A Meta-Theory for Component Interfaces with Contracts on Ports[☆]

Sebastian Bauer[a], Rolf Hennicker[a], Axel Legay[b]

[a]*Ludwig-Maximilians-Universität München, Germany*
[b]*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 theory satisfies itself the general laws of an interface theory for composition, refinement and communication compatibility. 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. We study reliable component interfaces and we provide methodological guidelines how to design reliable interfaces and how to adapt them to changing environments. Two instances of our approach are presented. First, 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 notions of refinement and compatibility. The second instance uses I/O-predicates as interface specifications.

Keywords: Reactive components, interface specifications, contracts, refinement, compatibility, modal I/O-transition systems, I/O-predicates.

1. Introduction

The development of large, reliable component systems relies heavily on the use of interfaces. According to the state of practice, interfaces are typically described using Word/Excel text documents or modeling languages such as UML/XML which are known to be ambiguous. Hence, rigorous development methods are mandatory which support interface composition, stepwise refinement and the consideration of compatibility issues when components interact. These requirements together with concise rules specifying 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 [17]. There the notion of an *interface theory* has been introduced which consists of an interface algebra together with a component algebra.

[☆]This work has been partially sponsored by the EU project ASCENS, 257414.
Email addresses: `bauerse@pst.ifi.lmu.de` (Sebastian Bauer), `hennicke@pst.ifi.lmu.de` (Rolf Hennicker), `axel.legay@inria.fr` (Axel Legay)

In this paper we follow the spirit 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 by early ideas on manual proof methods of Misra, Chandy [36] and Jones [28], along with the wide acceptance of the pre-/postcondition style of specification in programming [35, 42], and more general semantical rules independent from language representation [14].

In [5], 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 [5] how the contract theory can be instantiated by using modal transitions systems [40] 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 [9] which is needed in many examples. Also compatibility predicates concerning the communication between components have not been integrated in [5]. Moreover, 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). In Sect. 3, a concrete instance of a labeled interface theory is provided in terms of modal I/O-transition systems [31] with weak modal refinement according to [27] and weak modal compatibility in the sense of [9].

We show, in Sect. 4, how a theory of component interfaces can be defined on top of any framework satisfying our abstract rules of a labeled 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 and guarantees for the environment that is going to be connected on this particular port. All notions of an interface theory, i.e. composition, refinement and compatibility, 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 if a single connection between component interfaces is established. We also study multiple connections and provide sufficient conditions to ensure reliability preservation in this case. As a proof of concept, we instantiate in Sect. 5 our generic constructions and build a *modal* theory of component interfaces on top of the labeled interface theory with modal I/O-transitions systems (MIOs). We consider a small case study concerning a message transmission system. In Sect. 6 we propose general methods for the design and adaptation of reliable component interfaces and illustrate them with the MIO examples. In Sect. 7 we discuss another instantiation of our approach using predicates over input/output variables as interface specifications. We finish with some concluding remarks in Sect. 8.

This paper is a significantly restructured and extended revision of the conference paper [7]. The new approach relies on a symmetric compatibility notion (inherited from the underlying interface theory) instead of a unidirectional environment correctness notion. This simplifies the approach and makes it better accessible. Much emphasis has been put on the notion of reliability of component interfaces, a notion which is now motivated in detail and illustrated by examples. An additional section has been included that presents elaborated methods to design and to adapt component interfaces. They are illustrated by our case study. All theorems and facts are justified by detailed proofs.

Related Work. As observed above, our work extends classical interface theories [17, 19, 13] with an explicit treatment of assumptions-guarantees. Other works on interface automata, e.g. [21], exploit the concept of assumption and guarantee to improve the efficiency of compatibility checking. Those works, which build on former approaches developed for concurrent transition systems [15, 22, 29], 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 [37] 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 in [11, 10], a similar approach to the one of [37] 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 [35], that is based on pre- and postconditions as state predicates and invariants for the system itself. This approach, which builds on seminal ideas proposed by Dijkstra and Lamport [20, 30], is similar to ours in the sense that pre- and postconditions shall be viewed as assumption and guarantee, respectively. In [33] this idea has been further developed by specifying pre/postconditions not only for the provided methods of a component but also for the required ones such that correctness of matching can be checked when components are composed.

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.[12] proposed a contract theory where assumptions 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 [39, 23] leverage this problem but ignore the multiple treatment of assumptions and guarantees. None of the approaches proposes a methodology for designing and adapting reliable components.

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. A formal notion of an abstract interface theory was, to our knowledge, first proposed by de Alfaro and Henzinger in [17]. In their work, an interface theory consists of an interface algebra together with a component algebra to distinguish between interface specifications and component implementations. Later, in [18], the authors introduced the term *interface language* which simplifies the approach by considering just interfaces requiring independent implementability and incremental design. Our notion of an interface theory is close to an interface language in the sense of [18].

We assume given a class $\mathfrak{S}$ of interface specifications and 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. An interface theory must offer a refinement relation $\leq$ to relate "concrete" and "abstract" specifications, such that $S \leq T$ means that S is a correct refinement of T. Refinement must be compositional, i.e. it must be preserved by interface composition which is expressed by requirement (I1) below. An interface theory must also address the relationship between communicating components. For this purpose the binary compatibility predicate $\leftrightarrows$ is introduced, such that $S \leftrightarrows T$ means that there is no communication error when S and T interact. The compatibility predicate 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 compatibility must be preserved by refinement as stated in requirement (I2) below.

Definition 1 (Interface Theory). An *interface theory* is a quadruple $(\mathfrak{S}, \otimes, \leq, \leftrightarrows)$ consisting of

- a set $\mathfrak{S}$ of interface specifications,
- a partial, commutative[1] and associative[2] composition operator $\otimes : \mathfrak{S} \times \mathfrak{S} \to \mathfrak{S}$; we call S and T *composable*, if $S \otimes T$ is defined,
- a reflexive and transitive refinement relation $\leq \subseteq \mathfrak{S} \times \mathfrak{S}$; we call S and T *equivalent*, written $S \approx T$, if $S \leq T$ and $T \leq S$,
- a symmetric compatibility predicate $\leftrightarrows \subseteq \mathfrak{S} \times \mathfrak{S}$ such that $S \leftrightarrows T$ implies $S \otimes T$ defined.

For all interfaces $S, S', T, T' \in \mathfrak{S}$ the following properties are required:

I1. *Compositional Refinement:* If $S \otimes T$ is defined, $S' \leq S$ and $T' \leq T$, then $S' \otimes T'$ is defined and $S' \otimes T' \leq S \otimes T$.

I2. *Preservation of Compatibility:* If $S \leftrightarrows T$, $S' \leq S$ and $T' \leq T$, then $S' \leftrightarrows T'$.

[1] Commutativity means that for all $S, T \in \mathfrak{S}$, if $S \otimes T$ is defined then $T \otimes S$ is defined and $S \otimes T = T \otimes S$ are set-theoretically equal.

[2] Associativity means that for all $S, T, R \in \mathfrak{S}$, if S, T and R are pairwise composable then $(S \otimes T) \otimes R$ and $S \otimes (T \otimes R)$ are defined and $(S \otimes T) \otimes R = S \otimes (T \otimes R)$.

Interface theories provide a nice abstract framework focusing on rudimentary requirements for component-based design. But there is a lack of structure to express explicit, distinguished interaction points when components are composed. To cope with this issue we propose a simple extension of interface theories such that any interface is equipped with a set of labels. It turns out that this simple supplement is sufficient to develop a full, generic theory of component interfaces with contracts on ports. Intuitively labels determine the visible actions supported by an interface. Some straightforward properties for the labeling of interfaces are required by conditions (L1) - (L3). (L1) follows the idea that communication happens via shared labels which are not available anymore after composition; thus we rely on binary communication. (L2) expresses that interfaces without shared labels can always be composed (in practice, this would lead to arbitrary interleaving). (L3) states that abstract and concrete interfaces have the same labels.[3]

Definition 2 (Labeled Interface Theory). A *labeled interface theory* is a tuple $(\mathfrak{S}, \otimes, \leq, \leftrightarrows, \mathcal{L}, \ell)$ such that $(\mathfrak{S}, \otimes, \leq, \leftrightarrows)$ is an interface theory extended by

- a set $\mathcal{L}$ of labels,
- a function $\ell : \mathfrak{S} \to \wp_{\mathrm{fin}}(\mathcal{L})$ assigning a finite set of labels to each interface.

For all interfaces $S, T, R \in \mathfrak{S}$ the following properties are required:

L1. If $S \otimes T$ is defined, then $\ell(S \otimes T) = (\ell(S) \cup \ell(T)) \setminus (\ell(S) \cap \ell(T))$.

