Component Interfaces with Contracts on Ports*

Sebastian Bauer[1], Rolf Hennicker[1], and Axel Legay[2]

[1] Ludwig-Maximilians-Universität München, Germany
[2] INRIA/IRISA Rennes, France

Abstract. We show how the abstract concept of a (labeled) interface theory can be canonically extended to an abstract framework for component interfaces with ports. The resulting component framework satisfies itself the general laws of an interface theory (concerning the composition, refinement, and environment correctness notions). The ports of a component interface represent the interaction points of a component. Each port is equipped with a contract specifying the assumptions on and the guarantees for the environment of a component. As a particular instance we consider modal component interfaces such that component behaviors and the assume and guarantee behaviors of ports are given in terms of modal I/O-transition systems with weak modal refinement and with a weak modal environment correctness notion. The modal approach is particularly useful to specify loose environment assumptions.

1 Introduction

The development of large, reliable component systems relies heavily on the use of interfaces. Hence, rigorous development methods are mandatory which support interface composition, stepwise refinement and the consideration of compatibility issues when a component is put in an environment. These requirements together with concise rules how the different dimensions of system development should work together are formulated in an abstract way in the seminal work of De Alfaro and Henzinger [10]. There the notion of an *interface theory* has been introduced which consists of an interface algebra together with a component algebra thus distinguishing interface specifications and component implementations.

In this paper we follow the idea of De Alfaro and Henzinger to study abstract concepts and rules that later on can be instantiated by concrete frameworks. But we will focus more specifically on the domain of reactive component systems such that interfaces should be equipped with additional structure that makes more explicit their possible connections. For that purpose we rely on ports as interaction points of a component as it is quite standard in many design languages.

Independently, a number of contract theories, based on assume-guarantee (AG) reasoning have been developed, with a similar aim of approaching compositional design. Contract theories differ from interface theories in that they strictly follow the principle of separation of concerns. They separate the specification of assumptions from specification of guarantees, a choice largely inspired

* This work has been partially sponsored by the EU project ASCENS, 257414.

C.S. Păsăreanu and G. Salaün (Eds.): FACS 2012, LNCS 7684, pp. 19–35, 2013.

by early ideas on manual proof methods of Misra, Chandy [25] and Jones [18], along with the wide acceptance to pre-/post-condition style of specification in programming [24,30], and more general semantical rules independent from language representation [8].

In [4], we have shown how a theory of contracts can be built on top of a given abstract specification theory. Contracts are just pairs (A, G) of an assumption and a guarantee specification. We have shown in [4] how the contract theory can be instantiated by using modal transitions systems [29] with strong modal refinement. This approach, however, did only work for specification theories which admit a "quotient" construction as specification building primitive and therefore could not be applied to instances that support weak refinement abstracting away silent τ-transitions [5] which is much more powerful. Compatibility issues concerning the communication between modal transition systems have not been integrated in [4]. On the other hand, the entities of our contract theory were just pairs (A, G) disallowing any structural splitting which is necessary if we want to deal with components with more than one port.

In the current paper we first introduce the notion of a labeled interface theory in Sect. 2, which resembles an interface theory in the sense of De Alfaro and Henzinger with the additional provision that a set of labels is assigned to any interface (which intuitively represents an action alphabet). Moreover, in addition to interface refinement, we introduce an environment correctness relation $S \rightarrow E$ to express when an environment E satisfies the interaction requirements of an interface S. We show, in Sect. 3, how a theory of component interfaces can be defined on top of any framework satisfying our abstract rules of an interface theory. A distinguished feature of component interfaces is that they have a set of ports such that each port P is equipped with a port contract (A^P, G^P) specifying the assumptions on the environment that is going to be connected on this particular port, and the guarantees of the component on that port. Hence our approach deviates from approaches that use single port protocols not allowing to extract distinguished assumptions and guarantees. All notions of an interface theory, i.e. composition, refinement and environment correctness, are propagated to the level of component interfaces which themselves are shown to satisfy the requirements of an interface theory. We also discuss reliability of component interfaces which means that the component frame, intended to specify the overall visible behavior of a component, supports the guarantees shown on the ports. We prove that reliability is compositional.

As a proof of concept, we instantiate in Sect. 4 our generic constructions and build a modal theory of component interfaces on top of a labeled interface theory with modal I/O-transitions systems and weak modal refinement as a basis [21],[5]. In particular, we consider a small case study in Sect. 4.2.

Related Work. As observed above, our work extends classical interface theories [10,12,7] with an explicit treatment of assumptions-guarantees. Other works on interface automata, e.g. [14], exploit the concept of assumption and guarantee to improve the efficiency of compatibility checking. However, they are not

comparable to our approach as they exploit assumption and guarantee at the operational level, but not at the design one. An intermediary step between those approaches is the work of Parizek and Plasil [26] that proposes a compositional methodology to reduce the verification of a composite component to the one of a series of smaller verifications on single components. Recently, a similar approach to the one of [26] was followed in the BIP toolset developed by Sifakis et al. [3].

Independently, a number of contract theories, based on explicit assume-guarantee reasoning have been developed, with a similar aim of approaching the compositional design. Among them, one finds the work of Meyer [24], that is based on pre and post conditions as state predicates and invariants for the system itself. This approach, which builds on seminal ideas proposed by Dijkstra and Lamport [13,19], is similar to ours in the sense that pre and post conditions shall be viewed as assumption and guarantee, respectively.

Some works [2] introduced contracts in the refinement calculus. In this formalism, processes are described with guarded command operating on shared variables. This formalism is best suited to reason on untimed system, while our approach is general and could be instantiated on other types of data. Additionally, each of the above mentioned work suffers from the absence of multiple treatment of assumptions/guarantees and rely on a unique language while our abstract language can work with arbitrary interface theories.

More recently, Benveniste et al.[6] proposed a contract theory in where assumption and guarantees are represented by trace structures. While this work is of clear interest, it suffers from the absence of effective representation for the embedded interface theory. Extensions such as the one proposed in [28,15] leverage this problem but ignore the multiple treatment of assumptions and guarantees.

