^{-1}$
(satisfaction)

Figure 1: Method for proving $gl\{E, gl_I\{I_1, I_2, I_3\}\} \preccurlyeq \varphi$

A. The property φ and A are expressed w.r.t. the interface $\mathcal{P}_K$ of K. We proceed as follows: (1) define a *contract* $\mathcal{C}$ for $\mathcal{P}_K$ which *conforms* to φ; (2) define K as a composition of subcomponents K_i and a contract $\mathcal{C}_i$ for each of them; possibly iterate this step if needed; (3) prove that any set of implementations I_i for K_i *satisfying* the contracts $\mathcal{C}_i$, when composed, satisfy the top-level contract $\mathcal{C}$ (*dominance*) — and thus guarantee φ; (4) provide such implementations.

The global property φ appears at the top, while the implementations I_i are at the bottom. The correctness proof for a particular system is therefore split into 3 phases: *conformance* (denoted $\preccurlyeq$) of the top-level contract $\mathcal{C}$ to φ, *dominance* between the contracts $\mathcal{C}_i$ and $\mathcal{C}$, and *satisfaction* (denoted $\models$) of the $\mathcal{C}_i$ by the implementation I_i. Thus, *conformance* relates properties of closed systems — as a contract defines a closed system made of the composition of its assumption and guarantee — while *dominance* relates contracts and *satisfaction* relates components to contracts.

2.2 Contract framework

We develop our methodology on a generic framework that supports hierarchical components and mechanisms to reason about composition. The following notions and properties form the basis of this framework. Here, we use glue operators [14] to generalize the operation of parallel composition found in most traditional frameworks.

Definition 2.1 (Component algebra) A component algebra *is a structure of the form* $(\mathcal{K}, GL, \circ, \cong)$ *where:*

- $\mathcal{K}$ *is a set of* components. *Each component* $K \in \mathcal{K}$ *has as its* interface *a set of* ports, *denoted* $\mathcal{P}_K$.
- GL *is a set of* glue (composition) operators. *A glue is a partial function* $2^{\mathcal{K}} \longrightarrow \mathcal{K}$ *transforming a set of components into a new component. Each* $gl \in GL$ *is defined on a set of ports* S_{gl} *from the original set of components — called its* support set *— and defines a new interface* $\mathcal{P}_{gl}$ *for the new component — called its* exported interface. $K = gl(\{K_1, \ldots, K_n\})$ *is defined if* $K_1, \ldots, K_n \in \mathcal{K}$ *have disjoint interfaces,* $S_{gl} = \bigcup_{i=1}^{n} \mathcal{P}_{K_i}$ *and the interface of* K *is* $\mathcal{P}_{gl}$*, the exported interface of* gl.
- $\cong \subseteq \mathcal{K} \times \mathcal{K}$ *is an equivalence relation. In general, this equivalence is derived from equality or equivalence of semantic sets.*

- *$\circ$ is a partial operation on GL to hierarchically compose glues. $gl \circ gl'$ is defined if $\mathcal{P}_{gl'} \subseteq S_{gl}$. Then, its support set is $S_{gl} \backslash \mathcal{P}_{gl'} \cup S_{gl'}$ and its interface is $\mathcal{P}_{gl}$ (cf. Figure 2).*
  Furthermore, $\circ$ must be coherent with $\cong$ in the sense that $gl\{gl'\{\mathcal{K}_1\}, \mathcal{K}_2\} \cong (gl \circ gl')\{\mathcal{K}_1 \cup \mathcal{K}_2\}$ for any sets of components $\mathcal{K}_i$ such that all terms are defined.

To simplify the notation, we write $gl\{K_1, \ldots, K_n\}$ instead of $gl(\{K_1, \ldots, K_n\})$. Figure 2 shows how hierarchical components and connectors are built from atomic ones. Note that exported ports of internal connectors (which are not connected) are not represented in this figure.

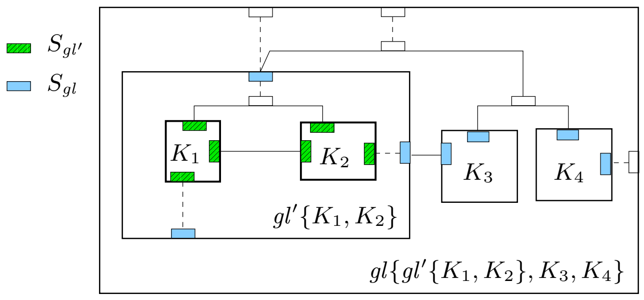


Figure 2: Hierarchical components and connectors

We use the notion of *context* to restrict the way a component may be further composed.

Definition 2.2 (Context) *A* context *for an interface $\mathcal{P}$ is a pair (E, gl) where E is such that $\mathcal{P} \cap \mathcal{P}_E = \emptyset$ and gl is defined on $\mathcal{P} \cup \mathcal{P}_E$.*

We introduce two refinement relations to reason about contracts: *conformance*, which we informally introduced when discussing our methodology; and *refinement under context*, used to define *satisfaction* and *dominance*. Refinement under context is usually considered as a derived relation and chosen as the weakest relation implying conformance and ensuring *compositionality*, i.e., preservation by composition. We loosen the coupling between these two refinements to obtain stronger reasoning schemata for dominance.

Definition 2.3 (Contract framework) *A* contract framework *is a tuple $(\mathcal{K}, GL, \circ, \cong, \{\sqsubseteq_{E,gl}\}, \preccurlyeq)$ where:*

- *$(\mathcal{K}, GL, \circ, \cong)$ is a component algebra.*
- *$\{\sqsubseteq_{E,gl}\}$ is a* refinement under context *relation parameterized by a context. Given a context (E, gl) for an interface $\mathcal{P}$, $\sqsubseteq_{E,gl}$ is a preorder over the set of components on $\mathcal{P}$.*
- *$\preccurlyeq \subseteq \mathcal{K} \times \mathcal{K}$ is a* conformance *relation between components with the same interface. It is a preorder such that for any K_1, K_2 on the same interface $\mathcal{P}$ and for any context (E, gl) for $\mathcal{P}$, $K_1 \sqsubseteq_{E,gl} K_2 \implies gl\{K_1, E\} \preccurlyeq gl\{K_2, E\}$.*

Example 2.4 *Typical notions of conformance $\preccurlyeq$ are* trace inclusion *or its structural counter part,* simulation[1]*. For these notions of conformance, refinement under context is usually defined as:*

$$K_1 \sqsubseteq^{\preccurlyeq}_{E,gl} K_2 \triangleq gl\{K_1, E\} \preccurlyeq gl\{K_2, E\}$$

That is, $\sqsubseteq^{\preccurlyeq}$ is defined as the weakest preorder included in $\preccurlyeq$ that is compositional.

Definition 2.5 (Contract) *A* contract *$\mathcal{C}$ for an interface $\mathcal{P}$ consists of:*

- *a context $\mathcal{E} = (A, gl)$ for $\mathcal{P}$; A is called the* assumption
- *a component G on $\mathcal{P}$ called the* guarantee

[1]Simulation is formally defined in section 3.2.

We write $\mathcal{C} = (A, gl, G)$ rather than $\mathcal{C} = ((A, gl), G)$. The interface of the environment is implicitly defined by gl while A expresses a constraint on it and G a constraint on the refinements of K. The "mirror" contract $\mathcal{C}^{-1}$ of $\mathcal{C}$ is (G, gl, A), i.e. a contract for the environment.

Definition 2.6 (Satisfaction of contract) *A component K* satisfies *a contract $\mathcal{C} = (A, gl, G)$, denoted $K \models \mathcal{C}$, if and only if $K \sqsubseteq_{A,gl} G$.*

In interface theories [4], a single constraint is used to represent both A and G (gl is predefined), because each transition is controlled either by the component or the environment. However, in frameworks with rendez-vous interaction, several pairs (A, G) may be derived from an interface, as both component and environment may prevent a rendez-vous from taking place. Thus, we keep assumptions and guarantees separate.

Our notion of contract has a structural part, which makes this definition very general by encompassing any composition framework. A more practical advantage is related to system design: it allows us to separate the architecture and the properties (requirements) of the system under construction, which evolve independently during the development process. In particular, in frameworks where interaction is rich, refinement can be ensured by relying heavily on the structure of the system and less importantly on the behavioral properties of the environment.

2.3 Reasoning within a contract framework

Compositional reasoning is usually based on the following rule: if $I \preccurlyeq S$, then $I \parallel E \preccurlyeq S \parallel E$ (where $\parallel$ denotes generically parallel composition). Thus, from $I_1 \preccurlyeq S_1$ and $I_2 \preccurlyeq S_2$, we get $I_1 \parallel I_2 \preccurlyeq S_1 \parallel S_2$. The proof is as follows:

1. $I_1 \preccurlyeq S_1 \implies I_1 \parallel I_2 \preccurlyeq S_1 \parallel I_2$
2. $I_2 \preccurlyeq S_2 \implies I_2 \parallel S_1 \preccurlyeq S_2 \parallel S_1$

If we consider refinement under context, and not simply refinement in an open system, reasoning is a bit more complex because we have to handle closed systems. In particular, the previous rule is not sufficient anymore. Remember that for a contract framework we require that $I \sqsubseteq_E S$ imply $I \parallel E \preccurlyeq S \parallel E$. Now, if $I_1 \sqsubseteq_{I_2} S_1$ and $I_2 \sqsubseteq_{I_1} S_2$, this is not sufficient to derive $I_1 \parallel I_2 \preccurlyeq S_1 \parallel S_2$. In fact, we only have:

- $I_1 \sqsubseteq_{I_2} S_1 \implies I_1 \parallel I_2 \preccurlyeq S_1 \parallel I_2$
- $I_2 \sqsubseteq_{I_1} S_2 \implies I_2 \parallel I_1 \preccurlyeq S_2 \parallel I_1$

This means that to obtain a rule similar to the usual compositional reasoning, we need the following:

$$\begin{cases} I_1 \sqsubseteq_{I_2} S_2 \\ I_2 \sqsubseteq_{I_1} S_1 \end{cases} \implies I_1 \parallel I_2 \preccurlyeq S_1 \parallel S_2$$

Furthermore, we want to be able to apply incremental design by incorporating parts of the environment into the system. Hence the more general rule:

$$\begin{cases} I_1 \sqsubseteq_{I_2 \parallel E} I_2 \\ I_2 \sqsubseteq_{I_1 \parallel E} I_1 \end{cases} \implies I_1 \parallel I_2 \sqsubseteq_E S_1 \parallel S_2$$

Thus, we require refinement under context to be a relation parameterized by a context such that:

$$\begin{cases} I_1 \sqsubseteq_{gl_{E_2}\{I_2,E\},gl_1} S_1 \\ I_2 \sqsubseteq_{gl_{E_1}\{I_1,E\},gl_2} S_2 \end{cases} \implies gl\{I_1, I_2\} \sqsubseteq_{E,gl_E} gl\{S_1, S_2\}$$

Note that we never use this rule directly, because we do not want to prove refinement in the actual context, but in an abstract one that is refined by the concrete context. Thus, we will only need the property defined below, which we refer to as *compositionality*.