L2. If $\ell(S) \cap \ell(T) = \emptyset$, then $S \otimes T$ is defined.

L3. If $S \leq T$, then $\ell(S) = \ell(T)$.

For any finite, non-empty 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 are pairwise composable and if each label of each S_i is shared with at most one other interface S_j ($j \neq i$). Obviously, any subset of a composable set of interfaces is composable. 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 $|I| = 1$ and $I = \{i\}$, then $\otimes(S_i)_{i \in I} = S_i$.
- 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 $\{j\} = I \setminus I'$.

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

[3] At the cost of a more technical development, this condition could be relaxed by allowing more labels for refinements.

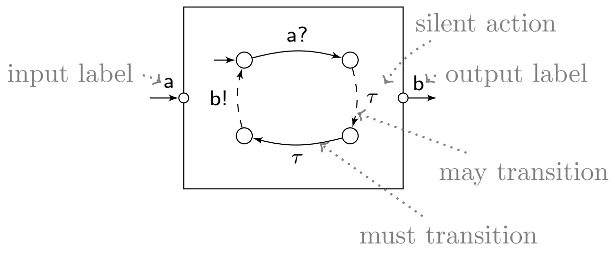


Figure 1: Modal I/O-transition system

3. Labeled Modal Interface Theory

As a concrete instance of our approach we will use modal I/O-transition systems (MIOs). Modal transition systems (MTS) have been introduced in [32] and later extended by Input/Output alphabets in [31]. In [9] we have introduced an interface theory for MIOs with a novel weak compatibility predicate and a weak refinement relation for MIOs (on the basis of weak refinement for MTS in [27]). As a verification tool we use the MIO Workbench presented first in [9]. 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 very well loose specifications and refinements. In particular, the ability of may-transitions will be very useful to specify contracts with loose assumptions later on.

In this section we recall the interface theory of MIOs using the notions of [9] and we extend them by introducing labels explicitly. Compared to [9] we will incorporate a minor syntactic modification, since we will not use explicit names for internal (silent) actions but represent them uniformly by τ.

We assume a global set of (observable) action labels $\mathcal{L}^{act}$ and a distinguished (non-observable) action $\tau \notin \mathcal{L}^{act}$. An *I/O-labeling* $L = (I, O)$ consists of disjoint sets of *input labels* $I \subseteq \mathcal{L}^{act}$ and *output labels* $O \subseteq \mathcal{L}^{act}$. 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* $\dashrightarrow_M \subseteq S_M \times (I_M \cup O_M \cup \{\tau\}) \times S_M$, 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* from the initial state $s_{0,M}$ of M w.r.t. may-transitions is denoted by $\mathscr{R}(M)$. For $l \in (I_M \cup O_M \cup \{\tau\})$, we write $s \overset{l}{\dashrightarrow}_M s'$ for $(s, l, s') \in \dashrightarrow_M$ and $s \xrightarrow{l}_M s'$ for $(s, l, s') \in \longrightarrow_M$. Since $\longrightarrow_M \subseteq \dashrightarrow_M$, $s \xrightarrow{l}_M s'$ implies $s \overset{l}{\dashrightarrow}_M s'$.

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.

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 $\mathfrak{S}^{MIO}$.[4] It is straightforward to extend MIOs by a labeling function $\ell^{act} : \mathfrak{S}^{MIO} \to \wp_{\text{fin}}(\mathcal{L}^{act})$ defined by $\ell^{act}(M) = I_L \cup O_L$ for each MIO M with I/O-labeling $L = (I_L, O_L)$.

3.1. Composable MIOs and Their Synchronous Composition

Two MIOs M, N with labelings $L_M = (I_M, O_M)$ and $L_N = (I_N, O_N)$ resp. are composable, if their labels overlap only on complementary types, i.e. $\ell^{act}(M) \cap \ell^{act}(N) = (I_M \cap O_N) \cup (I_N \cap O_M)$. Hence, whenever a label is shared, then it is either an input label of the first MIO and an output label of the second or conversely. A finitely indexed set $(M_i)_{i \in I}$ of MIOs is composable, if the single MIOs 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.

The synchronous composition of two composable MIOs is defined as the usual product of transition systems such that transitions with shared actions are performed (only) simultaneously. After composition the shared labels become invisible modeled by τ. A synchronization transition in the composition is a must-transition only if both of the single transitions are must-transitions. Formally, the *synchronous composition* of two composable MIOs M and N with labelings $L_M = (I_M, O_M)$ and $L_N = (I_N, O_N)$ resp. is given by the MIO

$$M \otimes^{\text{sy}} N = (L, S_M \times S_N, (s_{0,M}, s_{0,N}), \dashrightarrow, \longrightarrow)$$

such that $L = ((I_M \cup I_N) \setminus (\ell^{act}(M) \cap \ell^{act}(N)), (O_M \cup O_N) \setminus (\ell^{act}(M) \cap \ell^{act}(N)))$ and the transition relations are the smallest relations satisfying:

- for all $a \in \ell^{act}(M) \cap \ell^{act}(N)$,
  - if $s \overset{a}{\dashrightarrow}_M s'$ and $t \overset{a}{\dashrightarrow}_N t'$, then $(s,t) \overset{\tau}{\dashrightarrow} (s',t')$,
  - if $s \xrightarrow{a}_M s'$ and $t \xrightarrow{a}_N t'$, then $(s,t) \xrightarrow{\tau} (s',t')$,
- for all $a \in (\ell^{act}(M) \setminus \ell^{act}(N)) \cup \{\tau\}$,
  - if $s \overset{a}{\dashrightarrow}_M s'$, then $(s,t) \overset{a}{\dashrightarrow} (s',t)$ for all $t \in S_N$,
  - if $s \xrightarrow{a}_M s'$, then $(s,t) \xrightarrow{a} (s',t)$ for all $t \in S_N$,
- for all $a \in (\ell^{act}(N) \setminus \ell^{act}(M)) \cup \{\tau\}$,
  - if $t \overset{a}{\dashrightarrow}_N t'$, then $(s,t) \overset{a}{\dashrightarrow} (s,t')$ for all $s \in S_M$,
  - if $t \xrightarrow{a}_N t'$, then $(s,t) \xrightarrow{a} (s,t')$ for all $s \in S_M$.

[4]By considering isomorphism classes we get commutativity of MIO composition as required for an interface theory.

An example for synchronous composition of MIOs will be given in Sect. 5 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. The next lemma shows that synchronous composition is pseudo-associative and that the conditions (L1) and (L2) required for labeled interface theories are satisfied by MIOs.

Lemma 1. *Let M, N and R be MIOs.*

1. *If M, N and R are pairwise composable, then $(M \otimes^{\mathrm{sy}} N) \otimes^{\mathrm{sy}} R$ and $M \otimes^{\mathrm{sy}} (N \otimes^{\mathrm{sy}} R)$ are defined and $(M \otimes^{\mathrm{sy}} N) \otimes^{\mathrm{sy}} R = M \otimes^{\mathrm{sy}} (N \otimes^{\mathrm{sy}} R)$.*
2. *If $M \otimes^{\mathrm{sy}} N$ is defined, then $\ell^{act}(M \otimes^{\mathrm{sy}} N) = (\ell^{act}(M) \cup \ell^{act}(N)) \setminus (\ell^{act}(M) \cap \ell^{act}(N))$.*
3. *If $\ell^{act}(M) \cap \ell^{act}(N) = \emptyset$, then $M \otimes^{\mathrm{sy}} N$ is defined.*

Proof. 1.: If M, N and R are pairwise composable, then the notion of composability implies that $M \otimes^{\mathrm{sy}} N$ and R are composable and that M and $N \otimes^{\mathrm{sy}} R$ are composable. Then it is straightforward to prove that $(M \otimes^{\mathrm{sy}} N) \otimes^{\mathrm{sy}} R = M \otimes^{\mathrm{sy}} (N \otimes^{\mathrm{sy}} R)$ by taking into account that we consider MIOs up to bijections between states.
2. is obvious according to the construction of the labeling of composed MIOs.
3. follows from the definition of composability. □

3.2. 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 [27] which supports observational abstraction, i.e., silent transitions can be dropped and inserted as long as the modalities and the simulation relation are preserved.

For denoting sequences of transitions that abstract from silent transitions, we use the following notation. Let M be a MIO with I/O-labeling $L_M = (I_M, O_M)$. We write $s \overset{\widehat{\tau}}{\dashrightarrow}_M s'$ if there is a (possibly empty) sequence of may-transitions from s to s' all labeled by τ, and likewise for must-transitions. For $l \in (I_M \cup O_M)$, we write $s \overset{\widehat{l}}{\dashrightarrow}_M s'$ for $s \overset{\widehat{\tau}}{\dashrightarrow}_M r \overset{l}{\dashrightarrow}_M t \overset{\widehat{\tau}}{\dashrightarrow}_M s'$, and likewise for must-transitions.