2 Labeled Interface Theories

The idea of an interface theory is to capture basic requirements that should be satisfied by any formal framework supporting behavior specifications of components. We assume given a set $\mathfrak{S}$ of interface specifications such that any interface is equipped with a finite set of labels (representing the alphabet of actions an interface may perform). An interface theory includes a composition operator $\otimes$ to combine interfaces to larger ones. The composition operator is, in general, partial since it is not always syntactically meaningful to compose interfaces, due to syntactic constraints. Additionally, an interface theory must offer a refinement relation $\leq$ to relate "concrete" and "abstract" specifications, i.e. $S \leq T$ means that S is a correct refinement of T. Intuitively, the refinement relation expresses that the implementation requirements of the (abstract) interface T are respected by the refinement S. Refinement must be compositional in the sense that it must be preserved by the composition operator expressed by requirement (A1) below. An interface theory must also address the relationship between components and their environment. For this purpose we introduce an environment correctness relation $\rightarrow$ such that $S \rightarrow E$ means that E is a correct environment for S. Intuitively, this relation expresses that the communication requirements of S are

satisfied by the environment E (which is itself just another interface); we may say that S "feels well" in the environment E. Hence, the environment correctness relation is unidirectional and it is orthogonal to the refinement relation; the former concerns the "horizontal" dimension while the latter concerns the "vertical" dimension of system development. Both relations must be compatible in the sense that environment correctness must be preserved by refinement as stated in requirement (A2) below. This means that interface specifications and correct environments can be replaced by specialized versions without disrupting the correctness of the environment. Requirement (A3) concerns the relation between interface composition and environment correctness. Intuitively, it states that correct environments can be composed to a larger correct environment. More precisely, if S in the context of E feels well in E' and if S in the context of E' feels well in E, then S feels well in the larger environment $E \otimes E'$.

Definition 1 (Labeled Interface Theory). *A* labeled interface theory *is a quadruple* $(\mathfrak{S}, \mathcal{L}, \ell, \otimes, \leq, \rightarrow)$ *consisting of*

- *a set* $\mathfrak{S}$ *of interface specifications,*
- *a set* $\mathcal{L}$ *of labels,*
- *a function* $\ell : \mathfrak{S} \to \wp_{\text{fin}}(\mathcal{L})$ *assigning a finite set of labels to each interface,*
- *a partial, commutative*[1] *composition operator* $\otimes : \mathfrak{S} \times \mathfrak{S} \to \mathfrak{S}$*; we call* S *and* E *composable, if* $S \otimes E$ *is defined and require the following rules for composable interfaces:*
  - *C1. If* $S \otimes E$ *is defined, then* $\ell(S \otimes E) = (\ell(S) \cup \ell(E)) \setminus (\ell(S) \cap \ell(E))$.
  - *C2. If* $\ell(S) \cap \ell(E) = \emptyset$*, then* $S \otimes E$ *is defined.*
  - *C3. Pseudo-associativity: If* S, E *and* E' *are pairwise composable and* $\ell(S) \cap \ell(E) \cap \ell(E') = \emptyset$*, then* $(S \otimes E) \otimes E'$ *and* $S \otimes (E \otimes E')$ *are defined and* $(S \otimes E) \otimes E' = S \otimes (E \otimes E')$.
- *a reflexive and transitive refinement relation* $\leq \subseteq \mathfrak{S} \times \mathfrak{S}$ *such that* $S \leq T$ *implies* $\ell(S) = \ell(T)$,
- *an environment correctness relation* $\rightarrow \subseteq \mathfrak{S} \times \mathfrak{S}$ *such that, if* $S \rightarrow E$ *then* $S \otimes E$ *is defined; we write* $S \leftrightarrows E$ *and call* S *and* E *compatible, if* $S \rightarrow E$ *and* $E \rightarrow S$.

For all interfaces $S, S', E, E' \in \mathfrak{S}$ *the following properties must hold:*

A1. *Compositional Refinement:*
If $S \otimes E$ *is defined,* $S' \leq S$ *and* $E' \leq E$*, then* $S' \otimes E'$ *is defined and* $S' \otimes E' \leq S \otimes E$.

A2. *Preservation of Environment Correctness:*
If $S \rightarrow E$ *and* $S' \leq S$*,* $E' \leq E$*, then* $S' \rightarrow E'$.

A3. *Environment Composition:*
If $S \otimes E \rightarrow E'$ *and* $S \otimes E' \rightarrow E$ *and* $\ell(E) \cap \ell(E') = \emptyset$*, then* $S \rightarrow E \otimes E'$.[2]

[1] Commutativity means that for all $S, E \in \mathfrak{S}$, if $S \otimes E$ is defined then $E \otimes S$ is defined and $S \otimes E = E \otimes S$; "=" means set-theoretic equality of elements.

[2] In particular, $S \otimes (E \otimes E')$ must be defined.

A formal notion of an *interface theory* was, to our knowledge, first proposed by de Alfaro and Henzinger in [10]. In their work, an interface theory consists of an interface algebra together with a component algebra thus distinguishing between interface specifications and component implementations. Later, in [11], the authors introduced the term *interface language* which simplifies the approach by considering just interfaces with the requirements that independent implementability and incremental design are supported. Our notion of an interface theory is close to an interface language in the sense of [11]. The differences are the following: (1) We associate a set of labels to each interface. (2) We require that interface composition is commutative and pseudo-associative. (3) Instead of using a binary compatibility predicate to express that two interfaces can work properly together, we introduce a unidirectional environment correctness relation. If it is applied in both directions we obtain compatibility. (4) We require compositional refinement for any composable interfaces and not only for compatible ones. (5) Our notion of environmental composition is a variant of incremental design in [11]. For any finite index set I we consider I-sorted sets $(S_i)_{i\in I}$ (i.e. finite families) of interfaces. We call $(S_i)_{i\in I}$ *composable*, if the single interfaces S_i are pairwise composable and if labels of each S_i are shared with at most one other interface S_j $(j \neq i)$ of the family. Obviously, any subset of a composable set of interfaces is composable.

For non-empty index sets I we extend the binary notion of interface composition to I-sorted sets of composable interfaces by the following inductive definition along the size $|I|$ of I:

- If $|I| = 1$, then $\otimes(S_i)_{i\in I} = S_i$ where $I = \{i\}$.
- If $|I| > 1$ and $(S_i)_{i\in I}$ is composable, then $\otimes(S_i)_{i\in I} \triangleq \otimes(S_i)_{i\in I'} \otimes S_j$ for some subset $I' \subseteq I$ with $|I'| = |I| - 1$ and for S_j with $I \setminus I' = \{j\}$.