Definition 2.7 (Compositionality) *A refinement under context $\{\sqsubseteq_{E,gl}\}$ is preserved by composition iff for any context (E, gl) for an interface $\mathcal{P}$ and gl_E, E_1, E_2 such that $E = gl_E\{E_1, E_2\}$, the following holds for any I, S on $\mathcal{P}$:*

$$I \sqsubseteq_{E,gl} S \implies gl_1\{I, E_1\} \sqsubseteq_{E_2,gl_2} gl_1\{S, E_1\}$$

where gl_1 and gl_2 are such that $gl \circ gl_E = gl_2 \circ gl_1$.

A refinement under context as defined above has this property. This comes directly from the following:

$$\begin{cases} I \sqsubseteq_{gl_E\{E_1,E_2\},gl} S \\ E_1 \sqsubseteq_{gl_1\{I,E_2\},gl_2} E_1 \end{cases} \implies gl_1\{I, E_1\} \sqsubseteq_{E_2,gl_2} gl_1\{S, E_1\}$$

Now, to check refinement in an abstraction of the actual context, we need another property of refinement. Most frameworks offer the following rule:

$$I_1 \sqsubseteq_{S_2,gl} S_1 \wedge I_2 \sqsubseteq S_2 \implies I_1 \sqsubseteq_{I_2,gl} S_1$$

Here, $I_2 \sqsubseteq S_2$ denotes that I_2 refines S_2 in any context. This is quite limited because this requires to find a way of "breaking the symmetry" of the dependency between component and environment. This is why the following rule, which implies the previous one and is commonly referred to as circular reasoning, is more interesting. Unfortunately, it is not sound in general.

$$I_1 \sqsubseteq_{S_2,gl} S_1 \wedge I_2 \sqsubseteq_{S_1,gl} S_2 \implies I_1 \sqsubseteq_{I_2,gl} S_1$$

For each particular framework, this property can be proved by an induction based on the semantics of composition and refinement. For example, in a contract framework based on I/O automata, circular reasoning is sound because exactly one component has control over each interaction [13].

However, circular reasoning does not hold for parallel composition with synchronizations (as they exist in e.g. in Petri Nets or process algebras) or instantaneous mutual dependences between inputs and outputs (as they exist in synchronous formalisms). Two reasons for the non validity of circular reasoning for $\sqsubseteq^{\preccurlyeq}$ are illustrated in Figure 3.

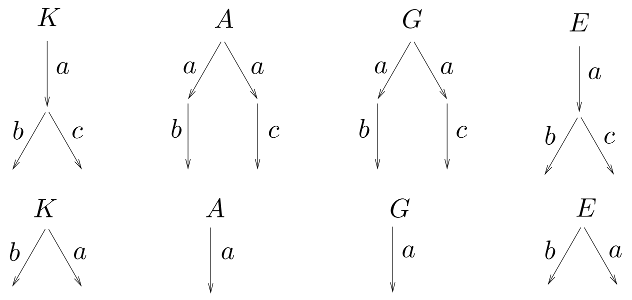


Figure 3: $K \sqsubseteq^{\preccurlyeq}_{A,gl} G$ and $E \sqsubseteq^{\preccurlyeq}_{G,gl} A$ but $K \not\sqsubseteq^{\preccurlyeq}_{E,gl} G$.

Suppose that composition is defined as the synchronization between actions with the same name and interleaving of others. The upper example shows that non-determinism of the abstract environment is a problem. In the example of the second row, both the assumption A and the guarantee G forbid b to occur. This allows their respective refinements according to $\sqsubseteq^{\preccurlyeq}$, E and K, to offer b — since they can rely on G resp. A to forbid its actual occurrence. But obviously, the composition of the implementations $gl\{E, K\}$ now allows b.

Because circular reasoning is not sound for all refinements under context, it may be useful to use a more restrictive definition of refinement under context in order to make circular reasoning sound. This is the reason why refinement under context and conformance are decoupled in our definition of a contract framework.

2.4 Dominance

Dominance is the key notion that distinguishes reasoning in a contract framework from theories based on refinement between components. Intuitively, dominance means for contracts what refinement means for components. More specifically, proving that a contract $\mathcal{C}$ dominates $\mathcal{C}'$ means showing that every implementation of $\mathcal{C}$, that is, every component satisfying $\mathcal{C}$, is also an implementation of $\mathcal{C}'$.

Definition 2.8 (Binary dominance) *Let $\mathcal{C}$ and $\mathcal{C}'$ be two contracts for the same interface $\mathcal{P}$, with $\mathcal{C} = (A, gl, G)$ and $\mathcal{C}' = (A', gl', G')$ (implying that $\mathcal{P}_G = \mathcal{P}_{G'}$). $\mathcal{C}$* dominates *$\mathcal{C}'$ iff $gl = gl'$ (and as a consequence $\mathcal{P}_A = \mathcal{P}_{A'}$) and:*

$$\text{for any } K \text{ on } \mathcal{P}, K \models \mathcal{C} \implies K \models \mathcal{C}'$$

Moreover, one may need in some cases a notion of *well-formedness* of a system. It is generally a property that cannot be expressed in the considered component framework: deadlock-freedom, termination, etc. One expects however such a property to be preserved top-down by conformance.

Definition 2.9 (well-formedness) *We define a* well-formedness *constraint as a property preserved by conformance, that is if S is well-formed and $S' \preccurlyeq S$, then S' is well-formed.*

As typically a top-level description is not fine-grained enough to guarantee well-formedness, we are also interested in preserving well-formedness in a bottom-up fashion — that is from implementation contracts to global contracts. This may then be achieved by strengthening the definition of dominance, for example by adding the following constraint:

$\star$ for any E on $\mathcal{P}_A$, $E \models (G', gl, A') \implies E \models (G, gl, A)$

Contract algebras. Some contract frameworks define an algebra of contracts. Examples are [1] and the SPEEDS L0 framework presented in the next section. Having an algebra of contracts is in principle attractive. Our dominance problem would then boil down to checking that $\widetilde{gl}(\mathcal{C}_1, \ldots, \mathcal{C}_n)$ dominates $\mathcal{C}$ for a given $\widetilde{gl}$ defined on contracts.

We do not propose such an algebra for two reasons. The first one is that this would impose much stronger constraints on the general notion of contract framework: we would need for each composition operator gl the existence of an operator $\widetilde{gl}$ on contracts such that $K_1 \models \mathcal{C}_1 \wedge K_2 \models \mathcal{C}_2 \iff gl\{K_1, K_2\} \models \widetilde{gl}(\mathcal{C}_1, C_2)$. This is clearly not possible for arbitrary component frameworks. In fact, it is not even possible to define greatest lower bounds or least upper bounds of contracts for arbitrary settings. Also, there is in general no largest implementation satisfying a contract. Our aim is to propose a generic notion of contract framework which imposes minimal constraints on the component framework it is associated with.

The second reason for not proposing an algebra of contracts is that we show in the following that it is possible to reason efficiently about contracts without ever composing them.

If one cannot compose contracts or wants to avoid it, a dominance check involves in general not just a pair of contracts. A typical situation would be the one depicted on Figure 1, where a set of contracts $\{\mathcal{C}_i\}_{i=1}^n$ are attached to disjoint interfaces $\{P_i\}_{i=1}^n$. Besides, a composition operator gl_I is defined on $P = \bigcup_{i=1}^n P_i$ and a contract $\mathcal{C}$ is given for $P' \subseteq P$.

We thus provide a broader notion of dominance: a set of contracts $\{\mathcal{C}_i\}_{i=1..n}$ dominates a contract $\mathcal{C}$ w.r.t. a composition operator gl_I iff any set of components satisfying the contracts $\mathcal{C}_i$, when composed using gl_I, makes a component satisfying $\mathcal{C}$.

In order to hide ports of the lower-level contracts which do not appear at the interface of the top-level contract, we relax the constraints on the composition operators by only requiring that they agree on their common ports. For this, we need a notion of *projection* of a component K onto a subset P' of its interface, denoted $\Pi_{P'}(K)$, which is quite natural and must preserve the following properties.

Definition 2.10 (Projection) *If Π is a* projection*, then for any components K_i and A on disjoint interfaces $\mathcal{P}$ and $\mathcal{P}_A$, and any composition operator gl on $\mathcal{P} \cup \mathcal{P}_A$:*

1. *for $P' \subseteq \mathcal{P}$ and for any gl_1, gl_2 with $P_{gl_1} = P' \cup \mathcal{P}_A$, $P_{gl_2} = \mathcal{P} \backslash P'$ s.t. $gl = gl_1 \circ gl_2$:*

$$K_1 \sqsubseteq_{A,gl} K_2 \wedge \Pi_{P'}(K_2) \sqsubseteq_{A,gl_1} G \implies \Pi_{P'}(K_1) \sqsubseteq_{A,gl_1} G$$

   Note that G is defined on P'.

2. *for $P'_A \subseteq \mathcal{P}_A$ and for any gl_1, gl_2 on $P_{gl_1} = \mathcal{P} \cup P'_A$, $P_{gl_2} = \mathcal{P}_A \backslash P'_A$ s.t. $gl = gl_1 \circ gl_2$:*

$$K \sqsubseteq_{\Pi_{P'_A}(A),gl_1} G \implies K \sqsubseteq_{A,gl} G$$

   Note that G is defined on P.

These properties state that ports of the component (and symmetrically of the environment) which do not appear in interactions with the environment (resp. the component) may be abstracted away when checking refinement.

Note that we might use an equivalence relation between composition operators rather than equality. This would be, e.g., in the context of L1 connectors, an equivalence based on equality of the associated sets of interactions. More generally, this can be any relation such that $gl \cong gl'$ implies that $gl\{K_1, \dots, K_n\} \cong gl'\{K_1, \dots, K_n\}$ for any set of components such that compositions are defined.

We now state formally our semantic definition of dominance.