Let M and N be MIOs with the same I/O-labeling. A relation $R \subseteq S_M \times S_N$ is a *weak modal refinement relation* between M and N if for all $(s_M, s_N) \in R$ and for all $l \in \ell^{act}(M) = \ell^{act}(N)$ the following holds:

1. $s_N \xrightarrow{l}_N s'_N \Rightarrow \exists s'_M \in S_M \,.\, s_M \xrightarrow{\widehat{l}}_M s'_M \wedge (s'_M, s'_N) \in R$.
2. $s_N \xrightarrow{\tau}_A s'_N \Rightarrow \exists s'_M \in S_M \,.\, s_M \xrightarrow{\widehat{\tau}}_M s'_M \wedge (s'_M, s'_N) \in R$.
3. $s_M \overset{l}{\dashrightarrow}_M s'_M \Rightarrow \exists s'_N \in S_N \,.\, s_N \overset{\widehat{l}}{\dashrightarrow}_N s'_N \wedge (s'_M, s'_N) \in R$.
4. $s_M \overset{\tau}{\dashrightarrow}_M s'_M \Rightarrow \exists s'_N \in S_N \,.\, s_N \overset{\widehat{\tau}}{\dashrightarrow}_N s'_N \wedge (s'_M, s'_N) \in R$.

Remember that any must-transition is also a may-transition. Hence, by rules 3. and 4., must-transitions in M must be allowed by corresponding may-transitions in N.

M is a *weak modal refinement* of N, written $M \leq^*_{\mathrm{m}} N$, if there exists a weak modal refinement relation R between M and N such that $(s_{0,M}, s_{0,N}) \in R$. If all transitions of M and N are must-transitions weak refinement coincides with weak bisimulation. Obviously, weak modal refinement is reflexive and transitive. Two MIOs M and N are *equivalent*, written $M \approx^*_{\mathrm{m}} N$, if $M \leq^*_{\mathrm{m}} N$ and $N \leq^*_{\mathrm{m}} M$, i.e. M co-simulates N. Examples of weak modal refinements are given later in the context of component interfaces.

3.3. Weak modal compatibility

For communication compatibility of MIOs we follow the implicit assumption, taken from interface automata [16, 18], 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. We say that two MIOs M and N are *weakly modally compatible*, denoted by $M \rightleftarrows^{\mathrm{m}}_{\mathrm{w}} N$, if they are composable and in each reachable state of the composition, if M may send out an output directed to N then N must accept the corresponding input possibly after a delay caused by silent must-transitions of N and the same must hold in the other direction. Formally this means, see [9], that for each reachable state $(s, t) \in \mathscr{R}(M \otimes^{\mathrm{sy}} N)$, if there exists $s \overset{a}{\dashrightarrow}_M s'$ with $a \in O_M \cap I_N$, then there exists $t \xrightarrow{\hat{\tau}}_N t'' \xrightarrow{a}_N t'$ and the symmetric condition must hold for outputs of N. Examples of weak modal compatibility are given below in Sect. 5.

Now we have all notions provided to present the modal labeled interface theory.

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

Proof. From the results in [9] it follows that $(\mathfrak{S}^{MIO}, \otimes^{\mathrm{sy}}, \leq^*_{\mathrm{m}}, \rightleftarrows^{\mathrm{m}}_{\mathrm{w}})$ is an interface theory in the sense of Def. 2. In particular, weak modal refinement is compositional w.r.t. synchronous composition of MIOs and weak modal compatibility is preserved by weak modal refinement. Moreover, by Lemma 1, the conditions (L1) and (L2) for labeled interface theories are satisfied and condition (L3) is trivially satisfied by definition.

4. 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 refining the ideas of an interface theory by introducing more structure. In addition to pure interfaces, we require that component interfaces define access points in terms of distinguished ports that are used for connecting components. In the remainder of this section we assume given an arbitrary labeled interface theory $(\mathfrak{S}, \otimes, \leq, \leftrightarrows, \mathcal{L}, \ell)$.

4.1. 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, 25]. The disadvantage of using such port protocols is that it is often not feasible to figure out precisely 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 explicit distinguished guarantee and assumption specifications for each port of a component following the principles of assume/guarantee reasoning; cf. e.g. [28]. Hence we consider *contracts on ports* where both assumptions and guarantees are interface specifications of our underlying interface theory. We require that guarantees and assumptions on a port are compatible to ensure correct communication between component implementations and component environments.

Definition 3 (Port Contract). A *port contract* is a pair (A, G) of composable interface specifications $A, G \in \mathfrak{S}$ such that $\ell(A) = \ell(G)$ and $G \leftrightarrows A$. We write $\ell(P)$ for $\ell(G)$ $(= \ell(A))$ and call $\ell(P)$ the *port labels* of P.

Port contracts can be refined following the co/contravariant approach where assumptions can be relaxed in the refinement while guarantees may be strengthened.

Definition 4 (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 $A \leq A'$.

A component interface consists of two parts. First, any component interface has a finite set of ports with associated contracts. Secondly, following the terminology in [38], 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 require 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 frame must be composable with assumptions on ports. 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. These conditions are of purely syntactic nature. Semantic constraints will be considered in Def. 9.

Definition 5 (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 specification, 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.

Example 1. Fig. 2 shows the interface of a *Broker* component specified in terms of MIOs. The broker is always able to receive standard messages $m?$ or confidential messages $cM?$ on its first port P^B_1. A standard message is immediately delivered (output $s!$) on the second port P^B_2 while a confidential message is only delivered after successful authentication. For that purpose the broker sends an authentication request *req*! (on port P_2^B) and is then ready to receive the authentication *rcv*? after which the message will be sent with output *s*!. Since the interface only assumes that a message or a confidential message *may* be sent by the environment to the first port (assumption A_1^B), it can only guarantee that the message or an authentication request *may* be issued on the second port (guarantee G_2^B). But if an authentication has been received with *rcv*?, G_2^B guarantees that the message will be sent with *s*!. Obviously the assumption and guarantee on the first port P_1^B are weakly modally compatible. Also the assumption and guarantee on on P_2^B are weakly modally compatible. In Sect. 6.1, Ex. 3 we will discuss how the *Broker* component interface is constructed in a systematic way.

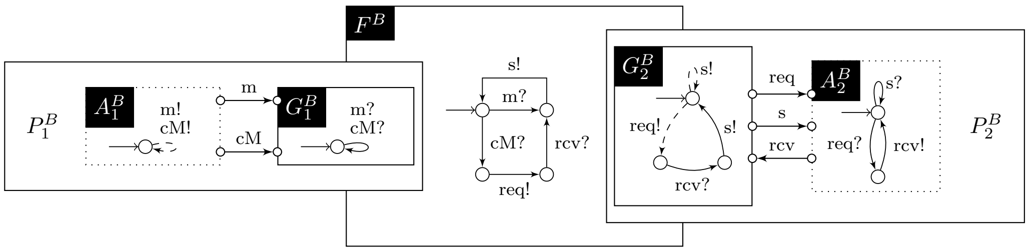


Figure 2: Interface of a Broker component

4.2. 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. 4.4 and 4.5. The composition of two component interfaces B and C is only possible if ports of B can be connected to ports of C in a syntactically meaningful way. The simplest solution would be to require that there is exactly one port of B which can be syntactically matched with exactly one port of C. 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. For a binary port connection between two ports, say P^B of B and P^C of C, we assume that P^B and P^C have the same set of labels and that the guarantee interfaces (assume interfaces resp.) of the two ports are composable. Then B and C can be composed if there is a set of binary connections between ports of the two components such that the non-connected ports of B and C have pairwise disjoint labels and if the two component frames are composable. The non-connected ports become the ports of the composition. Fig. 3 illustrates how the composition works if there are two connected ports and two open ports. In order to stay as abstract as possible, we do not introduce an explicit notion of connector here as in architectural languages like Wright [1].