$\otimes(S_i)_{i\in I}$ is well-defined, since by commutativity and pseudo-associativity of the binary composition the definition is independent of the choice of I'.

3 A Theory of Component Interfaces with Port Contracts

In this section we show how a theory of component interfaces can be constructed on top of any arbitrary labeled interface theory. Our goal is not to define yet another language for component-based design but to focus on fundamental, abstract properties of component interfaces which refines the concept of an interface theory of Sect. 2 by introducing more structure. In addition to pure interfaces, we require that component interfaces define access points in terms of distinguished ports which are used for the composition of component interfaces. In the remainder of this section we assume given an arbitrary labeled interface theory $(\mathfrak{S}, \mathcal{L}, \ell, \otimes, \leq, \rightarrow)$.

3.1 Port Contracts and Component Interfaces

We follow the idea that a port is an interaction point of a component. To specify the legal interactions on a port often port protocols are used, e.g. [1,16]. The disadvantage of using such port protocols is that they usually mix up assumptions

and guarantees. Mostly it is rather difficult or even not feasible to figure out what are the guarantees of a component at a port and what is assumed from the environment for communication on that port. To overcome this deficiency we propose to use explicit distinguished guarantee and assumption behavior specifications for each port of a component following the principles of assume/guarantee reasoning; cf. e.g. [18]. Hence we consider contracts on ports where assumptions and guarantees are both provided by an interface specification of our underlying interface theory.

Definition 2 (Port Contract). *A* port contract *is a pair* (A, G) *with* $A, G \in \mathfrak{S}$ *such that* $\ell(A) = \ell(G)$ *and* $G \rightarrow A$*, i.e. A is a correct environment for G.*[3] *We write* $\ell(P)$ *for* $\ell(A)$ $(= \ell(G))$ *and call* $\ell(P)$ *the* port labels *of P.*

The condition $G \rightarrow A$ is motivated by the intuition that any port contract should specify the assumptions on the environment in such a way that the guaranteed behavior (shown at this port) works fine in any such environment. Port contracts can be refined following the co/contravariant approach where assumptions can be relaxed in the refinement while guarantees may be strengthened.

Definition 3 (Port Contract Refinement). *A port contract* $P' = (A', G')$ *refines a port contract* $P = (A, G)$*, written* $P' \sqsubseteq P$*, if* $G' \leq G$ *and* $A \leq A'$.[4]

A component interface consists of two parts. First, any component interface has a finite set of ports with associated contracts. Formally, the ports are given by a finitely indexed set of port contracts. Secondly, following the terminology in [27], there is a *frame* specification describing the possible visible behaviors of the full component. The idea is that the frame shows the dependencies of actions on the single ports. We assume that the label sets of the ports are pairwise disjoint and that the label set of the component frame is the disjoint union of the port labels. Moreover, the set of assumed behaviors on each port together with the frame must be composable. This is necessary to guarantee that whenever the assumptions on the ports are met by the environment one can indeed construct the composition of the frame with the environment.

Definition 4 (Component Interface). *A* component interface C *is a pair* $C = ((P_i)_{i \in I}, F)$ *such that* $(P_i)_{i \in I}$ *is a finitely indexed set of port contracts* $P_i = (A_i, G_i)$ *and* $F \in \mathfrak{S}$ *is an interface, called* component frame*, such that the following conditions are satisfied:*

1. *For all* $i, j \in I$ *with* $i \neq j$, $\ell(P_i) \cap \ell(P_j) = \emptyset$.
2. $\ell(F) = \bigcup_{i \in I} \ell(P_i)$.
3. $(A_i)_{i \in I} \cup \{F\}$ *is a composable set of interfaces.*

The set of labels of the component interface C is given by $\ell(C) = \ell(F)$.

[3] In particular, A and G are composable.
[4] Note that $P' \sqsubseteq P$ implies $\ell(P') = \ell(P)$.

3.2 Composition of Component Interfaces

In this section we describe the composition of component interfaces merely based on syntactic considerations. In particular, we do not require yet that guarantees of one component port must satisfy the assumptions of the connected port of the other component.[5] Semantic requirements like this are studied in Sects. 3.4 and 3.5. The composition of two component interfaces C and D is only possible if ports of C can be connected to ports of D in a syntactically meaningful way. The simplest solution would be to require that there is exactly one port of C which can be syntactically matched with exactly one port of D. In that way we would, however, not be able to construct cyclic architectures. Therefore we consider the case in which several binary port connections can be established between two component interfaces (even none). For a binary port connection between two ports, say P^C of C and P^D of D, we assume that P^C and P^D have the same set of labels and that the guarantee interfaces of the two ports are composable.[6] Then C and D can be composed if there is a set of binary connections between ports of the two components such that the non-connected ports of C and D have pairwise disjoint labels and if the two component frames are composable. The non-connected ports become the ports of the composition.

Definition 5 (Component Interface Composition).
Let $C = ((P_i^C)_{i\in I}, F^C)$ and $D = ((P_j^D)_{j\in J}, F^D)$ be component interfaces. C and D are composable *if there exist subsets $I_0 \subseteq I$, $J_0 \subseteq J$ and a bijective connector function $\kappa : I_0 \to J_0$ such that*

1. *for all $i \in I_0$, $P_i^C = (A_i^C, G_i^C)$ and $P_{\kappa(i)}^D = (A_{\kappa(i)}^D, G_{\kappa(i)}^D)$, the pair $G_i^C, G_{\kappa(i)}^D$ is composable and $\ell(P_i^C) = \ell(P_{\kappa(i)}^D)$,*
2. *$\ell(C) \cap \ell(D) = \bigcup_{i\in I_0} \ell(P_i)$,*
3. *F^C and F^D are composable.*

Then the composition of C and D is defined by

$$C \boxtimes D = ((P_i^C)_{i\in(I\setminus I_0)} \cup (P_j^D)_{j\in(J\setminus J_0)}, F^C \otimes F^D).$$

Obviously, $\boxtimes$ is commutative since the underlying composition operator $\otimes$ and the set-theoretic union of (non-connected) ports is commutative. It is also straightforward to prove that the rules (C1) - (C3) of Def. 1 are satisfied for the composition of component interfaces. Moreover, it is easy to see that whenever C and D are composable component interfaces, then $C \boxtimes D$ is a component interface.