Definition 2.11 (Dominance for a composition) *Let $\mathcal{C}$ be a contract on P, $\{\mathcal{C}_i\}_{i=1}^n$ a set of contracts on $\mathcal{P}_i$ and gl_I a composition operator such that $S_{gl_I} = \bigcup_{i=1}^n \mathcal{P}_i$ and $P \subseteq \mathcal{P}_{gl_I}$. Then $\{\mathcal{C}_i\}_{i=1}^n$* dominates $\mathcal{C}$ with respect to gl_I *iff for any components $\{K_i\}_{i=1}^n$:*

$$(\forall i, K_i \models \mathcal{C}_i) \implies \Pi_P(gl_I\{K_1, \dots, K_n\}) \models \mathcal{C}$$

A sufficient condition for dominance has been proposed in [12] when circular reasoning is sound. It relies on the fact that local assumptions are indeed discharged, that is, implied by the environment defined by the guarantees of the peers and the global assumption A. Yet, it requires that $\forall i, gl \circ gl_I = gl_i$. The generalization of this rule to composition operators on different sets of ports is given below.

Theorem 2.12 *If $\forall i.\ \exists gl_{E_i}.\ gl \circ gl_I = gl_i \circ gl_{E_i}$ and circular reasoning is sound, then to prove that $\mathcal{C}$ dominates $\{\mathcal{C}_i\}_{i=1..n}$ w.r.t. gl, it is sufficient to prove that:*

$$\begin{cases} \Pi_P(gl_I\{G_1, \dots, G_n\}) \models \mathcal{C} \\ \forall i, \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A, G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_n\}) \models \mathcal{C}_i^{-1} \end{cases}$$

Note that the well-formedness condition ($\star$) discussed previously is obviously preserved by this rule.

In SPEEDS HRC, discussed below, the required decomposability of $gl \circ gl_I$ into $gl_i \circ gl_{E_i}$ is trivial for L0 but not for L1. Due to lack of space, we do not develop this issue here.

This condition shows that the proof of a dominance relation boils down to a set of satisfaction checks, one for proving refinement between the guarantees, the second for discharging individual assumptions.

This result is particularly significant because one can check dominance while avoiding composition of contracts, which is, as stated before, impossible in the general case and leads to state explosion in most concrete contract frameworks.

2.5 Combining two refinement relations

We consider here using two relations of refinement under context $\sqsubseteq^\alpha$ and $\sqsubseteq^\beta$ instead of a single one. Mainly, the goal is to reintroduce some asymmetry between the property that must be guaranteed by the component and the one to be guaranteed by its environment, because full symmetry might be either not sufficient or unnecessarily strong to ensure circular reasoning.

In particular, we consider the following rule which we call pseudo circular reasoning for $\sqsubseteq^\alpha$ and $\sqsubseteq^\beta$:

$$K \sqsubseteq^\alpha_{A,gl} G \wedge E \sqsubseteq^\beta_{G,gl} A \implies K \sqsubseteq^\alpha_{E,gl} G \quad (0)$$

It guarantees that $\sqsubseteq^\alpha$ is preserved in a circular-like fashion if the environment refines A via some relation $\sqsubseteq^\beta$.

This kind of pseudo circular reasoning can be used in different ways to make dominance checks easier. We will use both of them in our application to the SPEEDS framework in the next section where we have two refinements under context, one supporting circular reasoning, and one not.

Relaxing standard non circular reasoning. Suppose that $\sqsubseteq^\alpha$ does not allow circular reasoning. Suppose that we have a second refinement under context relation $\sqsubseteq^\beta$ which is strong enough to guarantee pseudo circular reasoning.

In this case, we can use (0) as a rule for verifying dominance instead of rule (1) which was proposed for the case where circular reasoning is not possible.

$$K \sqsubseteq^\alpha_{A,gl} G \wedge E \sqsubseteq A \implies K \sqsubseteq^\alpha_{E,gl} G \quad (1)$$

Rule (0) relaxes rule (1) by requiring assumptions to be refined via $\sqsubseteq^\beta$, which is generally weaker than refinement in any context. Thus, condition (0) allows us to establish dominance more often than rule (1).

Relaxing circular reasoning. Suppose now that circular reasoning is sound for $\sqsubseteq^\alpha$. Suppose again that we have another, weaker, refinement under context relation $\sqsubseteq^\beta$.

In this case, rule (0) relaxes the rule for circular reasoning

$$K \sqsubseteq^\alpha_{A,gl} G \wedge E \sqsubseteq^\alpha_{G,gl} A \implies K \sqsubseteq^\alpha_{E,gl} G \quad (2)$$

We can prove that pseudo circular reasoning allows the following sufficient condition for dominance when $\sqsubseteq^\alpha$ is stronger than $\sqsubseteq^\beta$.

Theorem 2.13 *If $\forall i.\ \exists gl_{E_i}.\ gl \circ gl_I = gl_i \circ gl_{E_i}$ and pseudo circular reasoning is sound for $\sqsubseteq^\alpha$ and $\sqsubseteq^\beta$, then to prove that $\mathcal{C}$ dominates $\{\mathcal{C}_i\}_{i=1..n}$ w.r.t. gl, it is sufficient to prove:*

$$\begin{cases} \Pi_P(gl_I\{G_1,\ldots,G_n\}) \models^\alpha \mathcal{C} \\ \forall i, \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A, G_1,\ldots,G_{i-1},G_{i+1},\ldots,G_n\}) \models^\beta \mathcal{C}_i^{-1} \end{cases}$$

Now suppose that $\sqsubseteq^\beta$ is not only weaker but also less costly to check than $\sqsubseteq^\alpha$. Then pseudo-circular reasoning allows establishing dominance more often, and in addition it increases the scalability of dominance checks.

If there is a well-formedness condition, then pseudo circular reasoning can be adopted only if both relations $\sqsubseteq^\alpha$ and $\sqsubseteq^\beta$ preserve this well-formedness condition.

3 Two contract frameworks

We show now how the general theory introduced in the previous section is applied in practice. In the SPEEDS project, we have defined a modeling framework HRC — standing for Heterogeneous Rich Components — to offer designers an environment for contract-based design. HRC offers the possibility to define hierarchical components where interactions and data exchange are defined by explicit connectors between ports which define the component's interface. Components are associated with behaviors (implementations) and contracts are represented by a pair of behaviors (that is, properties) expressing an assumption on the environment and a guarantee, that is a property that the component's implementation must realize in any environment obeying the constraint imposed by the assumption.

In HRC, we choose to represent behaviors on an interface (that is represented by a set of ports $\mathcal{P}$) by labeled transition systems, which are sufficiently enriched to represent hybrid and stochastic behaviors as well as implementations realizing complex data transformations. To simplify the presentation, we restrict ourselves here to abstract transition systems without data nor any other extension.

Definition 3.1 (LTS as behaviors) *A labeled transition system (LTS) on a set of ports $\mathcal{P}$ is defined as a tuple $(Q, q^0, \mathcal{P}, \longrightarrow)$ where Q is a set of states, q^0 an initial initial state and $\longrightarrow\ \subseteq Q \times 2^{\mathcal{P}} \times Q$ is a transition relation.*

For hierarchical model composition, there exist several models defining different levels of abstraction: they range from low-level semantic composition (called L0) to user level composition which generalizes the composition primitives existing in commercial design tools. We discuss here two layers: L0 which we already mentioned, and L1 which offers a richer set of composition operators.

In L0, connectors are simple name matchings which connect an output port to one (or more) input ports, possibly in a hierarchical fashion. The composition semantics is synchronous, that is, a transition of a composed system involves all components in a globally maximal interaction. L1-connectors are more complex; they are inspired by BIP connectors [6]. In L1, a connector defines a set of possible interactions on a set of ports, which may be output, input or event ports. For compatibility reasons, connectors on inputs and outputs must obey the same constraints as L0 connectors, that is, connect exactly one output to a set of inputs and define exactly one interaction involving all connected ports. The L1 execution model is more asynchronous as executions of different connectors are interleaved, and maximal progress is only enforced for individual connectors.

3.1 L0 contract framework

At the semantic level, the L0 contract framework of the SPEEDS HRC model is based on a simple trace-based representation, and uses set operations for the definition of the operators. In other words, LTS are used at the L0 level as recognizers for the trace representation. This is convenient, as the synchronous composition semantics translates into simple intersection of trace sets. In this context, we are not concerned with the specific form of a trace (more details can be found elsewhere [2]). Instead, we simply assume that for any set of ports $\mathcal{P}$ there exists a set of corresponding traces G over those ports which we call *behaviors* or *runs* over $\mathcal{P}$. As discussed in Section 2.4, the set of behaviors is equipped with a projection operator $\mathbf{proj}_{\mathcal{P}_1,\mathcal{P}_2}(G)$ which restricts the behaviors to ports in $\mathcal{P}_1 \subseteq \mathcal{P}_2$, and a corresponding inverse projection $\mathbf{proj}^{-1}_{\mathcal{P}_1,\mathcal{P}_2}(G)$ to extend the behaviors to a larger set of ports $\mathcal{P}_2$.

A component K with interface $\mathcal{P}_K$ at level L0 is defined as a set of behaviors over $\mathcal{P}_K$. The behaviors correspond to the history of ports that are visited when traversing the transitions of the LTS. Composition, in the original formulation, is defined as a new LTS obtained by the Cartesian product of the transition systems, and by retaining only the pairs of transitions whose labels of ports match, given the correspondence induced by the connectors. If the matching ports of the two components had the same names, composition at the level of trace sets would boil down to a simple intersection of the sets of behaviors. Because this is not true in general, and is forbidden by the definitions of our framework (components must have disjoint sets of ports under composition), we must introduce the connectors as explicit components that establish a synchronous relation between the histories of connected ports. The collection of these simple connectors forms the composition operators Γ of our framework at the L0 level. In addition to that, to make intersection work, we must also equalize the ports of all trace sets using inverse projection, to have a coherent representation of the composite. In particular, if $\mathcal{K} = \{K_1, \dots, K_n\}$ is a set of components such that $\mathcal{P}_1, \dots, \mathcal{P}_n$ are pairwise disjoint, then a composition operator Γ for $\mathcal{K}$ is a component K_Γ defined on the ports $\mathcal{P} = p_\Gamma \cup (\cup_{i=1}^n \mathcal{P}_i)$, and

$$\begin{aligned} K &= \Gamma\{K_1, \dots, K_n\} \\ &= \mathbf{proj}_{p_\Gamma,\mathcal{P}}\left(K_\Gamma \cap \mathbf{proj}^{-1}_{\mathcal{P}_1,\mathcal{P}}(K_1) \cap ... \cap \mathbf{proj}^{-1}_{\mathcal{P}_n,\mathcal{P}}(K_n)\right) \end{aligned}$$

Component K_Γ is always taken as an identity operator, and is used exclusively to rename ports in the composition and to construct the new interface p_Γ. In the following, at the semantic level, we implicitly assume the appropriate connector components are used whenever a composition is required, and instead use components with equal sets of ports for convenience.