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

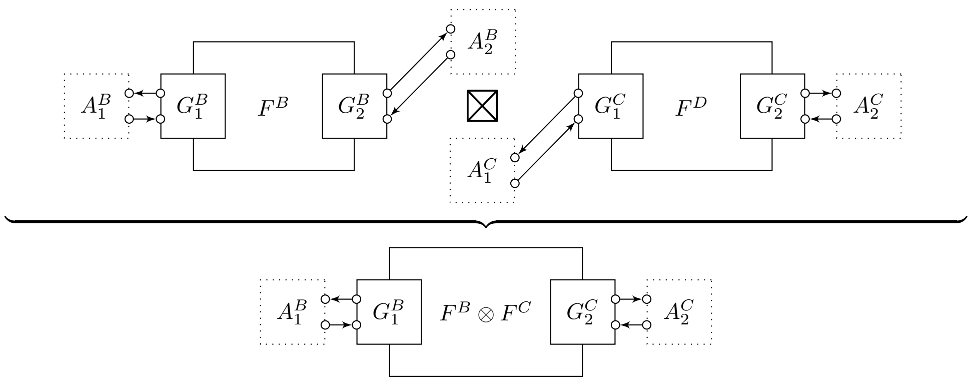


Figure 3: Composition of component interfaces

Definition 6 (Component Interface Composition). Let $B = ((P_i^B)_{i\in I}, F^B)$ and $C = ((P_j^C)_{j\in J}, F^C)$ be component interfaces. B and C are *composable* if there exist non-empty 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 $i \in I_0$, $\ell(P_i^B) = \ell(P_{\kappa(i)}^C)$ and for $P_i^B = (A_i^B, G_i^B)$, $P_{\kappa(i)}^C = (A_{\kappa(i)}^C, G_{\kappa(i)}^C)$, G_i^B is composable with $G_{\kappa(i)}^C$ and A_i^B is composable with $A_{\kappa(i)}^C$,
2. $\ell(F^B) \cap \ell(F^C) = \bigcup_{i\in I_0} \ell(P_i^B) (= \bigcup_{j\in J_0} \ell(P_j^C))$,
3. F^B and F^C are composable.

Then the composition of B and C is defined by

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

According to condition 1. - 3. of the definition, the bijective connector function κ is unique and therefore the notation $\boxtimes$ needs no qualification concerning κ.

Lemma 3. *If B and C are composable component interfaces, then $B\boxtimes C$ is a well-defined component interface.*

Proof. We have to show that the conditions of Def. 5 are satisfied by $B \boxtimes C$. The proof is straightforward by simple set-theoretic reasoning using the properties (L1) - (L3) of labeled interface theories.

1. Obviously, the first condition of Def. 5 is satisfied, since $\ell(F^B) \cap \ell(F^C) = \bigcup_{i\in I_0} \ell(P_i^B)$ $= \bigcup_{j\in J_0} \ell(P_j^C))$ and therefore the labels of non-connected ports P_i^B, P_j^C are disjoint for all $i \in I \setminus I_0, j \in J \setminus J_0$. The labels of the (still open) ports of B (C resp.) are anyway pairwise disjoint.
2. Also the second condition of Def. 5 is satisfied: $\ell(F^B \otimes F^C) =$, by cond. (L1) for labeled interface theories, $(\ell(F^B) \cup \ell(F^C)) \setminus (\ell(F^B) \cap \ell(F^C)) =$ (by cond. 2 of the definition) $(\ell(F^B) \cup \ell(F^C)) \setminus \bigcup_{i\in I_0} \ell(P_i^B) =$ (by cond. 2 for component interfaces) $(\bigcup_{i\in I} \ell(P_i^B) \cup \bigcup_{j\in J} \ell(P_j^C)) \setminus \bigcup_{i\in I_0} \ell(P_i^B) = (\bigcup_{i\in(I\setminus I_0)} \ell(P_i^B) \cup \bigcup_{j\in(J\setminus J_0)} \ell(P_j^C))$.

3. For the third condition of Def. 5 we have to show that $(A_i^B)_{i\in(I\setminus I_0)} \cup (A_j^C)_{j\in(J\setminus J_0)} \cup \{F^B \otimes F^C\}$ is composable. Since the labels of these assumptions are pairwise disjoint, they are also pairwise composable by cond. (L2) for labeled interface theories. It remains to show that $F^B \otimes F^C$ is composable with each of the assumptions. W.l.o.g. consider some A_i^B with $i \in (I \setminus I_0)$. Since B is a component interface, A_i^B is composable with F^B. Since $\ell(A_i^B) \cap \ell(F^C) = \emptyset$, A_i^B is composable with F^B by cond. (L2) for labeled interface theories. Then, by associativity of composition, A_i^B is composable with $F^B \otimes F^C$. □

Obviously, component composition $\boxtimes$ is commutative since the underlying composition operator $\otimes$ and the set-theoretic union of (non-connected) ports is commutative.

4.3. Refinement of Component Interfaces

Our definition of component interface refinement relies on refinement of ports, see Def. 4. A component interface B' refines another one B if, first, both have the same number of ports such that each port of B is refined by the corresponding port of B' and, secondly, the frame of B is refined by the frame of B' 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 7 (Component Interface Refinement). Let $C = ((P_i^C)_{i\in I}, F^C)$ and $C' = ((P_j^{C'})_{j\in J}, F^{C'})$ be two component interfaces. C' *refines* C, written $C' \sqsubseteq C$, if there exists a bijection $\rho : I \to J$ such that

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

Note that reflexivity and transitivity of $\sqsubseteq$ is inherited from the underlying refinement relation $\leq$ for interfaces.

Theorem 4 (Compositionality of Component Interface Refinement). *Let B, B', C and C' be component interfaces such that B and C are composable and $B' \sqsubseteq B$ as well as $C' \sqsubseteq C$ holds. Then B' and C' are composable and $B' \boxtimes C' \sqsubseteq B \boxtimes C$.*

PROOF. We first show that B' and C' are composable. For simplicity, we assume that the bijections between the index sets used for composition and for refinements are identities.

1. Consider condition (1) of Def. 6. Let $P_i^B = (A_i^B, G_i^B)$ and $P_i^C = (A_i^C, G_i^C)$ be connected ports of B and C and let $P_i^{B'} = (A_i^{B'}, G_i^{B'})$ and $P_i^{C'} = (A_i^{C'}, G_i^{C'})$ such that $P_i^{B'} \sqsubseteq P_i^B$ and $P_i^{C'} \sqsubseteq P_i^C$. By assumption P_i^B and P_i^C have the same labels and their guarantees (assumptions resp.) are composable. Since $P_i^{B'} \sqsubseteq P_i^B$ and $P_i^{C'} \sqsubseteq P_i^C$ also $\ell(P_i^{B'}) = \ell(P_i^B)$ and $\ell(P_i^{C'}) = \ell(P_i^C)$, hence $\ell(P_i^{B'}) = \ell(P_i^{C'})$. Moreover, since $G_i^{B'} \leq G_i^B$ and G_i^B is composable with G_i^C, it follows by (I1) (compositionality of refinement of interface theories), that $G_i^{B'}$ is composable with G_i^C. Since $G_i^{C'} \leq G_i^C$ we obtain, again by (I1), that $G_i^{B'}$ is composable with $G_i^{C'}$. Similarly for the assumptions.

2. For condition (2) of Def. 6 we have to show that $\ell(F^{B'}) \cap \ell(F^{C'}) = \bigcup_{i\in I_0} \ell(P_i^{B'})$. This follows from $\ell(F^B) \cap \ell(F^C) = \bigcup_{i\in I_0} \ell(P_i^B)$ and pairwise port refinement, taking into account that the labels of a frame are the disjoint union of the labels of the ports.

3. Condition (3) of Def. 6, i.e. composability of $F^{B'}$ and $F^{C'}$, follows from composability of F^B and F^C and from $F^{B'} \le F^B$ and $F^{C'} \le F^C$ and by compositional refinement.

By compositional refinement we also obtain $F^{B'} \otimes F^{C'} \le F^B \otimes F^C$, i.e. the second condition of Def. 7 is satisfied. The first condition of Def. 7 follows from the pairwise refinement relations between the ports of B and B' (C and C' resp.) such that, in particular, the open ports of the $B \boxtimes C$ are pairwise refined by the open ports of $B' \boxtimes C'$. □

4.4. Compatible Component Interfaces

In order to obtain an interface theory for component interfaces with ports, we still need to define a suitable compatibility predicate. We propose a very intuitive notion, that is two component interfaces B and C are compatible if assumptions and guarantees on connected ports match. This can be easily expressed by the refinement relation of the underlying interface theory. Fig. 4 shows the condition for compatible component interfaces in the case of a single port connection.