3.3 Refinement of Component Interfaces

Our definition of component interface refinement relies on refinement of ports, see Def. 3, which has been inspired by assume/guarantee reasoning and the

[5] Similarly to interface specifications which may be syntactically composable without being semantically compatible.

[6] Since $\ell(P^C) = \ell(P^D)$ one could equivalently require (taking into account the properties of a port contract) that the assumption interfaces are composable.

notions of behavioral subtyping, see e.g. [23]. A component interface C refines another one D if, first, both have the same number of ports which are pairwise refined and, secondly, the frame of C refines the frame of D in accordance with the refinement relation of the underlying interface theory. Hence component behaviors and guarantees are specialized in the refinement while assumptions are relaxed.

Definition 6 (Component Interface Refinement). *Let $C = ((P_i^C)_{i\in I}, F^C)$ and $D = ((P_j^D)_{j\in J}, F^D)$ be two component interfaces. C refines D, written $C \sqsubseteq D$, if there exists a bijection $\rho : I \to J$ such that*

1. *$P_i^C \sqsubseteq P_{\rho(i)}^D$ for all $i \in I$, and*
2. *$F^C \leq F^D$.*

Note that $C \sqsubseteq D$ implies $\ell(C) = \ell(D)$ and that reflexivity and transitivity of $\sqsubseteq$ is inherited from the underlying refinement relation $\leq$ for interfaces.

Next, we show that the property of compositional refinement required by condition (A1) of an interface theory is also valid for the refinement relation between component interfaces.

Theorem 1 (Compositional Refinement for Component Interfaces). *Let C, C', D, and D' be component interfaces such that C and D are composable and $C' \sqsubseteq C$ as well as $D' \sqsubseteq D$ holds. Then C' and D' are composable and $C' \boxtimes D' \sqsubseteq C \boxtimes D$.*

3.4 Correct Component Environments

Finally, in order to obtain an interface theory for component specifications, we need to define a suitable environment correctness relation. The idea is that the communication requirements of a component interface C are satisfied by another component interface D, playing the role of the environment for C, if (1) C and D are composable, and (2) all port connections that can be established between C and D have the property, that each (environment) assumption, say A_i^C of a connected port of C is satisfied by the guarantee $G_{\kappa(i)}^D$ of the corresponding port of D; i.e. $G_{\kappa(i)}^D \leq A_i^C$.

Definition 7. *Let $C = ((P_i^C)_{i\in I}, F^C)$ and $D = ((P_j^D)_{j\in J}, F^D)$ be two component interfaces which are composable according to a bijective connector function $\kappa : I_0 \to J_0$ for subsets $I_0 \subseteq I$ and $J_0 \subseteq J$. D is a* correct environment *for C, denoted by $C \twoheadrightarrow D$, if for all $i \in I_0$, $G_{\kappa(i)}^D \leq A_i^C$.*

Theorem 2 (Preservation of Environment Correctness). *Let C, C', D, and D' be component interfaces such that $C' \sqsubseteq C$ and $D' \sqsubseteq D$. If $C \twoheadrightarrow D$, then also $C' \twoheadrightarrow D'$.*

Corollary 1. *Component interfaces together with their composition, refinement and environment correctness relation form a labeled specification theory with labels $\mathcal{L}$ over any labeled interface theory $(\mathfrak{S}, \mathcal{L}, \ell, \otimes, \leq, \to)$.*

3.5 Reliability of Component Interfaces

Up to know, we have not studied the relation between the frame of a component and the guarantees at the ports of the component. Thus it could be possible that a component C states a guarantee on a port, which is not really supported by the component frame. In such a case the component interface would not be reliable on that port. Indeed the actual behavior of a component is specified by its frame and a user who wants to connect to a certain port is trusting the guarantee on that port which should be established by any component implementation that is a refinement of the component frame. In general, we can still relax this consideration, since we can assume that the component is put into a context where the assumptions on all other ports are met. Consider, for instance, a component interface C and the port $P_1 = (A_1, G_1)$ of C. Then G_1 shows the guarantee of C on port P_1 whenever the component is put in the environment $A_2 \otimes \ldots \otimes A_n$ for the other ports. In other words, the frame F, which specifies the dependencies between the ports, should produce in the context of the environment $A_2 \otimes \ldots \otimes A_n$ a behavior that satisfies the guarantee G_1 on the first port. Formally, this can be expressed by requiring that $A_2 \otimes \ldots \otimes A_n \otimes F$ is a refinement of G_1. Hence A_1 is not used as an assumption for G_1 but only as an assumption for the other guarantees G_j with $j > 1$.[7]

Definition 8 (Reliable Component Interface). *Let $C = ((P_i)_{i \in I}, F)$ be a component interface with port contracts $P_i = (A_i, G_i)$ for all $i \in I$. C is* reliable on a port P_j *($j \in I$), if $\otimes(A_i)_{i \in I \setminus \{j\}} \otimes F \leq G_j$. C is* reliable *if C is reliable on all ports P_i for all $i \in I$.*

The next proposition shows that reliable components can themselves rely on all environments which satisfy the assumptions on each port of the component; i.e. the component frame (as well as all refinements F' of F) "feel well" in each environment made up by the composition of single environments that satisfy the assumptions on each port.

Proposition 1. *Let $C = ((P_i)_{i \in I}, F)$ be a reliable component interface with port contracts $P_i = (A_i, G_i)$ for all $i \in I$. For all $i \in I$, let E_i be interfaces such that $E_i \leq A_i$. Then $F \rightarrow \otimes(E_i)_{i \in I}$.*

An important issue is, of course, to study to what extent reliability of component interfaces is preserved by composition. We can show that this is indeed the case if reliable components are correct environments for each other and if the composition relies on the connection of two ports. If there are more port connections used for the composition, then the ports of each single component (used for the connections) must be independent to achieve this result. Intuitively this means, that the frame allows arbitrary interleaving between the behaviors of those ports. Formally we require that under the assumptions of the other ports

[7] It would be desirable to use all assumptions $A_1 \otimes \ldots \otimes A_n$ for each guarantee G_j. But this can raise serious problems if there are cyclic dependencies between assumptions and guarantees on connected ports.

the frame is a refinement of the product of the behaviors (i.e. guarantees) of the ports under consideration.