The definition of $\circ$ is straightforward. Since composition operators are themselves components, their composition follows the same principle as component composition. Finally, the $\cong$ relations on $\mathcal{K}$ is taken as equality of sets of traces.

It is easy to define a notion of conformance $\preccurlyeq$ for the L0 model. This notion is equivalent to the traditional notion of refinement, and is defined as trace containment. More formally, if K_1 and K_2 are components over the same set of ports $\mathcal{P}$, then $K_1 \preccurlyeq K_2$ if and only if $K_1 \subseteq K_2$.

L0 contracts. Contracts are defined in L0 as pairs (A, G) of components over the same set of ports. In particular, A represents the *assumptions* of the contract, or, equivalently, the behaviors that are considered acceptable by the contract. Likewise, G expresses the guarantees, or those behaviors that are possible under the contract, provided the assumptions are satisfied. The composition operator Γ is implied by port name matching.

The definition of *refinement* under context $\sqsubseteq_{E,\Gamma}$ is derived from the definition of composition and conformance as in Example 2.4. $K_1 \sqsubseteq_{E,\Gamma} K_2$ if and only if $\Gamma\{K_1, E\} \preccurlyeq \Gamma\{K_2, E\}$. The relation so defined is a preorder, and satisfies by definition the conditions required by our framework.

In the L0 model, contract satisfaction is defined as refinement under the context of the assumptions. Formally, a component K satisfies a contract $\mathcal{C} = (A, G)$ if and only if

$$K \cap A \subseteq G.$$

Observe that the above is equivalent to $K \cap A \subseteq G \cap A$. In our framework, this translates into $\Gamma\{K, A\} \preccurlyeq \Gamma\{G, A\}$, where Γ is the appropriate composition operator that computes the identity relation. By definition, this is the same as $K \sqsubseteq_{A,\Gamma} G$. Thus, the definition of satisfaction in HRC is consistent with the more general definition of satisfaction of our framework.

In the L0 model there exists a unique maximal component satisfying a contract $\mathcal{C}$, namely:

$$M_{\mathcal{C}} = G \cup \neg A, \tag{1}$$

where $\neg$ denotes the operation of complementation on the set of all behaviors over ports $\mathcal{P}_A$. The operation of computing a canonical form is well defined, since the maximal implementation is unique, and it is idempotent. It is easy to show that $K \models \mathcal{C}$ if and only if $K \subseteq M_{\mathcal{C}}$. We say that a contract $\mathcal{C} = (A, G)$ is in *canonical form* when $G = M_{\mathcal{C}}$. Every contract has an equivalent contract in canonical form, which is obtained by replacing G with $M_{\mathcal{C}}$. In the following, we focus on contracts in canonical form, since several expressions can be simplified. The limitation is that complementation may not be effective in certain models (such as timed models). In those cases, the use of canonical forms is precluded, and the more generic L1 theory is required.

Parallel composition of contracts in L0. Contract composition formalizes how contracts related to different components should be combined to specify a single, compound, component. Let $\mathcal{C}_1 = (A_1, G_1)$ and $\mathcal{C}_2 = (A_2, G_2)$ be contracts. First, composing these two contracts amounts to composing their promises. Regarding assumptions, however, the situation is more subtle. Suppose first that these two contracts possess disjoint sets of ports and variables. At a first sight, the assumptions of the composite should intuitively be simply the conjunction of the assumptions of the rich components, since the environment should satisfy all the assumptions. In general, however, part of the assumptions A_1 will be already satisfied by composing $\mathcal{C}_1$ with $\mathcal{C}_2$ acting as a partial environment for $\mathcal{C}_1$. Therefore, G_2 can contribute to relaxing assumption A_1, and vice-versa. Whence the following definition:

Definition 3.2 (Composition of contracts) *The parallel composition $\mathcal{C}_1 \,||\, \mathcal{C}_2$ is defined as the contract $\mathcal{C} = (A, G)$ such that:*

$$\begin{array}{rcl} A & = & (A_1 \cap A_2) \cup \neg(G_1 \cap G_2), \\ G & = & G_1 \cap G_2. \end{array}$$

Note that the so defined contract is in canonical form.

The following result, which, as discussed is not true in the general case, expresses the strong compositional properties of the L0 contract theory.

Lemma 3.3 *If $K \models \mathcal{C}$ and $K \models \mathcal{C}'$, then $K \cap K' \models \mathcal{C} \,||\, \mathcal{C}'$.*

Dominance in L0. Dominance is defined in L0 as a contravariant relation between assumptions and guarantees. The relation between guarantees is required for general dominance, whereas the second condition for assumptions is intended for preserving well-formedness.

Definition 3.4 (L0-Dominance) *A contract $\mathcal{C} = (A, G)$ dominates a contract $\mathcal{C}' = (A', G')$ if and only if $A \supseteq A'$ and $G \subseteq G'$.*

Dominance amounts to relaxing assumptions and reinforcing promises. Note that if $\mathcal{C}$ dominates $\mathcal{C}'$ and $\mathcal{C}'$ dominates $\mathcal{C}$, then $\mathcal{C} = \mathcal{C}'$. Furthermore, if $\mathcal{C}$ dominates $\mathcal{C}'$ then $M_{\mathcal{C}} \models \mathcal{C}'$. This property implies the following result, which relates the definition of dominance in L0 to the more general definition of our contract framework:

Lemma 3.5 *If $K \models \mathcal{C}$ and $\mathcal{C}$ dominates $\mathcal{C}'$, then $K \models \mathcal{C}'$.*

As a partial order, dominance admits both greatest lower bounds and least upper bounds, which we call conjunction and disjunction of contracts, respectively. Let $\mathcal{C}_1 = (A_1, G_1)$ and $\mathcal{C}_2 = (A_2, G_2)$ be contracts. The greatest lower bound of $\mathcal{C}_1$ and $\mathcal{C}_2$, written $\mathcal{C} = \mathcal{C}_1 \sqcap \mathcal{C}_2$, is given by $\mathcal{C} = (A, G)$ where $A = A_1 \cup A_2$ and $G = G_1 \cap G_2$. Similarly, the least upper bound of $\mathcal{C}_1$ and $\mathcal{C}_2$, written $\mathcal{C} = \mathcal{C}_1 \sqcup \mathcal{C}_2$, is given by $\mathcal{C}' = (A', G')$ where $A = A_1 \cap A_2$ and $G = G_1 \cup G_2$. Note that the result of these operations are contracts in canonical form. Minimal and maximal contracts can also be defined, as well as complementation, making L0 contracts a boolean algebra.

Discussion. The L0 contract framework has strong compositional properties, which derive from its simple definition and operators. The theory, however, depends on the effectiveness of certain operators, complementation in particular, which are necessary for the computation of canonical forms. While the complete theory can be formulated without the use of canonical forms, complementation remains fundamental in the definition of contract composition, which is at the basis of system construction.

Circular reasoning is sound for a contract framework based on canonical forms. This is because any behavior that is not allowed to the environment, is instead allowed by the guarantees. This is no longer the case for contracts which are *not* in canonical form. This is a limitation of the L0 framework, since working with canonical forms could prove computationally hard.

In the following section we analyze a higher level model, where we dispense with the composition operator for contracts. We will show that the high level framework, called L1, can be used consistently with L0 by proving that properties in L1 are preserved in L0 by soundness results.

3.2 L1 contract framework

The HRC L1 framework (1) allows to define more complex composition operators and (2) decouples the notions of conformance and refinement in a context in order to enable circular reasoning even for contracts that are not in normal form, and thus avoid the limitations of the L0 contract framework.

For defining the L1 contract framework, we follow the general theory exhibited in section 2. It is therefore enough to provide the L1 composition model defined by a set of connectors, the semantics of composition, and the two refinement relations and show that they satisfy the requirements of section 2.

L1 composition is based on *interactions*, which involve non-empty sets of ports. An interaction is defined by the components that synchronize when it takes place, and the ports through which these components synchronize. In the following, interactions are usually denoted α.

Definition 3.6 (L1 connectors) *A* connector *γ is defined by a support set S_γ, an exported port p_γ and a set $\mathcal{I}_\gamma$ of interactions in S_γ.*

The intuition behind *support set* and *exported port* is illustrated in Figure 2, where connectors relate a set of inner ports and an outer port of a component. One should keep in mind that a connector, and thus an exported port, represents a set of interactions rather than a single interaction.

Now, suppose given a set of interfaces $\mathcal{P}_i$ for components K_i and denote $P = \bigcup_i \mathcal{P}_i$. If $p_\gamma \subseteq S_{\gamma'}$, then γ and γ' form together a hierarchical connector whose interaction set is like $\mathcal{I}_{\gamma'}$ where each interaction α in which p_γ occurs is replaced by a set of interactions identical to α except that the occurrence of p_γ is replaced by an interaction of I_γ.

Two connectors γ_1 and γ_2 are *disjoint* if $p_{\gamma_1} \neq p_{\gamma_2}$, $p_{\gamma_1} \notin S_{\gamma_2}$ and $p_{\gamma_2} \notin S_{\gamma_1}$. Note that S_{γ_1} and S_{γ_2} may have ports in common, as a port may be connected to several connectors.

Definition 3.7 (L1 composition operators) *A composition operator Γ on a support set S_Γ is a set of disjoint connectors with support sets included in S_Γ.*

Composition operators can be composed using two operations: union and hierarchical composition of connectors. This defines an operator $\circ$ with the properties required by Definition 2.1.