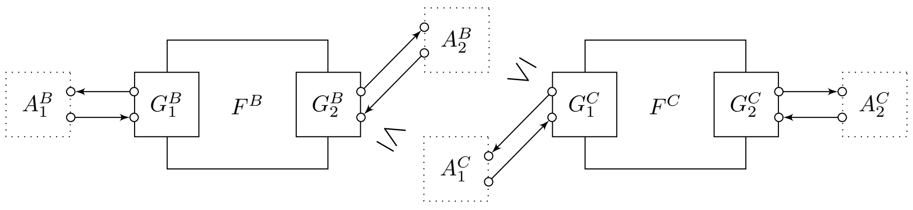


Figure 4: Compatibility of Component Interfaces

Definition 8. Let $B = ((P_i^B)_{i\in I}, F^B)$ and $C = ((P_j^C)_{j\in J}, F^C)$ be two component interfaces. B and C are *compatible*, denoted by $B \leftrightarrows C$, if they are composable according to a bijective connector function $\kappa : I_0 \to J_0$ for non-empty subsets $I_0 \subseteq I$ and $J_0 \subseteq J$, such that for all $i \in I_0$,

$$G_i^B \le A_{\kappa(i)}^C \quad \text{and} \quad G_{\kappa(i)}^C \le A_i^B.$$

Theorem 5 (Preservation of Compatibility). *Let B, B', C and C' be component interfaces such that $B' \sqsubseteq B$ and $C' \sqsubseteq C$. If $B \leftrightarrows C$, then also $B' \leftrightarrows C'$.*

Proof. Composability of B' and C' follows from composability of B and C and from Thm. 4. For simplicity, we assume that the bijections between the index sets used for the refinements are identities. To prove $B' \leftrightarrows C'$, we show that $G_i^{B'} \le A_{\kappa(i)}^{C'}$ for all $i \in I_0$. The converse property $G_{\kappa(i)}^{C'} \le A_i^{B'}$ is proved analogously.

Since $B' \sqsubseteq B$, we get $G_i^{B'} \le G_i^B$. Since $B \leftrightarrows C$, we have $G_i^B \le A_{\kappa(i)}^C$. Since $C' \sqsubseteq C$, we have $A_{\kappa(i)}^C \le A_{\kappa(i)}^{C'}$. Hence, by transitivity of refinement, $G_i^{B'} \le A_{\kappa(i)}^{C'}$. □

As a consequence of Thms. 4 and 5 we get the following result.

Corollary 6. *Let $\mathfrak{C}$ be the class of all component interfaces with port contracts built over an arbitrary labeled interface theory $(\mathfrak{S}, \otimes, \leq, \leftrightarrows, \mathcal{L}, \ell)$. Then $(\mathfrak{C}, \boxtimes, \sqsubseteq, \mathrel{\substack{\twoheadleftarrow \\ \twoheadrightarrow}})$ is an interface theory.*

4.5. Reliable 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 interface 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 a user who wants to connect to a certain port is trusting the guarantee on that port which should be established by 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 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.

Definition 9 (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* if

$$\otimes (A_i)_{i \in I \setminus \{j\}} \otimes F \leq G_j \text{ for all } j \in I.$$

Discussion. One may wonder why the assumption A_j is not also used for ensuring the guarantee G_j on port P_j. Assume we would require $\otimes(A_i)_{i \in I} \otimes F \leq G_j$. Then $\otimes(A_i)_{i \in I} \otimes F$ is a closed system (with no visible labels) while G_j is supposed to be open for the environment. This cannot work. Another possibility would be to require $\otimes(A_i)_{i \in I} \otimes F \leq (G_j \otimes A_j)$. This would be an interesting approach. It follows the idea that on each port P_j a component guarantees to cooperate, as indicated by G_j, with any environment (connected to P_j) that satisfies A_j. But this would only make sense if the cooperation between A_j and G_j can be observed which does not fit to the intuition of interface composition making all communications internal. As a consequence we would need significantly more ingredients for the underlying interface theories. An attempt could be to use two different composition operators, $\otimes$ that hides communication and $\overline{\otimes}$ which does not hide communication, and require for reliability the condition $(\otimes(A_i)_{i \in I \setminus \{j\}} \otimes F) \,\overline{\otimes} A_j \leq (G_j \overline{\otimes} A_j)$. How this could work and which additional laws are needed is an interesting topic for future research.

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 the reliable components to be composed are compatible and if the composition uses a single connection between two ports; see Thm. 7 below. 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 the frame is a refinement of the product of the behaviors (i.e. guarantees) of the ports under consideration.

Definition 10. 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 $\emptyset \neq I_0 \subseteq I$. 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. $\otimes(G_j)_{j \in I_0} \leftrightarrows \otimes(A_j)_{j \in I_0}$.

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

Theorem 7 (Preservation of reliability). *Let B and C 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 $B \overset{\leftarrow}{\twoheadrightarrow} C$ implies that $B \boxtimes C$ is reliable.*

Proof. Let $B = ((P_i^B)_{i \in I}, F^B)$ and $C = ((P_j^C)_{j \in J}, F^C)$ be composable according to a bijective connector function $\kappa : I_0 \to J_0$ for non-empty subsets $I_0 \subseteq I$ and $J_0 \subseteq J$.

W.l.o.g. let $i' \in I \setminus I_0$. We have to show that

$$\otimes(A_i^B)_{i \in I \setminus (I_0 \cup \{i'\})} \otimes F^B \otimes F^C \otimes (\otimes(A_j^C)_{j \in J \setminus J_0}) \leq G_{i'}^B.$$

This is shown using the following refinement steps:

$$\begin{aligned}
&\otimes (A_i^B)_{i \in I \setminus (I_0 \cup \{i'\})} \otimes F^B \otimes F^C \otimes (\otimes(A_j^C)_{j \in J \setminus J_0}) \\
&\leq \otimes(A_i^B)_{i \in I \setminus (I_0 \cup \{i'\})} \otimes F^B \otimes (\otimes(G_j^C)_{j \in J_0}) \\
&\qquad \text{(by independence of } (P_j^C)_{j \in J_0} \text{ and compositional refinement)} \\
&\leq \otimes(A_i^B)_{i \in I \setminus (I_0 \cup \{i'\})} \otimes F^B \otimes (\otimes(A_i^B)_{i \in I_0}) \\
&\qquad \text{(by } B \overset{\leftarrow}{\twoheadrightarrow} C\text{, i.e. } G_{\kappa(i)}^C \leq A_i^B \text{ for all } i \in I_0\text{, and compositional refinement)} \\
&= \otimes(A_i^B)_{i \in I \setminus \{i'\}} \otimes F^B \leq G_{i'}^B \qquad \text{(by reliability of } B\text{)}
\end{aligned}$$

□

The next important result implies that for compatible, reliable component interfaces B and C which are connected via independent ports the composition of the frame F^B with environments for B is compatible with the composition of F^C with environments for C.

Theorem 8. *Let $B = ((P_i^B)_{i \in I}, F^B)$ and $C = ((P_i^C)_{j \in J}, F^C)$ be reliable component interfaces which are composable according to a bijective connector function $\kappa : I_0 \to J_0$ for non-empty subsets $I_0 \subseteq I$ and $J_0 \subseteq J$. Assume that the connected ports on each side are independent. Then $B \overset{\leftarrow}{\twoheadrightarrow} C$ implies*

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

Proof. From independence of the ports $(P_i^B)_{i\in I_0}$ we get

$$\otimes\,(A_i^B)_{i\in(I\setminus I_0)} \otimes F^B \le \otimes(G_i^B)_{i\in I_0} \tag{1}$$

Moreover, independence of the ports $(P_i^C)_{j\in J_0}$ and compatibility of B and C implies

$$F^C \otimes (\otimes(A_j^C)_{j\in(J\setminus J_0)}) \le \otimes(G_j^C)_{j\in J_0} \le \otimes(A_i^B)_{i\in I_0}. \tag{2}$$

By independence of ports $(P_i^B)_{i\in I_0}$ we have $\otimes(G_i^B)_{i\in I_0} \leftrightarrows \otimes(A_i^B)_{i\in I_0}$. The claim follows from transitivity of refinement and preservation of compatibility. $\square$

Example 2. Fig. 5 shows two component interfaces B and C built with MIOs. First we can observe, that both interfaces are reliable. For instance, considering the first port of B, $A_2^B \otimes^{\mathrm{sy}} F^B \le_{\mathrm{m}}^* G_1^B$. Considering the first port of C the proof obligation is $A^C \otimes^{\mathrm{sy}} A_2^C \otimes^{\mathrm{sy}} F^C \le_{\mathrm{m}}^* G_1^C$. This and all other proof obligations can be easily discharged since we deal with weak refinement.

The two interfaces B and C are composable and also compatible since trivially $G_i^B \le_{\mathrm{m}}^* A_i^C$ and $G_i^C \le_{\mathrm{m}}^* A_i^B$ for $i = 1, 2$. Moreover, the two ports of B are independent, since $F^B \le_{\mathrm{m}}^* G_1^B \otimes^{\mathrm{sy}} G_2^B$ and $(G_1^B \otimes^{\mathrm{sy}} G_2^B) \rightleftarrows_{\mathrm{w}}^{\mathrm{m}} (A_1^B \otimes^{\mathrm{sy}} A_2^B)$.