Definition 9. *Let $C = ((P_i)_{i\in I}, F)$ be a component interface with port contracts $P_i = (A_i, G_i)$ for all $i \in I$ and let $I_0 \subseteq I$. The ports $(P_i)_{i\in I_0}$ are* independent *w.r.t. F, if*

1. $\otimes(A_i)_{i\in I\setminus I_0} \otimes F \leq \otimes(G_j)_{j\in I_0}$, *and*
2. $\otimes(G_j)_{j\in I_0} \to \otimes(A_j)_{j\in I_0}$.

Of course, any single port is independent and a set of ports is independent if and only if it could be collapsed into a single port.

Theorem 3 (Contract Composition Preserves Reliability). *Let C and D be two reliable and composable component interfaces such that the connected ports on each side are independent (which is trivially satisfied if only two ports are connected). Then $C \leftrightarrows D$ implies that $C \boxtimes D$ is reliable.*

We can further prove that whenever $C \leftrightarrows D$ then the composition of F^C with any environments of non-connected ports of C is compatible with the composition of F^D with any environments of non-connected ports of D.

Lemma 1. *Let $C = ((P_i^C)_{1\leq i\leq m}, F^C)$ and $D = ((P_i^D)_{1\leq i\leq n}, F^D)$ be reliable component specifications which are composable according to a bijective connector function $\kappa : I_0 \to J_0$ for subsets $I_0 \subseteq I$ and $J_0 \subseteq J$. Assume that the ports $(P_i^C)_{i\in I_0}$ and $(P_j^D)_{j\in J_0}$ are independent w.r.t. F^C and F^D, respectively. If $C \leftrightarrows D$ then it holds that*

$$\left(\otimes(A_i^C)_{i\in(I\setminus I_0)} \otimes F^C\right) \leftrightarrows \left(F^D \otimes (\otimes(A_j^D)_{j\in(J\setminus J_0)})\right).$$

4 Component Interfaces with MIOs

As a concrete instance of our approach we will use modal I/O-transition systems (MIOs) for the representation of component frames and for the specification of assumptions and guarantees on ports. Modal transition systems have been introduced in [22] and later extended to MIOs in [21]. We have chosen MIOs as our basic formalism since they allow us to distinguish between transitions which are optional (*may*) or mandatory (*must*) and thus support well loose specifications and refinements. In particular the ability for may-transitions is very useful to specify contracts with loose assumptions. In Sect. 4.1 we construct a labeled modal interface theory on the basis of [5], which will then be used, in Sect. 4.2, to build modal component interfaces along the lines of our abstract framework in Sect. 3.

4.1 Labeled Modal Interface Theory

We assume a global set of (observable) action labels $\mathcal{L}^{act}$ and a distinguished (non-observable) action $\tau \notin \mathcal{L}^{act}$. Each MIO is based on an *I/O-labeling* $L =$

(I_L, O_L) consisting of disjoint sets of *input labels* $I_L \subseteq \mathcal{L}^{act}$ and *output labels* $O_L \subseteq \mathcal{L}^{act}$. A *modal I/O-transition system* $M = (L_M, S_M, s_{0,M}, \dashrightarrow_M, \longrightarrow_M)$ consists of an I/O-labeling $L_M = (I_M, O_M)$, a finite set of *states* S_M, an *initial state* $s_{0,M} \in S_M$, a *may-transition relation* $\dashrightarrow_M \subseteq S_M \times (\bigcup L_M \cup \{\tau\}) \times S_M$, and a *must-transition relation* $\longrightarrow_M \subseteq \dashrightarrow_M$, i.e. any must-transition is also a may-transition. The set of the *reachable states* of M is denoted by $\mathscr{R}(M)$ with $s \in \mathscr{R}(M)$ if, and only if there is a finite sequence of may-transitions from $s_{0,M}$ to s in M.

All facts and definitions that we provide for particular MIOs are independent of the names of the states of the MIO. In fact we will use MIOs as representatives of their isomorphism classes w.r.t. bijections on states and the set of those isomorphism classes is denoted by $\mathfrak{S}^{MIO}$. The labeling function $\ell^{act} : \mathfrak{S}^{MIO} \to \wp_{\text{fin}}(\mathcal{L}^{act})$ is defined by $\ell^{act}(M) = I_L \cup O_L$ for each MIO M with I/O-labeling $L = (I_L, O_L)$.

Figure 1 shows the pictorial representation of MIOs used in the following. The I/O-labeling of a MIO is shown on its frame. Input and output labels are indicated by the names on the incoming and outgoing arrows. On the transitions, input labels are suffixed with "?" and output labels are suffixed with "!". May-transitions are drawn with a dashed arrow; must-transitions with a solid arrow.

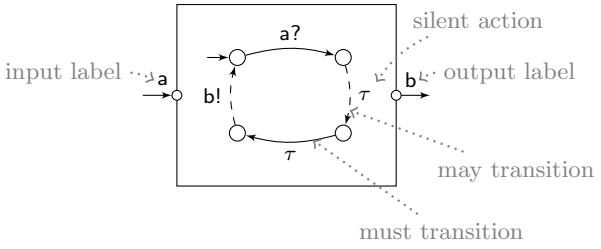


Fig. 1. Modal I/O-transition system

Composable MIOs and Their Synchronous Composition. Two MIOs can be composed if their labels overlap only on complementary types. This means that whenever a label is shared, then the label is either an output label of the first MIO and an input label of the second one or an input label of the first MIO and an output label of the second one. Formally, for two I/O-labelings $K = (I_K, O_K)$ and $L = (I_L, O_L)$, their intersection is denoted by $K \bowtie L = (I_K \cup O_K) \cap (I_L \cup O_L)$. K and L are *composable* if $K \bowtie L = (I_K \cap O_L) \cup (I_L \cap O_K)$. Two MIOs M and N are *composable* if their I/O-labelings are composable. A finitely indexed set $(M_i)_{i \in I}$ of MIOs is composable, if the single interfaces M_i are pairwise composable. Then labels of each M_i can only be shared with at most one other MIO M_j $(j \neq i)$ of the family.