Definition 3.8 (L1 composition semantics) *Given a set of LTS $\{K_i\}_{i=1}^n$ with $K_i = (Q_i, q_i^0, \mathcal{P}_i, \longrightarrow_i)$, and Γ a composition operator with support set $S_\Gamma = \bigcup_{i=1}^n \mathcal{P}_i$, we define the composition $\Gamma\{K_1, \dots, K_n\}$ as $(Q, q^0, \mathcal{P}_\Gamma, \longrightarrow)$, where $Q = \prod_{i=1}^n Q_i$, $q^0 = (q_1^0, \dots, q_n^0)$ and $\longrightarrow$ is defined by:*
$\forall \gamma \in \Gamma, \forall \alpha \in \mathcal{I}_\gamma, \forall q^1 = (q_1^1, \dots, q_n^1), q^2 = (q_1^2, \dots, q_n^2) \in Q$, $q^1 \xrightarrow{p_\gamma} q^2$ iff the following conditions hold:

- *$\forall i, q_i^1 \xrightarrow{\mathcal{P}_i \cap \alpha} q_i^2$, with the convention that $\forall q, q \xrightarrow{\emptyset} q$, that is, components not involved in the interaction don't move.*
- *α is maximal w.r.t. $\mathcal{I}_\gamma$, that is, there is no $\alpha' \in \mathcal{I}_\gamma$ such that $\alpha \subsetneq \alpha'$ and the previous condition holds for α'.*

Thus,the semantics of composition is obtained as usual as the composition of transition systems where each global transition corresponds to a maximal enabled interaction of exactly one connector, and the *label* of a transition is the exported port of the corresponding connector[2]. In order to define a synchronous composition, Γ must consist of a set of conflicting connectors which are never enabled jointly.

L1 contracts. Let $\mathcal{P}$ be an interface and Γ a composition operator with $\mathcal{P}$ in its support set, which implicitly defines an interface $\mathcal{P}_E$ of the environment. An L1-contract for $\mathcal{P}$ is then of the form (A, Γ, G) where A is an LTS on $\mathcal{P}_E$ and G an LTS on $\mathcal{P}$.

It now remains to define the two refinement relations, *conformance* and *refinement under context*. As in L1, we take into account the representation of behaviors as LTS and choose L1-conformance to be *simulation*, that is, the structural counter part of L0-conformance (trace inclusion).

Definition 3.9 (L1-conformance: simulation) *Let K_1 and K_2 be two LTS. A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a* simulation *from K_1 to K_2 iff $q_1^0 \,\mathcal{R}\, q_2^0$ and for any pair $(q_1, q_2) \in Q_1 \times Q_2$:*

$$q_1 \,\mathcal{R}\, q_2 \wedge q_1 \xrightarrow{\alpha}_1 q_1' \implies \exists q_2' \; s.t. \; q_2 \xrightarrow{\alpha}_2 q_2' \wedge q_1' \,\mathcal{R}\, q_2'$$

$K_1 \preccurlyeq^{L1} K_2$ iff there exists such a relation.

Thus, L1-conformance is identical L0-conformance for behaviors without non-observable non-determinism, and otherwise it is stronger. Note that in verification tools, in order to check trace inclusion efficiently, one will generally check simulation anyway.

For refinement under context, which defines contract satisfaction, we choose a stronger relation than for L0, in order to be able to use circular reasoning for dominance checks.

Definition 3.10 (L1 satisfaction) *Given a component K and a contract (A, gl, G) for $\mathcal{P}_K$, we define satisfaction as: $K \sqsubseteq_{A,gl}^{L1} G \triangleq$*

$$\begin{cases} gl\{K, A_{det}\} \preccurlyeq^{L1} gl\{G, A_{det}\} & (4) \\ (q_K, q_A) \,\mathcal{R}\, (q_G, q_A') \wedge q_K \xrightarrow{\alpha}_K \implies q_G \xrightarrow{\alpha}_G & (5) \end{cases}$$

where A_{det} is the determinization of A and $\mathcal{R}$ the relation on states proving that $gl\{K, A_{det}\} \preccurlyeq^{L1} gl\{G, A_{det}\}$.

That is, $\sqsubseteq^{L1}$ strengthens the "standard" notion of refinement under context as introduced in Example 2.4 and used in the L0 framework by the condition (5) stating that every transition of a refining state must correspond to a transition in each corresponding abstract state — but the target states must be related as well only if the environment allows this transition. As a consequence, $\sqsubseteq^{L1}$ allows circular reasoning. Note that in frameworks with data, one usually requires preservation of certain predicates from the concrete to the abstract transition system; here we require preservation of transition enabledness, independently of data.

[2]One can easily define two semantics of different granularity: we have chosen the one referring to exported ports, but alternatively we could distinguish all the different interactions for each connector.

L1-satisfaction is not the loosest relation ensuring sound circular reasoning. Condition (5) can be relaxed by not requiring that $q_G \xrightarrow{\alpha}_G$ if the environment *controls* α and A *forbids* in q_A any interaction α_A that can be combined with α — meaning that: for any α_A such that $\alpha \cup \alpha_A$ is an interaction of the closed system $gl\{K, A\}$, then α_A is also an interaction of the closed system. Thus, the symmetry is broken between environment and component, because all refinements of A must forbid such α_A in q_A (there is no q'_A s.t. $q_A \xrightarrow{\alpha_A}_A q'_A$). For the sake of simplicity, we do not use this property here.

Generally, having a structural part in contracts allows the refinement relation to rely on the way a component is connected to its environment, and not only on an abstraction of its behavior, to forbid (or allow) an interaction in the composition.

Well-formedness. A natural notion of well-formedness for L1 would be deadlock freedom, a property that is not preserved by composition. In [12], we define a framework based on modal transition systems, in which deadlock freedom is indeed a well-formedness condition, that is, preserved by conformance. Here, we use only L1 refinement under context to improve L0 reasoning. In particular, we are only interested in L0-conformance and well-formedness.

3.3 Consistency between L0 and L1

In the previous sections, we have introduced two different notions of refinement under context: $\sqsubseteq^{L0}$ and $\sqsubseteq^{L1}$ where the second is strictly stronger than the first one. As already stated, circular reasoning is sound for $\sqsubseteq^{L1}$ but not for $\sqsubseteq^{L0}$.

Now, based on the results of section 2.5, we provide a way of relaxing the dominance checking for L1 and symmetrically we propose a verification condition for L0-satisfaction that does not require that the actual environment refines the abstraction of the environment in any context. These two improvements are as explained in section 2.5 based respectively on the following two theorems.

Theorem 3.11 *$K \sqsubseteq^{L1}_{A,gl} G \wedge E \sqsubseteq^{L0}_{G,gl} A$ implies $K \sqsubseteq^{L1}_{E,gl} G$.*

This allows us to relax our sufficient condition for dominance in L1. This is particularly interesting because checking L0-satisfaction is obviously easier than checking L1-satisfaction. Thus, checking that a set of contracts $\{\mathcal{C}_i\}_{i=1}^n$ L1-dominates a contract $\mathcal{C}$ requires one L1-satisfaction check and n L0-satisfaction checks. Moreover, this relaxed rule allows establishing dominance more often.

Note that the well-formedness condition for L1 is not preserved using this rule. This is however no problem because only the well-formedness condition of L0 has to be preserved in the global methodology, which is indeed the case.

Theorem 3.12 *$K \sqsubseteq^{L0}_{A,gl} G \wedge E \sqsubseteq^{L1}_{G,gl} A$ implies $K \sqsubseteq^{L0}_{E,gl} G$.*

This second theorem allows in the SPEEDS project building a complete tool chain based on a set of tools checking either L0-satisfaction or L1-dominance (implemented by a set of L1-satisfaction checks). We can check a complete contract hierarchy requiring dominance check using the existing L1 dominance checker, and at the very end check satisfaction with the more scalable L0 satisfaction checker.

4 Related work

The L0 theory of contracts is similar to other notions of components with assumptions that are found in the literature. For process spaces [9, 10], a process (or component, or interface) has two sets: X is the set of possible behaviors, and Y is the set of acceptable behaviors. A process has the requirement that $X \cup Y = \mathcal{R}$, the set of all behaviors. A process is the same as a contract $\mathcal{C}$ that has:

$$G = X, \quad A = Y.$$

So, processes correspond to contracts in canonical form. The notion of refinement and contract composition is the same as the one described here.

A similar approach is obtained using trace structures with failure behaviors [5, 11]. A trace structure (or component, or interface) has also two sets: S is the set of successful traces, and F is the set of failure traces. One obtains a contract in canonical form $\mathcal{C}$ by setting

$$G = S \cup F, \quad A = \neg F.$$

Again, the notion of refinement and composition are the same as for contracts.

The trace structures developed by Dill [5], and subsequently by Wolf [15], are particularly interesting because they address the problem of receptiveness by proving closure properties and by giving decision procedures. These models were taken as inspiration to derive our initial notions, and fall in our general methodology as additional instances.

Interface theories are based on a fixed rather than a generic model of composition — usually synchronous Input/Output (I/O) composition. Besides, not all interface theories separate the assumption and the guarantee. This work encompasses all these formalisms. Let us underline here that keeping A and G separateimproves reusability [7]. Indeed, checking refinement under a new context can be achieved much more easily, for example if the new environment refines the old one in any context. Besides, an interface is necessarily a composition, thus keeping assumption and guarantee separate avoid storing their product.

In [13], we have shown the usefulness of the preliminary version of our general contract framework (presented in [12]) for establishing results for two different interface theories: one for classical I/O automata [7] and another for Modal I/O automata [8]. In both frameworks circular reasoning is sound — intuitively, because at any step the ports controlled by the component and those controlled by the environment are strictly separate. Nevertheless, [7, 8] present formal proofs which are quite long and technical. We have shown in [13] that those that can be encoded into our general framework become much simpler and clear.

5 Conclusion and future work

We have presented a contract framework based on a generic component algebra. We employ a methodology that uses three notions of refinement: conformance, dominance and satisfaction. We have shown how to derive these notions from refinement of open systems and refinement under context. We have then analyzed the compositional properties of these relationships, and discussed the issue of circular reasoning when using refinement under context. We have shown that by imposing a looser connection between refinement and refinement under context we are able to derive powerful rules for reasoning about contracts.

These theoretical results are implemented in the context of the HRC model developed in the SPEEDS project. We have used two interaction paradigms from HRC to show concrete examples of these relations, and how to combine the respective tool chains. With respect to the existing implementations, there is a lot of room for their improvement.

In [3], we provide a more detailed description of the design and verification methodology. In particular, we provide for every design step, a set of checks to be carried out. However, we have to work on strategies to be deployed when some check fails, which should depend on the failed check and on whether a top-down or bottom-up approach is followed, and which should be able to exploit more detailed diagnostic information when it exists.