Also the ports 1 and 2 of C are independent, since $A^C \otimes^{\mathrm{sy}} F^C \le_{\mathrm{m}}^* G_1^C \otimes^{\mathrm{sy}} G_2^C$ and $(G_1^C \otimes^{\mathrm{sy}} G_2^C) \rightleftarrows_{\mathrm{w}}^{\mathrm{m}} (A_1^C \otimes^{\mathrm{sy}} A_2^C)$.

The composition of B and C is shown in Fig. 6. According to Thm. 7 it is reliable which is indeed the case since, obviously, $F^B \otimes^{\mathrm{sy}} F^C \le_{\mathrm{m}}^* G^C$. By Thm. 8 we also know that $F^B \rightleftarrows_{\mathrm{w}}^{\mathrm{m}} (A^C \otimes^{\mathrm{sy}} F^C)$ which can be easily checked. In this case it is important that the compatibility relation is weak disregarding silent must-transitions.

Let us now discuss modifications of B and C such that all transitions in $G_1^B, G_2^B, A_1^C, A_2^C$ and in G^C become must-transitions. The single component interfaces are still reliable. But the ports of C are not independent anymore since $F^B \not\le_{\mathrm{m}}^* (G_1^B \otimes^{\mathrm{sy}} G_2^B)$. The reason is that F^B supports only alternating outputs $a!$ and $b!$ while the product $G_1^B \otimes^{\mathrm{sy}} G_2^B$ now requires that arbitrary interleaving of outputs must be possible. We can also see that the composition of B and C is not reliable anymore, i.e. $F^B \otimes^{\mathrm{sy}} F^C \not\le_{\mathrm{m}}^* G^C$, since the new G^C has only must-transitions.

5. Modal Component Interfaces

In Sect. 3 we have presented the modal labeled interface theory based on MIOs. In Sect. 4 we have provided a generic approach to construct component interfaces with port contracts. We will now apply this procedure to MIOs and obtain a theory of modal component interfaces. It is given by the tuple $(\mathcal{C}^{MIO}, \boxtimes^{\mathrm{sy}}, \sqsubseteq^{\mathrm{m}}, \leftrightarrows_{\mathrm{m}}^{*})$ such that $\mathcal{C}^{MIO}$ denotes the class of modal interfaces with modal port contracts, $\boxtimes^{\mathrm{sy}}$ denotes their synchronous composition, $\sqsubseteq^{\mathrm{m}}$ their refinement and $\leftrightarrows_{\mathrm{m}}^{*}$ denotes compatibility of modal component interfaces. According to Cor. 6, all requirements of an interface theory are satisfied.

As an illustration we consider a simple message transmission system which consists of two component interfaces: the *Broker* introduced in Ex. 1 receives messages from its environment and delivers them to a *Client*. A standard message is immediately delivered while a confidential message is only delivered after successful authentication of the client.

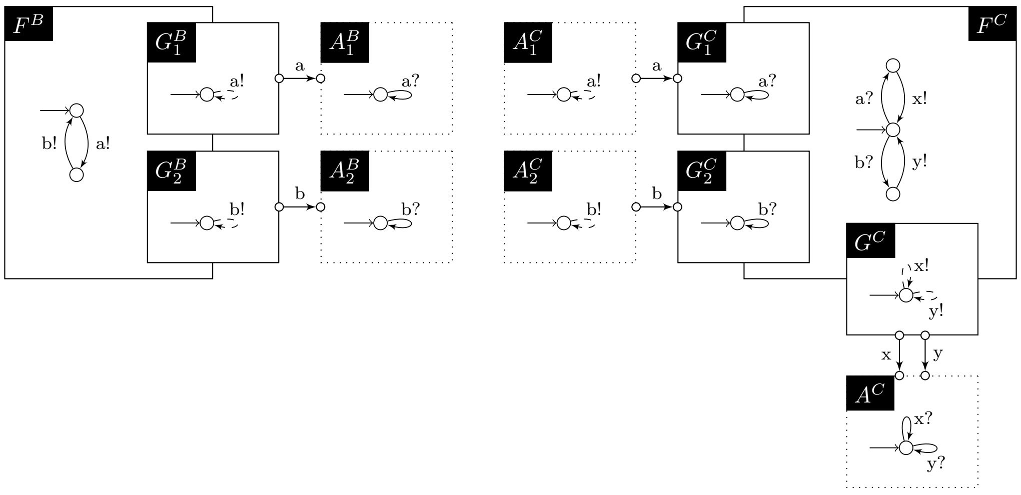


Figure 5: Compatible Component Interfaces with Independent Ports

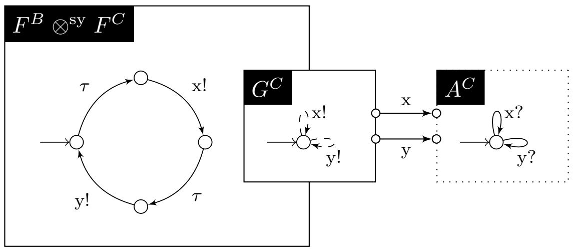


Figure 6: Composition of B and C

The architecture of the system is shown in Fig. 7. The meaning of the input and output actions is summarized in Table 1.

The details of the *Broker* and the *Client* component interfaces are shown in Fig. 8. The *Broker* interface has already been explained in Ex. 1. It is reliable according to Def. 9. The proof obligations are $A_1^B \otimes^{\text{sy}} F^B \leq_{\text{m}}^{*} G_2^B$ and $A_2^B \otimes^{\text{sy}} F^B \leq_{\text{m}}^{*} G_1^B$. They are detailed in Fig. 9. The weak modal refinement relations can be easily discharged, for instance, with the MIO Workbench [9]. Indeed it is crucial here and in the whole approach that we use weak versions of refinement and compatibility which allow us to abstract from silent transitions.

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

Table 1: Meaning of the actions

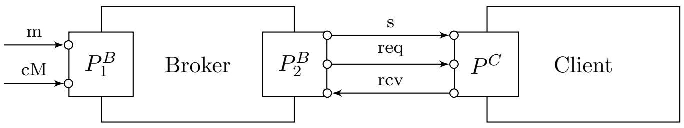


Figure 7: Architecture of the message transmission system

The interface of the *Client* component is much simpler. It has only one port such that the guarantee of the port coincides with the frame which immediately implies reliability. The behavior specifications are self-explanatory. Just note, that the assumption A^C requires that any answer *rcv*! of the client to an authentication request must be received *rcv*? by the environment of the client; hence $G^C \rightleftarrows^{\mathrm{m}}_{\mathrm{w}} A^C$.

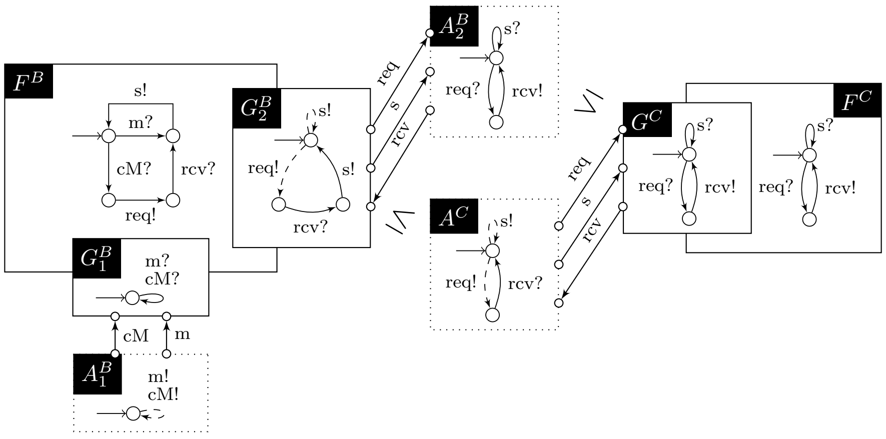


Figure 8: Compatible component interfaces for Broker and Client

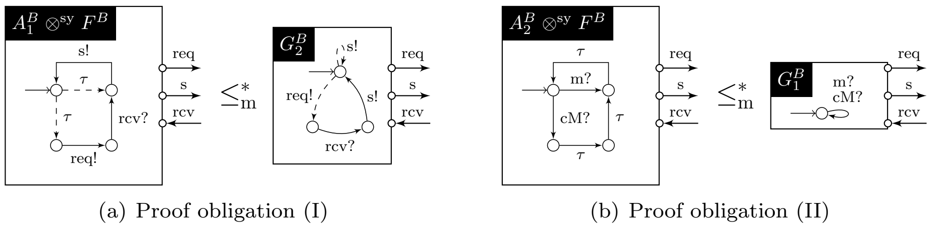


Figure 9: Proof obligations for reliability of the Broker interface

Clearly, the two component interfaces are composable by connecting the ports P_2^B and P^C, which both have the same labels and composable guarantees and assumptions. The two component interfaces are compatible, i.e. *Broker* $\leftrightarrows^*_{\mathrm{m}}$ *Client*. The proof obligations are $G^C \leq^*_{\mathrm{m}} A_2^B$ (which is trivially valid) and $G_2^B \leq^*_{\mathrm{m}} A^C$ which can be easily discharged. Their composition yields the interface *Broker* $\boxtimes^{\mathrm{sy}}$ *Client* shown in Fig. 10.[6]