Synchronous composition means that single transitions of two MIOs with shared actions are performed simultaneously. After composition the shared labels become invisible modeled by τ. Formally, the *synchronous composition*

of two composable I/O-labelings K and L removes shared labels from inputs and outputs, i.e., it yields the I/O-labeling $K \otimes^{\text{sy}} L = ((I_K \cup I_L) \setminus (K \bowtie L), (O_K \cup O_L) \setminus (K \bowtie L))$.

The synchronous composition is defined for composable MIOs M and N and denoted (also) by $M \otimes^{\text{sy}} N$. It is defined as the usual product of automata with synchronization on shared labels, which become τ in the product; a synchronization transition in $M \otimes^{\text{sy}} N$ is a must-transition if both synchronizing transitions are must-transitions. If one of the single synchronizing transitions is a proper may-transition, then the synchronization transition is also a proper may-transition. An example for synchronous composition of MIOs will be given in Sect. 4.2 when the frames of the interfaces of a Broker and a Client component are composed. The synchronous composition of MIOs is commutative (since we consider MIOs up to bijections between the sets of states). Also the rules (C1) to (C3) required for a labeled interface theory are true for MIOs.

Weak Modal Refinement. The basic idea of *modal* refinement is that required (*must*) transitions of an abstract specification must also occur in the concrete specification. Conversely, allowed (*may*) transitions of the concrete specification must be allowed by the abstract specification. We will use the weak form of modal refinement introduced by Hüttel and Larsen [17] which supports observational abstraction, i.e., τ-transitions can be dropped and inserted as long as the modalities and the simulation relation are preserved.

M is a *weak modal refinement* of N, written $M \leq^*_{\text{m}} N$, if there exists a weak modal refinement relation R between M and N such that $(s_{0,M}, s_{0,N}) \in R$. Two MIOs M and N are *equivalent*, written $M \approx^*_{\text{m}} N$, if M co-simulates N, i.e. $M \leq^*_{\text{m}} N$ and $N \leq^*_{\text{m}} M$. Weak modal refinement $\leq^*_{\text{m}}$ is reflexive and transitive. If all transitions of the abstract MIO are must-transitions it coincides with weak bisimulation. An example of weak modal refinement is given later in Fig. 4. As a crucial fact, weak modal refinement is preserved by synchronous composition.

Theorem 4 (Compositional Refinement (A1)). *For $i = 1, 2$, let M_i, N_i be MIOs such that $M_i \leq^*_{\text{m}} N_i$ and let M_1 and M_2 be composable. Then M_1 and M_2 are composable and $M_1 \otimes^{\text{sy}} M_2 \leq^*_{\text{m}} N_1 \otimes^{\text{sy}} N_2$.*

Modal Environment Correctness Relation. To discuss correctness of environments we follow the implicit assumption, taken from interface automata [9,11], that outputs are autonomous and must be accepted by a communication partner while inputs are subject to external choice and need not to be served. Hence, output transitions of a MIO express requirements on its environment. For the formal definition, we use one direction of the weak compatibility relation of [5]:

A MIO E is a *modally correct environment* for a MIO M, written $M \rightarrow^*_{\text{m}} E$, if M and E are composable and if for each reachable state $(s,t) \in \mathscr{R}(M \otimes^{\text{sy}} E)$, if M *may* send out in state s a message a shared with E, i.e. if there exists $s \overset{a}{\dashrightarrow}_M s'$ with $a \in O_M \cap I_E$, then E *must* be able to receive the message possibly after a series of internal must-transitions have been performed by E starting from state t, i.e. there exists $t \xrightarrow{\hat{\tau}}_E t'' \overset{a}{\dashrightarrow}_E t'$. The notation $t \xrightarrow{\hat{\tau}}_E t''$ expresses arbitrary many (must) τ-transitions.

Examples of weak modal refinement are given below in Sect. 4.2. Both requirements (A2) and (A3) of labeled interface theories are satisfied by MIOs with weak modal refinement and modal environment correctness.

Theorem 5 (Preservation of Environment Correctness (A2), Environmental Composition (A3)). *Let* $M, M', E, E' \in \mathfrak{S}^{MIO}$.

1. *If* $M \rightarrow^*_{\mathrm{m}} E$ *and* $M' \leq^*_{\mathrm{m}} M$, $E' \leq^*_{\mathrm{m}} E$, *then* $M' \rightarrow^*_{\mathrm{m}} E'$.
2. *If* $M \otimes^{\mathrm{sy}} E \rightarrow^*_{\mathrm{m}} E'$ *and* $M \otimes^{\mathrm{sy}} E' \rightarrow^*_{\mathrm{m}} E$ *and* E, E' *are composable, then* $M \rightarrow^*_{\mathrm{m}} E \otimes^{\mathrm{sy}} E'$.

As a consequence of the definitions and results from above, the MIO framework satisfies the requirements of a labeled interface theory according to Def. 1.

Corollary 2. $(\mathfrak{S}^{MIO}, \mathcal{L}^{act}, \ell^{act}, \otimes^{\mathrm{sy}}, \leq^*_{\mathrm{m}}, \rightarrow^*_{\mathrm{m}})$ *is a labeled interface theory.*

4.2 Modal Component Interfaces

On top of the modal interface theory defined in the last section we construct, along the lines of Sect. 3, a theory of modal component interfaces, cf. Cor. 1, represented by $(\mathcal{C}^{MIO}, \mathcal{L}^{act}, \ell^{act}, \boxtimes^{\mathrm{sy}}, \sqsubseteq^{\mathrm{m}}, \twoheadrightarrow^*_{\mathrm{m}})$.

As an illustration we consider a simple message transmission system which consists of two components: a broker component delivers received messages to a client component. A standard message is immediately delivered while a confidential message is only delivered after successful authentication of the client. The static structure of this component system is shown in Fig. 2. The meaning of the input and output actions is summarized in Table 1.

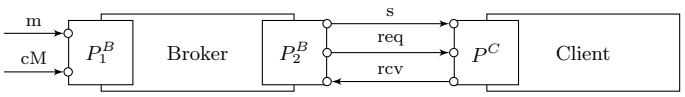


Fig. 2. The static structure of the message transmission system

Component Interface of the Broker Component. We first discuss the component interface of the broker component shown in Figure 3. It has a frame specification F^B and two port contracts $P_1^B = (A_1^B, G_1^B)$ and $P_2^B = (A_2^B, G_2^B)$.