References

[1] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis. Multiple viewpoint contract-based specification and design. In *Proc. of FMCO'07*, volume 5382 of *LNCS*, pages 200–225, 2008. 2.4

[2] L. Benvenuti, A. Ferrari, L. Mangeruca, E. Mazzi, R. Passerone, and C. Sofronis. A contract-based formalism for the specification of heterogeneous systems. In *Proc. of FDL'08*, Stuttgart, Germany, September 23–25, 2008. 2.1, 3.1

[3] S. Consortium. Speeds methodology for speculative and exploratory design in systems engineering. under submission for publication as a series of articles, 2010. 1, 2.1, 5

[4] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proc. of ESEC/SIGSOFT FSE'01*, pages 109–120. ACM Press, 2001. 2.2

[5] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*. ACM Distinguished Dissertations. MIT Press, 1989. 4

[6] G. Gößler and J. Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005. 3

[7] K. G. Larsen, U. Nyman, and A. Wasowski. Interface input/output automata. In *Proc. of FM'06*, volume 4085 of *LNCS*, pages 82–97, 2006. 4

[8] K. G. Larsen, U. Nyman, and A. Wasowski. Modal i/o automata for interface and product line theories. In *Proc. of ESOP'07*, volume 4421 of *LNCS*, pages 64–79, 2007. 4

[9] R. Negulescu. *Process Spaces and the Formal Verification of Asynchronous Circuits*. PhD thesis, University of Waterloo, Canada, 1998. 4

[10] R. Negulescu. Process spaces. In *Proc. of CONCUR'77*, volume 1877 of *LNCS*, 2000. 4

[11] R. Passerone. *Semantic Foundations for Heterogeneous Systems*. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94720, May 2004. 4

[12] S. Quinton and S. Graf. Contract-based verification of hierarchical systems of components. In *Proc. of SEFM'08*, pages 377–381. IEEE Computer Society, 2008. 2.1, 2.4, 3.2, 4

[13] S. Quinton and S. Graf. A framework for contract-based reasoning: Motivation and application. In proc. of FLACOS'08. Technical Report 377, University of Oslo, 2008. 2.3, 4

[14] J. Sifakis. A framework for component-based construction. In *Proc. of SEFM'05*, pages 293–300. IEEE Computer Society, 2005. 2.2

[15] E. S. Wolf. *Hierarchical Models of Synchronous Circuits for Formal Verification and Substitution*. PhD thesis, Department of Computer Science, Stanford University, October 1995. 4

A Condition for dominance

NB: For the proof with only one refinement under context ensuring soundness of refinement under context, one simply has to replace $\sqsubseteq^\alpha$ and $\sqsubseteq^\beta$ with $\sqsubseteq$.

Remember that the following theorem is given for $\sqsubseteq^\alpha$ stronger than $\sqsubseteq^\beta$.

Theorem 2.12 *If* $\forall i.\ \exists gl_{E_i}.\ gl \circ gl_I = gl_i \circ gl_{E_i}$ *and pseudo circular reasoning is sound, then to prove that* $\mathcal{C}$ *dominates* $\{\mathcal{C}_i\}_{i=1..n}$ *w.r.t. gl, it is sufficient to prove that:*

$$\begin{cases} \Pi_P(gl_I\{G_1,\ldots,G_n\}) \models^\alpha \mathcal{C} \\ \forall i, \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A,G_1,\ldots,G_{i-1},G_{i+1},\ldots,G_n\}) \models^\beta \mathcal{C}_i^{-1} \end{cases}$$

For all $i = 1..n$, let K_i be a component on $\mathcal{P}_i$. Suppose the following:

1. $\forall i.\ \exists gl_{E_i}.\ gl \circ gl_I = gl_i \circ gl_{E_i}$
2. $\Pi_P(gl_I\{G_1,\ldots,G_n\}) \sqsubseteq^\alpha_{A,gl} G$
3. $\forall i, \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A,G_1,\ldots,G_{i-1},G_{i+1},\ldots,G_n\}) \sqsubseteq^\beta_{G_i,gl_i} A_i$
4. $\forall i, K_i \sqsubseteq^\alpha_{A_i,gl_i} G_i$

We aim at proving $\Pi_P(gl_I\{K_1,\ldots,K_n\}) \models^\alpha \mathcal{C}$, that is, that: $\Pi_P(gl_I\{K_1,\ldots,K_n\}) \sqsubseteq^\alpha_{A,gl} G$. For this, we show by induction that for all l in $[\![0,n]\!]$, for all partition $\{J,K\}$ of $[\![1,n]\!]$ such that $|J| = l$:

$$\begin{cases} \Pi_P(gl_I\{\mathcal{K}_J \cup \mathcal{G}_K\}) \sqsubseteq^\alpha_{A,gl} G \\ \forall i \in K, \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A,\mathcal{E}^i_{J,K}\}) \sqsubseteq^\beta_{G_i,gl_i} A_i \end{cases}$$

with $\mathcal{K}_J = \{K_j\}_{j\in J}$, $\mathcal{G}_K = \{G_k\}_{k\in K}$ and with $\mathcal{E}^i_{J,K} = \mathcal{K}_J \cup (\mathcal{G}_K \backslash \{G_i\})$.

- $l = 0$. By (2) and (3) the property holds.
- $0 \leq l < n$. We suppose that our property holds for l. Let $\{J',K'\}$ be a partition of $[\![1,n]\!]$ such that $|J'| = l+1$. Let q be an element of J'. We fix $J = J' \backslash \{q\}$ and $K = K' \cup \{q\}$.

Step 1 We first prove that $\Pi_P(gl_I\{\mathcal{K}_{J'} \cup \mathcal{G}_{K'}\}) \sqsubseteq^\alpha_{A,gl} G$.

$$\begin{cases} K_q \sqsubseteq^\alpha_{A_q,gl_q} G_q \text{ from (5)} \\ \Pi_{\mathcal{P}_{A_q}}(gl_{E_q}\{A,\mathcal{E}^q_{J,K}\}) \sqsubseteq^\beta_{G_q,gl_q} A_q \end{cases}$$

The second property is our recurrence hypothesis, as $q \in K$. Thus, by circular reasoning:

$$K_q \sqsubseteq^\alpha_{\Pi_{\mathcal{P}_{A_q}}(gl_{E_q}\{A,\mathcal{E}^q_{J,K}\}),gl_q} G_q$$

By (1) and property 2. of Definition 2.10, this implies that:

$$K_q \sqsubseteq^\alpha_{gl_{E_q}\{A,\mathcal{E}^q_{J,K}\},gl\circ gl_I} G_q$$

As refinement under context is compositional, we obtain:

$$gl_I\{K_q,\mathcal{E}^q_{J,K}\} \sqsubseteq^\alpha_{A,gl} gl_I\{G_q,\mathcal{E}^q_{J,K}\}$$

This is equivalent to $gl_I\{\mathcal{K}_{J'} \cup \mathcal{G}_{K'}\} \sqsubseteq^\alpha_{A,gl} gl_I\{\mathcal{K}_J \cup \mathcal{G}_K\}$.
Finally, by using property 1. of Definition 2.10 and the recurrence hypothesis: $\Pi_P(gl_I\{\mathcal{K}_{J'} \cup \mathcal{G}_{K'}\}) \sqsubseteq^\alpha_{A,gl} G$.

Step 2 We now have to prove that:

$$\forall i \in K'.\ \Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A, \mathcal{E}^i_{J',K'}\}) \sqsubseteq^{\beta}_{G_i,gl_i} A_i$$

We fix $i \in K'$. We have proved in step 1 that

$$K_q \sqsubseteq^{\alpha}_{gl_{E_q}\{A,\mathcal{E}^q_{J,K}\},gl\circ gl_I} G_q$$

As $\sqsubseteq^{\alpha}$ is stronger than $\sqsubseteq^{\beta}$, we also have:

$$K_q \sqsubseteq^{\beta}_{gl_{E_q}\{A,\mathcal{E}^q_{J,K}\},gl\circ gl_I} G_q$$

$K = K' \cup \{q\}$ so $i \in K$. Thus, by compositionality of refinement under context, we have:

$$gl_{E_i}\{K_q, A, \mathcal{E}^q_{J,K\setminus\{i\}}\} \sqsubseteq^{\beta}_{G_i,gl_i} gl_{E_i}\{G_q, A, \mathcal{E}^q_{J,K\setminus\{i\}}\}$$

This boils down to $gl_{E_i}\{A, \mathcal{E}^i_{J',K'}\} \sqsubseteq^{\beta}_{G_i,gl_i} gl_{E_i}\{A, \mathcal{E}^i_{J,K}\}$.
Hence, using property 1. of projection and the recurrence hypothesis: $\Pi_{\mathcal{P}_{A_i}}(gl_{E_i}\{A, \mathcal{E}^i_{J',K'}\}) \sqsubseteq^{\beta}_{G_i,gl_i} A_i$.

Conclusion By applying our property to $l = n$, we get:

$$\Pi_P(gl_I\{K_1, \dots, K_n\}) \sqsubseteq^{\alpha}_{A,gl} G$$

□

B Consistency between L0 and L1

We prove here the following theorems:

1. L1-satisfaction implies L0-satisfaction
2. Soundness of circular circular reasoning for $\sqsubseteq^{L1}$
3. Soundness of pseudo circular reasoning for $\sqsubseteq^{L1}$ and $\sqsubseteq^{L0}$
4. Soundness of pseudo circular reasoning for $\sqsubseteq^{L0}$ and $\sqsubseteq^{L1}$

We also provide characterizations of $\sqsubseteq^{L1}$ and $\sqsubseteq^{L0}$ which are used in the proofs.

Transitions are labeled by interactions rather than by the exported port of the corresponding connector.

Lemma B.1 *$\Gamma\{K_1, E\}$ simulates $\Gamma\{K_2, E\}$ if and only if there exists a relation $\mathcal{R} \subseteq (Q_1 \times Q_E) \times Q_2$ such that:*

- $(q_1^0, q_E^0)\,\mathcal{R}\,q_2^0$
- *If $(q_1, q_E)\,\mathcal{R}\,q_2$ and $(q_1, q_E) \xrightarrow{\alpha} (q_1', q_E')$ for $\alpha \in \mathcal{I}_\Gamma$ such that $\alpha = \alpha_P \cup \alpha_E$, then $\exists q_2'$ s.t. $q_2 \xrightarrow{\alpha_P} q_2'$ and $(q_1', q_E')\,\mathcal{R}\,q_2'$.*

B.1 Characterization of L1-satisfaction

Lemma B.2 *$K_1 \sqsubseteq^{L1}_{E,\Gamma} K_2$ if and only if there exists a relation $\mathcal{R} \subseteq (Q_1 \times 2^{Q_E}) \times Q_2$ such that:*

- $(q_1^0, \{q_E^0\})\,\mathcal{R}\,q_2^0$
- *If $(q_1, \mathcal{Q}_E)\,\mathcal{R}\,q_2$ and $q_1 \xrightarrow{\alpha_P}_1 q_1'$, there exists q_2' such that:*
  1. *$q_2 \xrightarrow{\alpha_P}_2 q_2'$*
  2. *if there exists α_E, $q_E \in \mathcal{Q}_E$ and $q_E' \in Q_E$ such that $q_E \xrightarrow{\alpha_E}_E q_E'$ and $\alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$, then $(q_1', \mathcal{Q}_E')\,\mathcal{R}\,q_2'$ where $\mathcal{Q}_E'$ is $\{q_E' \mid \exists q_E \in \mathcal{Q}_E \text{ s.t. } q_E \xrightarrow{\alpha_E}_E q_E'\}$.*

We use the convention that $\forall q \xrightarrow{\emptyset} q$, so the above condition includes cases where only K_1 or only E move on.