[6]The composed frame $F^B \otimes^{\mathrm{sy}} F^C$ is in fact equivalent to the guarantee G_1^B and hence could be

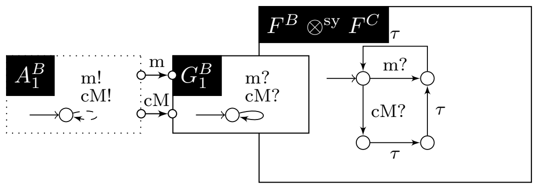


Figure 10: Composition *Broker* $\boxtimes^{sy}$ *Client*

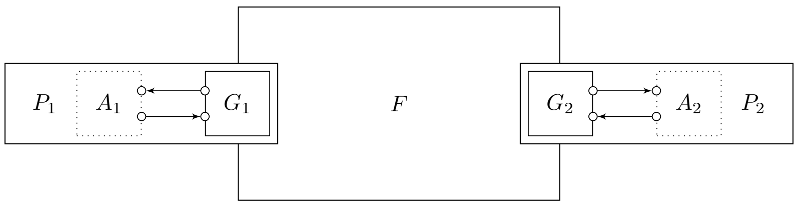


Figure 11: Component interface with two ports

Due to Thm. 7 the composed interface must be reliable (which is trivial in the example) since the single interfaces are reliable and compatible.

6. Design and Adaptation of Component Interfaces

In this section, we discuss methodological guidelines for component interface design and adaptation. They are implied by the structural and semantic assumptions on reliable component interfaces, in particular by the fact that the guarantee on one port must be satisfied by the frame when combined with the assumptions on *the other* ports.

6.1. Designing Reliable Component Interfaces.

Our objective is to design a reliable component interface with two ports as shown in Fig. 11. We rely on a given component frame F and on a partition of the labels of F into ports. We propose three methods for the design of the single guarantees and assumptions on the ports.

Construction of assumptions. In the first case, we assume that after the frame specification F has been fixed we have provided guarantees G_1 and G_2 on each port. The goal is to find appropriate assumptions A_1 and A_2. According to the conditions for port contracts and reliability, we must find solutions for the following constraints:

(1) $A_1 \leftrightarrows G_1, A_2 \leftrightarrows G_2,$ and
(2) $A_1 \otimes F \leq G_2, A_2 \otimes F \leq G_1.$

An interpretation of (2) is to find a "controller" A_1 of F such that their composition satisfies G_2 and, similarly, to find a controller A_2 of F to satisfy G_1. Additionally, the controllers must be compatible with the corresponding guarantees to satisfy (1).

minimized. Whether minimization of MIOs w.r.t. equivalence (see Sect. 3.2) is possible is an interesting issue which is out of the scope of this paper.

Construction of port contract. In the second case, we assume that for the given frame F we have provided a port contract (A_1, G_1) for the first port. Then, the goal is to find an appropriate port contract (A_2, G_2) for the second port. Of course, the constraints (1) and (2) are still valid, but this time we can directly compute $G_2 = A_1 \otimes F$ which gives us the strongest guarantee to satisfy part one of constraint (2). It remains to design A_2 in such a way that:

$$A_2 \leftrightarrows G_2 \text{ and } A_2 \otimes F \leq G_1.$$

We propose the following method to design A_2: First we look for a most liberal assumption, say A'_2, which is compatible with G_2, i.e. $A'_2 \leftrightarrows G_2$. Then we check, whether $A'_2 \otimes F \leq G_1$. If this is the case, we take $A_2 = A'_2$ and are done. Otherwise we must try to find a stronger assumption $A_2 \leq A'_2$ such that $A_2 \otimes F \leq G_1$ holds. If we succeed we have for free $A_2 \leftrightarrows G_2$, since compatibility is preserved by refinement.

Construction of guarantees. In the last case, we assume that we have given the frame F and assumptions A_1 and A_2 on each port. The goal is to find appropriate guarantees G_1 and G_2 satisfying the constraints (1) and (2). To satisfy (2) we can compute $G_1 = A_2 \otimes F$ and $G_2 = A_1 \otimes F$ which are the strongest possible guarantees under the given assumptions. To satisfy (1) the guarantees may still be weakened. But, in general, there is no guarantee that the constraints are indeed solvable.

Example 3. We illustrate the second design method on the construction of the *Broker* interface. Given the frame F^B we have decided to guarantee on port P^B_1 that any kind of messages (standard and confidential ones) must always be accepted; see G^B_1. We have put the most liberal assumption A^B_1 on the environment on that port; the environment may always send messages (but does not need to send). Then we have computed the product $A^B_1 \otimes^{\mathrm{sy}} F^B$ which is shown in Fig. 9(a). We then decided that the guarantee on port P^B_2 should not show silent transitions which are usually not helpful to the user of a component. Therefore, we have proposed the actual guarantee G^B_2 such that $A^B_1 \otimes^{\mathrm{sy}} F^B \leq^*_{\mathrm{m}} G^B_2$. This intermediate weakening step slightly deviates from the method discussed above.

Then we continued, following the steps of our second design method above, by finding a most liberal assumption, call it A'^B_2, such that $A'^B_2 \rightleftarrows^{\mathrm{m}}_{\mathrm{w}} G^B_2$. A'^B_2 is a variant of A^B_2 such that the transition with *rcv*! is a may-transition. Then we had to check $A'^B_2 \otimes^{\mathrm{sy}} F^B \leq^*_{\mathrm{m}} G^B_1$, but this failed, since under the assumption of a may-transition for *rcv*! on port P^B_2 the component could not guarantee to continue accepting messages on port P^B_1 as guaranteed by G^B_1. Therefore we have strengthened the assumption A'^B_2 to the actual assumption A^B_2 with a must-transition for *rcv*!. Finally we have $A^B_2 \otimes^{\mathrm{sy}} F^B \leq^*_{\mathrm{m}} G^B_1$ (see Fig. 9(b)) and $A^B_2 \rightleftarrows^{\mathrm{m}}_{\mathrm{w}} G^B_2$ is guaranteed for free.

6.2. Adaptation of Component Interfaces.

We discuss a method to adapt component interfaces to changing environments. We assume a reliable component interface B which has been put in a new environment C, represented itself by a reliable component interface, such that both interfaces are not compatible. Our initial situation is shown in Fig. 12. The component interfaces do mutually not satisfy each others assumptions. Our goal is to overcome this problem by adaptation.

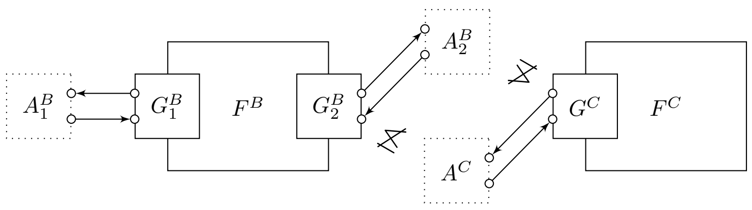


Figure 12: Adaptation of component interfaces: initial situation

We assume that we cannot change the environment C. Therefore, we must try to adapt the component interface B such that B remains reliable and B and C become compatible. The adaptation should be performed by adjusting assumptions and guarantees but not changing the frame of B. Our adaptation procedure consist of the following steps:

1. Replace A_2^B by G^C. Then the new assumption on the second port of B coincides with the guarantee on the port of C.
2. Replace G_2^B by A^C. Then the new guarantee on the second port of B coincides with the assumption on the port of C. This gives a correct port contract since we know that G^C and A^C are compatible (on the C port). Also B and C are now trivially compatible component interfaces, since assumptions and guarantees are the same.
3. Compute $G^C \otimes F^B$ (notice that, according to step 1, G^C is the new assumption on the second port) and replace G_1^B by $G^C \otimes F^B$. This means that $G^C \otimes F^B$ is the new guarantee on the first port of B and therefore B is reliable on the first port by construction.
4. Finally we have to find an assumption $\bar{A}_1^B$ on the first port of B such that
   (a) $\bar{A}_1^B \leftrightarrows G^C \otimes F^B$
   (notice that $G^C \otimes F^B$ is the new guarantee on the first port of B), and
   (b) $\bar{A}_1^B \otimes F^B \leq A^C$.