Frame F^B. The frame specification F^B specifies the reaction to the reception of messages on port P_1^B. If a standard message is received ($m?$), the message is delivered immediately to the client ($s!$) via port P_2^B and the broker is again ready to receive new messages. If a confidential message is received ($cM?$), the client is first asked for authentication ($req!$), and only after the reception of the valid authentication information ($rcv?$), the message is delivered ($s!$).

Table 1. Intuitive meaning of actions

Broker B		Client C	
$m?$	receive a message		
$cM?$	receive a confidential message		
$s!$	deliver the message to the client	$s?$	receive the message
$req!$	send out an authentication request	$req?$	receive an authentication request
$rcv?$	receive the (valid) authentication information	$rcv!$	send the authentication information

Port Contracts P_1^B, P_2^B. The assumption A_1^B of the port contract P_1^B allows the environment to generate new messages of any type at any time and the guarantee G_1^B ensures that the broker must always accept standard as well as confidential messages. Obviously, $G_1^B \rightarrow_{\mathrm{m}}^* A_1^B$ since G_1^B does not have any outputs.

The second port contract P_2^B specifies the interaction with the client environment. The assumption A_2^B requires that the client must accept messages and is obliged to answer any authentication request. The guarantee G_2^B expresses that the broker may directly send a message to the client but the broker may also ask the client for authentication before, and then it guarantees to take the authentication response. The port contract P_2^B is valid since $G_2^B \rightarrow_{\mathrm{m}}^* A_2^B$: every possible output of G_2^B must be accepted by any environment satisfying A_2^B.

Component Interface. The interface of the broker component is given by $B = (\{P_1^B, P_2^B\}, F^B)$. Obviously it is well-formed, since the syntactic conditions of Def. 4 are satisfied. In particular, $\{A_1^B, A_2^B, F^B\}$ is a composable set of modal interfaces: every action label occurs in at most two interfaces in this set, with complementary action types (input vs. output). The broker interface is also reliable. According to Def. 8 the proof obligations are $A_1^B \otimes^{\mathrm{sy}} F^B \leq_{\mathrm{m}}^* G_2^B$ and $A_2^B \otimes^{\mathrm{sy}} F^B \leq_{\mathrm{m}}^* G_1^B$. They are detailed in Fig. 4 and the weak modal refinement relations can be discharged, for instance, with the MIO Workbench [5].

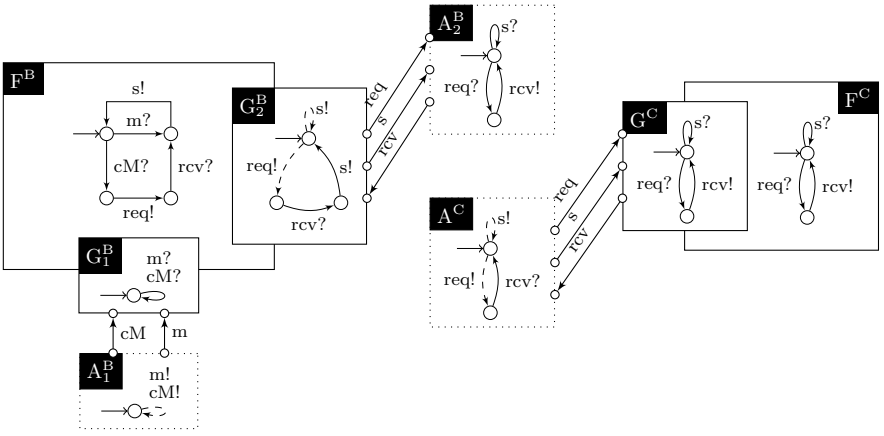
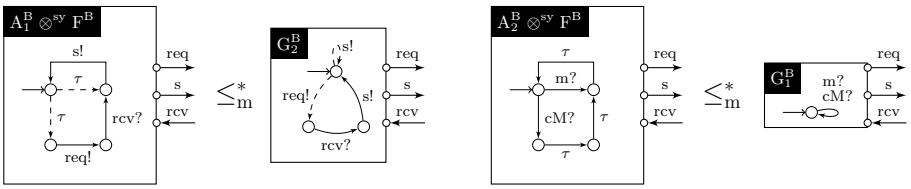


Fig. 3. Component interfaces of the broker and client component



(a) Proof obligation (I) (b) Proof obligation (II)

Fig. 4. Proof obligations for reliability of the component interface C^B

Component Interface of the Client Component. The interface C of the client component is much simpler; cf. Fig. 3. There is only one port such that the guarantee of the port coincides with the frame which immediately implies reliability The specifications are self-explanatory. Just note, that the assumptions A^C require that the environment must receive any answer to an authentication request; hence $G^C \rightarrow^*_m A^C$.

Composing Broker and Client Interfaces. Clearly, the two component interfaces B and C are (syntactically) composable by connecting the ports P_2^B and P_1^C, which both have the same labels and composable guarantees (and assumptions). The component interface $B \boxtimes^{sy} C$ resulting from the composition is shown in Fig. 5. Due to Thm. 3 the interface $B \boxtimes^{sy} C$ is reliable since the single interfaces are reliable and since they are correct environments for each other, i.e. $B \leftrightarrows^*_m C$. For the latter the proof obligations are $G^C \leq^*_m A_2^B$ (which is trivially valid) and $G_2^B \leq^*_m A^C$ which can be discharged, for instance, with the MIO Workbench.

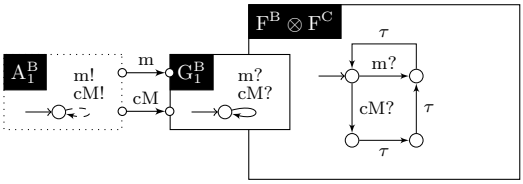


Fig. 5. Composition of broker and client component interfaces

5 Conclusion

We have presented an abstract framework how to construct a theory of component interfaces with port contracts on top of a given interface theory and we have instantiated this approach to obtain modal component interfaces on the basis of modal I/O-transition systems whose modalities are particularly useful for describing loose assumptions. In future work we plan to study other instantiations of our abstract component theory, on the one hand on the basis of other

formalisms for interface specifications like language-based ones or Petri nets, on the other hand by playing with other environment correctness notions and integrating data states with invariants and pre/postconditions on transitions. Also the MIO Workbench [5], which can actually be used to verify weak modal refinement and environment correctness, should be extended to explicitly support components with contracts on ports. Another issue concerns the applicability of our approach to well established design-languages like Wright [1] or UML which also rely on port-based communication but which include further structure like explicit connectors as in Wright or ports consisting of provided and required interfaces as in UML.