B.2 Characterization of L0-satisfaction

Lemma B.3 *$K_1 \sqsubseteq^{L0}_{E,\mathcal{I}} K_2$ if and only if there exists a relation $\mathcal{R} \subseteq (2^{Q_1} \times Q_E) \times 2^{Q_2}$ such that:*

- *$(\{q_1^0\}, q_E^0)\, \mathcal{R}\, \{q_2^0\}$*
- *If $(\mathcal{Q}_1, q_E)\, \mathcal{R}\, \mathcal{Q}_2$ and $(q_1, q_E) \xrightarrow{\alpha} (q_1', q_E')$ for $q_1 \in \mathcal{Q}_1$ with $\alpha = \alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$, then there exists $q_2 \in \mathcal{Q}_2$ and $q_2' \in Q_2$ such that $q_2 \xrightarrow{\alpha_P} q_2'$ and $(\mathcal{Q}_1', q_E')\, \mathcal{R}\, \mathcal{Q}_2'$ for $\mathcal{Q}_1'$ defined as $\{q_1' \mid \exists q_1 \in \mathcal{Q}_1\ s.t.\ q_1 \xrightarrow{\alpha_P}_1 q_1'\}$ and $\mathcal{Q}_2'$ defined as $\{q_2' \mid \exists q_2 \in \mathcal{Q}_2\ s.t.\ q_2 \xrightarrow{\alpha_P}_2 q_2'\}$.*

We use the convention that $\forall q \xrightarrow{\emptyset} q$, so the above condition includes cases where only K_1 or only E moves on.

B.3 L1-satisfaction implies L0-satisfaction

We suppose that $K_1 \sqsubseteq^{L1}_{E,\Gamma} K_2$, and then we prove that $K_1 \sqsubseteq^{L0}_{E,\Gamma} K_2$.

As $K_1 \sqsubseteq^{L1}_{E,\Gamma} K_2$, there exists a relation $\mathcal{R} \subseteq (Q_1 \times 2^{Q_E}) \times Q_2$ as in Lemma B.2.

We define $\mathcal{R}' \subseteq (2^{Q_1} \times Q_E) \times 2^{Q_2}$ as follows: we define $(\{q_1\}, q_E)\, \mathcal{R}'\, \{q_2\}$ iff there exists $\mathcal{Q}_E \subseteq Q_E$ such that $q_E \in \mathcal{Q}_E$ and $(q_1, \mathcal{Q}_E)\, \mathcal{R}\, q_2$.

Obviously $(\{q_1^0\}, q_E^0)\, \mathcal{R}'\, \{q_2^0\}$.

Now suppose $(\{q_1\}, q_E)\, \mathcal{R}'\, \{q_2\}$ and $(q_1, q_E) \xrightarrow{\alpha} (q_1', q_E')$ with $\alpha = \alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$. We must show that there exists $q_2' \in Q_2$ such that $q_2 \xrightarrow{\alpha_P} q_2'$ and $(\{q_1'\}, q_E')\, \mathcal{R}'\, \{q_2'\}$.

Let $\mathcal{Q}_E \subseteq Q_E$ be such that $q_E \in \mathcal{Q}_E$ and $(q_1, \mathcal{Q}_E)\, \mathcal{R}\, q_2$. We have $q_1 \xrightarrow{\alpha_P}_1 q_1'$. Thus, there exists q_2' such that $q_2 \xrightarrow{\alpha_P}_2 q_2'$.

Besides, as $q_E \xrightarrow{\alpha_E}_E q_E'$, we have $(q_1', \mathcal{Q}_E')\, \mathcal{R}\, q_2'$ where $\mathcal{Q}_E'$ is $\{q_E' \mid \exists q_E \in \mathcal{Q}_E\ s.t.\ q_E \xrightarrow{\alpha_E}_E q_E'\}$.

Hence, by definition of $\mathcal{R}'$: $(\{q_1'\}, q_E')\, \mathcal{R}'\, \{q_2'\}$.

□

B.4 Soundness of circular reasoning for $\sqsubseteq^{L1}$

Let K be a component on $\mathcal{P}$, (E, Γ) a context for $\mathcal{P}$ and $\mathcal{C} = (A, \Gamma, G)$ a contract for $\mathcal{P}$. Suppose that $K \sqsubseteq^{L1}_{A,\Gamma} G \wedge E \sqsubseteq^{L1}_{G,\Gamma} A$. We have to prove that $K \sqsubseteq^{L1}_{E,\Gamma} G$.

As $K \sqsubseteq^{L1}_{A,\Gamma} G$ and $E \sqsubseteq^{L1}_{G,\Gamma} A$, there exist two relations $\mathcal{R}_1$ and $\mathcal{R}_2$ on respectively $(Q_K \times 2^{Q_A}) \times Q_G$ and $(Q_E \times 2^{Q_G}) \times Q_A$ as in Lemma B.2. We define $\mathcal{R} \subseteq (Q_K \times 2^{Q_E}) \times Q_G$ as follows: for all $q_K \in Q_K$, $\mathcal{Q}_E \subseteq Q_E$ and $q_G \in Q_G$, $(q_K, \mathcal{Q}_E)\, \mathcal{R}\, q_G$ iff there exists $\mathcal{Q}_A \subseteq Q_A$, $\mathcal{Q}_G \subseteq Q_G$, $q_E \in \mathcal{Q}_E$ and $q_A \in \mathcal{Q}_A$ such that $q_G \in \mathcal{Q}_G$, $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$.

We have to prove that $\mathcal{R}$ ensures the conditions of Lemma B.2. Obviously, $(q_K^0, \{q_E^0\})\, \mathcal{R}\, q_G^0$.

Let $q_K \in Q_K$, $\mathcal{Q}_E \subseteq Q_E$ and $q_G \in Q_G$ be such that $(q_K, \mathcal{Q}_E)\, \mathcal{R}\, q_G$. Let $\mathcal{Q}_A \subseteq Q_A$, $\mathcal{Q}_G \subseteq Q_G$, $q_E \in \mathcal{Q}_E$ and $q_A \in \mathcal{Q}_A$ be such that $q_G \in \mathcal{Q}_G$, $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$. Now suppose $q_K \xrightarrow{\alpha_P}_K q_K'$. We have to prove that there exists q_G' such that:

1. $q_G \xrightarrow{\alpha_P}_G q_G'$
2. if there exists α_E, $q_E \in \mathcal{Q}_E$ and $q_E' \in Q_E$ such that $q_E \xrightarrow{\alpha_E}_E q_E'$ and $\alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$, then $(q_K', \mathcal{Q}_E')\, \mathcal{R}\, q_G'$ where $\mathcal{Q}_E'$ is $\{q_E' \mid \exists q_E \in \mathcal{Q}_E\ s.t.\ q_E \xrightarrow{\alpha_E}_E q_E'\}$.

Because $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $q_K \xrightarrow{\alpha_P}_K q_K'$, we know that there exists q_G' such that:

- $q_G \xrightarrow{\alpha_P}_G q_G'$
- if there exists α_A, $q_A \in \mathcal{Q}_A$ and $q_A' \in Q_A$ such that $q_A \xrightarrow{\alpha_A}_A q_A'$ and $\alpha_P \cup \alpha_A \in \mathcal{I}_\Gamma$, then $(q_K', \mathcal{Q}_A')\, \mathcal{R}\, q_G'$ with $\mathcal{Q}_A'$ defined as $\{q_A' \mid \exists q_A \in \mathcal{Q}_A\ s.t.\ q_A \xrightarrow{\alpha_A}_A q_A'\}$.

We show that this q'_G satisfies the two conditions required above from $\mathcal{R}$. Condition 1. is exactly the same as for $\mathcal{R}_1$.
Let us show that the second condition holds. Suppose that there exists α_E, $q_E \in \mathcal{Q}_E$ and $q'_E \in Q_E$ such that $q_E \xrightarrow{\alpha_E}_E q'_E$ and $\alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$. Let $\mathcal{Q}'_E$ be defined as above by $\mathcal{Q}'_E = \{q'_E \mid \exists q_E \in \mathcal{Q}_E \ s.t. \ q_E \xrightarrow{\alpha_E}_E q'_E\}$. We have to show that $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$.

As $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$ and $q_E \xrightarrow{\alpha_E}_E q'_E$, we know that there exists q'_A such that:

- $q_A \xrightarrow{\alpha_E}_A q'_A$
- if there exists α_G, $q_G \in \mathcal{Q}_G$ and $q'_G \in Q_G$ such that $q_G \xrightarrow{\alpha_G}_G q'_G$ and $\alpha_G \cup \alpha_E \in \mathcal{I}_\Gamma$, then $(q'_E, \mathcal{Q}'_G)\, \mathcal{R}\, q'_A$ with $\mathcal{Q}'_G$ defined as $\{q'_G \mid \exists q_G \in \mathcal{Q}_G \ s.t. \ q_G \xrightarrow{\alpha_G}_G q'_G\}$.

Thus, applying the second property offered by $\mathcal{R}_1$ to this α_E and q'_A, we obtain that $(q'_K, \mathcal{Q}'_A)\, \mathcal{R}_1\, q'_G$ where $\mathcal{Q}'_A$ is defined as $\{q'_A \mid \exists q_A \in \mathcal{Q}_A \ s.t. \ q_A \xrightarrow{\alpha_E}_A q'_A\}$.
Besides, as there exist indeed $q_G \in \mathcal{Q}_G$ and $q'_G \in Q_G$ such that $q_G \xrightarrow{\alpha_P}_G q'_G$, then applying the second property offered by $\mathcal{R}_2$, we obtain $(q'_E, \mathcal{Q}'_G)\, \mathcal{R}_2\, q'_A$ for $\mathcal{Q}'_G$ defined as $\{q'_G \mid \exists q_G \in \mathcal{Q}_G \ s.t. \ q_G \xrightarrow{\alpha_P}_G q'_G\}$.
Finally, according to the definition of $\mathcal{R}$, we can conclude that $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$.

□

B.5 Pseudo circular reasoning for $\sqsubseteq^{L1}$ and $\sqsubseteq^{L0}$

Let K be a component on an interface $\mathcal{P}$, (E, Γ) a context for $\mathcal{P}$ and $\mathcal{C} = (A, \Gamma, G)$ a contract for $\mathcal{P}$. We suppose that $K \sqsubseteq^{L1}_{A,\Gamma} G$ and $E \sqsubseteq^{L0}_{G,\Gamma} A$, and then we prove that $K \sqsubseteq^{L1}_{E,\Gamma} G$.

As $K \sqsubseteq^{L1}_{A,\Gamma} G$, there exists a relation $\mathcal{R}_1$ on $(Q_K \times 2^{Q_A}) \times Q_G$ as in Lemma B.2.

As $E \sqsubseteq^{L0}_{G,\Gamma} A$, there exists a relation $\mathcal{R}_2$ on $(2^{Q_E} \times Q_G) \times 2^{Q_A}$ as in Lemma B.3.

We define $\mathcal{R} \subseteq (Q_K \times 2^{Q_E}) \times Q_G$ as follows: for all $q_K \in Q_K$, $\mathcal{Q}_E \subseteq Q_E$ and $q_G \in Q_G$, we define $(q_K, \mathcal{Q}_E)\, \mathcal{R}\, q_G$ iff there exists $\mathcal{Q}_A \subseteq Q_A$ such that $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $(\mathcal{Q}_E, q_G)\, \mathcal{R}_2\, \mathcal{Q}_A$.

We have to prove that $\mathcal{R}$ ensures the conditions of Lemma B.2. Obviously, $(q^0_K, \{q^0_E\})\, \mathcal{R}\, q^0_G$.

Let $q_K \in Q_K, \mathcal{Q}_E \subseteq Q_E, q_G \in Q_G$ be such that $(q_K, \mathcal{Q}_E)\, \mathcal{R}\, q_G$. Let $\mathcal{Q}_A$ be such that $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $(\mathcal{Q}_E, q_G)\, \mathcal{R}_2\, \mathcal{Q}_A$. Now suppose $q_K \xrightarrow{\alpha_P}_K q'_K$. We have to prove that there exists p'_G such that:

1. $q_G \xrightarrow{\alpha_P}_G q'_G$
2. if there exists α_E, $q_E \in \mathcal{Q}_E$ and $q'_E \in Q_E$ such that $q_E \xrightarrow{\alpha_E}_E q'_E$ and $\alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$, then $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$ where $\mathcal{Q}'_E$ is $\{q'_E \mid \exists q_E \in \mathcal{Q}_E \ s.t. \ q_E \xrightarrow{\alpha_E}_E q'_E\}$.

Because $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and $q_K \xrightarrow{\alpha_P}_K q'_K$, we know that there exists q'_G such that:

- $q_G \xrightarrow{\alpha_P}_G q'_G$
- if there exists α_A, $q_A \in \mathcal{Q}_A$ and $q'_A \in Q_A$ such that $q_A \xrightarrow{\alpha_A}_A q'_A$ and $\alpha_P \cup \alpha_A \in \mathcal{I}_\Gamma$, then $(q'_K, \mathcal{Q}'_A)\, \mathcal{R}\, q'_G$ with $\mathcal{Q}'_A$ defined as $\{q'_A \mid \exists q_A \in \mathcal{Q}_A \ s.t. \ q_A \xrightarrow{\alpha_A}_A q'_A\}$.

We show that this q'_G satisfies the two conditions required above from $\mathcal{R}$. Condition 1. is exactly the same as for $\mathcal{R}_1$.
Let us show that the second condition holds. Suppose there exist α_E, $q_E \in \mathcal{Q}_E$ and $q'_E \in Q_E$ such that $q_E \xrightarrow{\alpha_E}_E q'_E$ and $\alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$. Let $\mathcal{Q}'_E$ be defined as above by $\mathcal{Q}'_E = \{q'_E \mid \exists q_E \in \mathcal{Q}_E \ s.t. \ q_E \xrightarrow{\alpha_E}_E q'_E\}$.
We have to show that $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$.

As $q_E \xrightarrow{\alpha_E}_E q'_E$ and $q_G \xrightarrow{\alpha_P}_G q'_G$, we know that $(q_E, q_G) \xrightarrow{\alpha} (q'_E, q'_G)$. Thus, from $(\mathcal{Q}_E, q_G)\, \mathcal{R}_2\, Q_A$ and because $q_E \in \mathcal{Q}_E$, we can conclude that there exist $q_A \in \mathcal{Q}_A$ and $q'_A \in Q_A$ such that $q_A \xrightarrow{\alpha_E} q'_A$ and $(\mathcal{Q}'_E, q'_G)\, \mathcal{R}_2\, \mathcal{Q}'_A$ for $\mathcal{Q}'_E$ defined as above and $\mathcal{Q}'_A$ defined as $\{q'_A \mid \exists q_A \in \mathcal{Q}_A \ s.t. \ q_A \xrightarrow{\alpha_E}_A q'_A\}$.

Now, applying the second property offered by $\mathcal{R}_1$ to this α_E, q_A and q'_A, we obtain that $(q'_K, \mathcal{Q}'_A)\, \mathcal{R}_1\, q'_G$ with $\mathcal{Q}_A$ as defined above.
Finally, according to the definition of $\mathcal{R}$, we can conclude that $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$.

□

B.6 Pseudo circular reasoning for $\sqsubseteq^{L0}$ and $\sqsubseteq^{L1}$

Let K be a component on an interface $\mathcal{P}$, (E, Γ) a context for $\mathcal{P}$ and $\mathcal{C} = (A, \Gamma, G)$ a contract for $\mathcal{P}$. We suppose that $K \sqsubseteq^{L0}_{A,\Gamma} G$ and $E \sqsubseteq^{L1}_{G,\Gamma} A$, and then we prove that $K \sqsubseteq^{L0}_{E,\Gamma} G$.

As $K \sqsubseteq^{L0}_{A,\Gamma} G$, there exists a relation $\mathcal{R}_1$ on $(2^{Q_K} \times Q_A) \times 2^{Q_G}$ as in Lemma B.3.

As $E \sqsubseteq^{L1}_{G,\Gamma} A$, there exists a relation $\mathcal{R}_2$ on $(Q_E \times 2^{Q_G}) \times Q_A$ as in Lemma B.2.

We define $\mathcal{R} \subseteq (2^{Q_K} \times Q_E) \times 2^{Q_G}$ as follows: for all $\mathcal{Q}_K \subseteq Q_K$, $q_E \in Q_E$ and $\mathcal{Q}_G \subseteq Q_G$, we define $(\mathcal{Q}_K, q_E)\, \mathcal{R}\, \mathcal{Q}_G$ iff there exists $q_A \in Q_A$ such that $(\mathcal{Q}_K, q_A)\, \mathcal{R}_1\, \mathcal{Q}_G$ and $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$.

We have to prove that $\mathcal{R}$ ensures the conditions of Lemma B.3. Obviously, $(\{q^0_K\}, q^0_E)\, \mathcal{R}\, \{q^0_G\}$.

Let $\mathcal{Q}_K \subseteq Q_K$, $q_E \in Q_E$ and $\mathcal{Q}_G \subseteq Q_G$ be such that $(\mathcal{Q}_K, q_E)\, \mathcal{R}\, \mathcal{Q}_G$. Let q_A be such that $(\mathcal{Q}_K, q_A)\, \mathcal{R}_1\, \mathcal{Q}_G$ and $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$. Now suppose $(q_K, q_E) \xrightarrow{\alpha} (q'_K, q'_E)$ for $q_K \in \mathcal{Q}_K$ with $\alpha = \alpha_P \cup \alpha_E \in \mathcal{I}_\Gamma$. Let $\mathcal{Q}'_K$ be defined as $\{q'_K \mid \exists q_K \in \mathcal{Q}_K\ s.t.\ q_K \xrightarrow{\alpha_P}_K q'_K\}$ and $\mathcal{Q}'_G$ as $\{q'_G \mid \exists q_G \in \mathcal{Q}_G\ s.t.\ q_G \xrightarrow{\alpha_P}_G q'_G\}$. We have to prove that there exists $q_G \in \mathcal{Q}_G$ and $q'_G \in Q_G$ such that $q_G \xrightarrow{\alpha_P} q'_G$ and that $(\mathcal{Q}'_K, q'_E)\, \mathcal{R}\, \mathcal{Q}'_G$.

As $(q_K, q_E) \xrightarrow{\alpha} (q'_K, q'_E)$, we know that $q_E \xrightarrow{\alpha_E}_E q'_E$. Then, because $(q_E, \mathcal{Q}_G)\, \mathcal{R}_2\, q_A$ and $q_E \xrightarrow{\alpha_E}_E q'_E$, there exists q'_A such that:

- $q_A \xrightarrow{\alpha_E}_A q'_A$
- if there exists α_G, $q_G \in \mathcal{Q}_G$ and $q'_G \in Q_G$ such that $q_G \xrightarrow{\alpha_G}_G q'_G$ and $\alpha_G \cup \alpha_E \in \mathcal{I}_\Gamma$, then $(q'_E, \mathcal{Q}'_G)\, \mathcal{R}\, q'_A$ with $\mathcal{Q}'_G$ defined as above.

This implies in particular that $(q_K, q_A) \xrightarrow{\alpha} (q'_K, q'_A)$. Thus, from $(q_K, \mathcal{Q}_A)\, \mathcal{R}_1\, q_G$ and because $q_A \in \mathcal{Q}_A$, we can conclude that there exists $q_G \in \mathcal{Q}_G$ and $q'_G \in Q_G$ such that $q_G \xrightarrow{\alpha_P} q'_G$ and $(\mathcal{Q}'_K, q'_A)\, \mathcal{R}_1\, \mathcal{Q}'_G$ for $\mathcal{Q}'_K$ and $\mathcal{Q}'_G$ defined as above.

This gives us the q_G and q'_G we were looking for. There only remains to prove that $(\mathcal{Q}'_K, q'_E)\, \mathcal{R}\, \mathcal{Q}'_G$.

Now, applying the second property offered by $\mathcal{R}_2$ to this α_P, q_G and q'_G, we obtain that $(q'_E, \mathcal{Q}'_G)\, \mathcal{R}_2\, q'_A$.

Hence, according to the definition of $\mathcal{R}$, the conclusion that $(q'_K, \mathcal{Q}'_E)\, \mathcal{R}\, q'_G$.

$\square$