Acknowledgement. We would like to thank the reviewers of the submitted version of this paper for their useful hints and remarks.

References

1. Allen, R., Garlan, D.: A formal basis for architectural connection. ACM Trans. Softw. Eng. Methodol. 6(3), 213–249 (1997)
2. Back, R.-J., von Wright, J.: Refinement calculus - a systematic introduction. Undergraduate texts in computer science. Springer (1999)
3. Basu, A., Bensalem, S., Bozga, M., Combaz, J., Jaber, M., Nguyen, T.-H., Sifakis, J.: Rigorous component-based system design using the bip framework. IEEE Software 28(3), 41–48 (2011)
4. Bauer, S.S., David, A., Hennicker, R., Larsen, K.G., Legay, A., Nyman, U., Wąsowski, A.: Moving from Specifications to Contracts in Component-Based Design. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 43–58. Springer, Heidelberg (2012)
5. Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On Weak Modal Compatibility, Refinement, and the MIO Workbench. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 175–189. Springer, Heidelberg (2010)
6. Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C.: Multiple Viewpoint Contract-Based Specification and Design. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2007. LNCS, vol. 5382, pp. 200–225. Springer, Heidelberg (2008)
7. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint markov chains. Theor. Comput. Sci. 412(34), 4373–4404 (2011)
8. Cau, A., Collette, P.: Parallel composition of assumption-commitment specifications: A unifying approach for shared variable and distributed message passing concurrency. Acta Inf. 33(2), 153–176 (1996)
9. de Alfaro, L., Henzinger, T.A.: Interface automata. Software Engineering Notes, 109–120 (2001)
10. de Alfaro, L., Henzinger, T.A.: Interface Theories for Component-Based Design. In: Henzinger, T.A., Kirsch, C.M. (eds.) EMSOFT 2001. LNCS, vol. 2211, pp. 148–165. Springer, Heidelberg (2001)
11. de Alfaro, L., Henzinger, T.A.: Interface-based Design. In: Broy, M., Grünbauer, J., Harel, D., Hoare, C.A.R. (eds.) Engineering Theories of Software-intensive Systems. NATO Science Series: Mathematics, Physics, and Chemistry, vol. 195, pp. 83–104. Springer (2005)

12. de Alfaro, L., Henzinger, T.A., Stoelinga, M.I.A.: Timed Interfaces. In: Sangiovanni-Vincentelli, A.L., Sifakis, J. (eds.) EMSOFT 2002. LNCS, vol. 2491, pp. 108–122. Springer, Heidelberg (2002)
13. Dijkstra, E.W.: Guarded Commands, Non-determinancy and A Calculus for the Derivation of Programs. In: Bauer, F.L., Samelson, K. (eds.) Language Hierarchies and Interfaces. LNCS, vol. 46, pp. 111–124. Springer, Heidelberg (1976)
14. Emmi, M., Giannakopoulou, D., Păsăreanu, C.S.: Assume-Guarantee Verification for Interface Automata. In: Cuellar, J., Maibaum, T., Sere, K. (eds.) FM 2008. LNCS, vol. 5014, pp. 116–131. Springer, Heidelberg (2008)
15. Goessler, G., Raclet, J.-B.: Modal contracts for component-based design. In: SEFM, pp. 295–303. IEEE Computer Society (2009)
16. Hennicker, R., Janisch, S., Knapp, A.: On the observable behaviour of composite components. Electr. Notes Theor. Comput. Sci. 260, 125–153 (2010)
17. Hüttel, H., Larsen, K.G.: The Use of Static Constructs in A Modal Process Logic. In: Meyer, A.R., Taitslin, M.A. (eds.) Logic at Botik 1989. LNCS, vol. 363, pp. 163–180. Springer, Heidelberg (1989)
18. Jones, C.B.: Development methods for computer programs including a notion of interference. PhD thesis, Oxford University Computing Laboratory (1981)
19. Lamport, L.: *win* and *sin*: Predicate transformers for concurrency. ACM Trans. Program. Lang. Syst. 12(3), 396–428 (1990)
20. Larsen, K.G.: Modal Specifications. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 232–246. Springer, Heidelberg (1990)
21. Larsen, K.G., Nyman, U., Wąsowski, A.: Modal I/O Automata for Interface and Product Line Theories. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 64–79. Springer, Heidelberg (2007)
22. Larsen, K.G., Thomsen, B.: A Modal Process Logic. In: 3rd Annual Symp. Logic in Computer Science, LICS 1988, pp. 203–210. IEEE Computer Society (1988)
23. Liskov, B., Wing, J.M.: A behavioral notion of subtyping. ACM Trans. Program. Lang. Syst. 16(6), 1811–1841 (1994)
24. Meyer, B.: Applying "design by contract". IEEE Computer 25(10), 40–51 (1992)
25. Misra, J., Mani Chandy, K.: Proofs of networks of processes. IEEE Trans. Software Eng. 7(4), 417–426 (1981)
26. Parizek, P., Plasil, F.: Modeling environment for component model checking from hierarchical architecture. Electr. Notes Theor. Comput. Sci. 182, 139–153 (2007)
27. Plasil, F., Visnovsky, S.: Behavior protocols for software components. IEEE Trans. Software Eng. 28(11), 1056–1076 (2002)
28. Quinton, S., Graf, S.: Contract-based verification of hierarchical systems of components. In: SEFM, pp. 377–381. IEEE Computer Society (2008)
29. Raclet, J.-B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. Inform. 108(1-2), 119–149 (2011)
30. Xu, Q., Cau, A., Collette, P.: On Unifying Assumption-commitment Style Proof Rules for Concurrency. In: Jonsson, B., Parrow, J. (eds.) CONCUR 1994. LNCS, vol. 836, pp. 267–282. Springer, Heidelberg (1994)