The last step is the most complicated one. It can be approached in the same way as proposed in the last subsection for constructing a port contract: First, we construct a most liberal assumption for the first port of B that is compatible with $G^C \otimes F^B$. Then, we check whether (b) is satisfied and, if not, we try to strengthen the assumption accordingly. If we are successful the final situation after adaptation is shown in Fig. 13.

Example 4. We apply our adaptation methodology to the *Broker* and *Client* example. Let us assume that the client has changed and does not support a treatment of authentication anymore. This is modeled by the new *Client* interface in Fig. 14 showing the initial situation of our adaptation problem.

Following the steps of our general procedure we achieve a solution shown in Fig. 15. The adaptation has been computed automatically, except for the new assumption $\bar{A}_1^B$ which has been easily designed by hand.

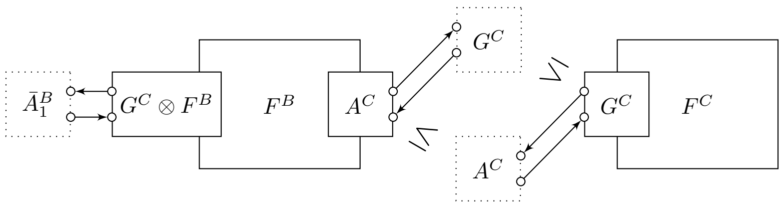


Figure 13: Adaptation of component interfaces: final situation

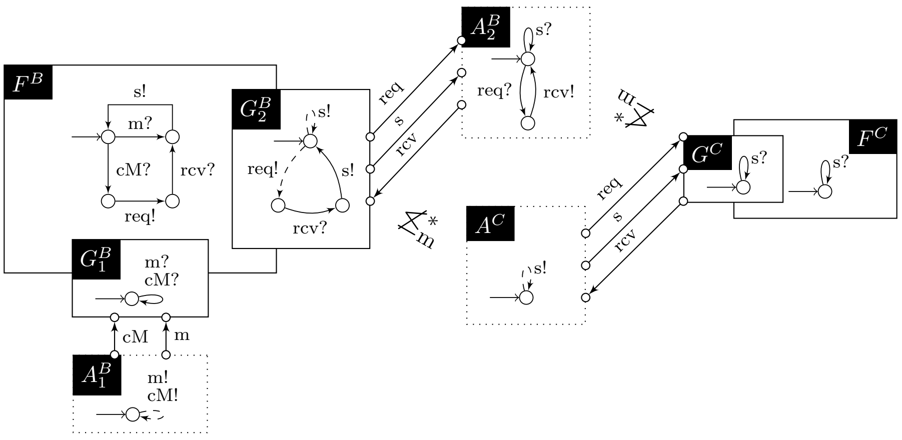


Figure 14: Broker interface with modified Client

7. Component Interfaces with I/O-Predicates

In this section we discuss another instantiation of our approach. The underlying labeled interface theory uses I/O-predicates as interface specifications. I/O-predicates follow the idea to specify programs as relations between input and output observations as suggested in the Unifying Theories of Programming [26]. In the following input observations will be determined by input variables and output observations by output variables.

We assume a universal set $\mathcal{L}^{\mathrm{p}}$ of (untyped) variables and a language that contains predicates over variables closed under the usual Boolean connectives and constants and under existential and universal quantification: If P is a predicate with free variables $FV(P)$ and $X \subseteq \mathcal{L}^{\mathrm{p}}$ then $\exists X : P$ and $\forall X : P$ are predicates as well with free variables $FV(P) \setminus X$. If $X = \emptyset$ then $\exists X : P$ stands for P and $\forall X : P$ stands for *true*. An I/O-predicate is a triple $P(In_P, Out_P)$ such that P is a predicate, $In_P \subseteq \mathcal{L}^{\mathrm{p}}$ and $Out_P \subseteq \mathcal{L}^{\mathrm{p}}$ are sets of input and output variables respectively, such that $In_P \cap Out_P = \emptyset$ and $FV(P) \subseteq In_P \cup Out_P$. The set of I/O-predicates is denoted by $\mathfrak{S}^{\mathrm{p}}$. It forms our underlying set of interface specifications. The set of labels is given by $\mathcal{L}^{\mathrm{p}}$ and the labeling function $\ell^{\mathrm{p}} : \mathfrak{S}^{\mathrm{p}} \to \wp_{\mathrm{fin}}(\mathcal{L}^{\mathrm{p}})$ is defined by $\ell^{\mathrm{p}}(P(In_P, Out_P)) = In_P \cup Out_P$ for each $P(In_P, Out_P) \in \mathfrak{S}^{\mathrm{p}}$.

Two I/O-predicates $P(In_P, Out_P)$ and $Q(In_Q, Out_Q)$ are *composable* if their variables overlap at most on complementary types, i.e. $In_P \cap In_Q = \emptyset$ and $Out_P \cap Out_Q = \emptyset$. We define the composition of two composable I/O-predicates $P(In_P, Out_P)$ and

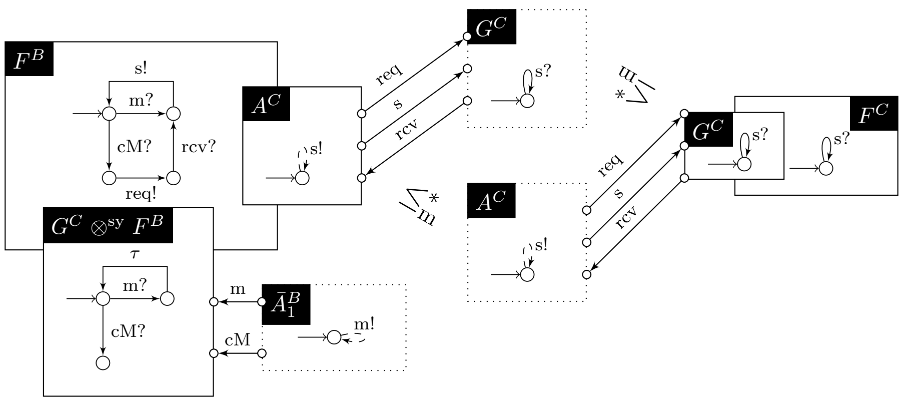


Figure 15: Adaptation of the Broker interface

$Q(In_Q, Out_Q)$ by

$$P(In_P, Out_P) \otimes^{\mathrm{p}} Q(In_Q, Out_Q) = (\exists (Out_P \cap In_Q) \cup (In_P \cap Out_Q) : P \wedge Q)(In_{PQ}, Out_{PQ})$$

with $In_{PQ} = (In_P \cup In_Q) \setminus \ell^{\mathrm{p}}(P) \cap \ell^{\mathrm{p}}(Q)$, $Out_{PQ} = (Out_P \cup Out_Q) \setminus \ell^{\mathrm{p}}(P) \cap \ell^{\mathrm{p}}(Q)$.

The composition operator is commutative and associative which follows from set-theory and from the logical commutativity and associativity laws. Obviously, the composition operator satisfies also the labeling conditions (L1) and (L2) for labeled interface theories in Def. 2. It can be considered as a generalization of the sequential composition of predicates via the existence of intermediate observations like in [26]. For instance, if $Out_P = In_Q$ and $Out_Q \cap In_P = \emptyset$ then our predicate composition expresses exactly relational composition:

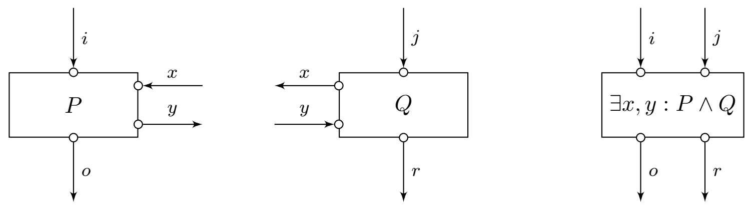
$$P(In_P, Out_P) \otimes^{\mathrm{p}} Q(In_Q, Out_Q) = (\exists In_Q : P \wedge Q)(In_P, Out_Q).$$

Fig. 16 shows an example for the most general situation of composition which involves shared input/output variables $\{x, y\}$ in both directions and inputs and outputs ($\{i, o\}$ and $\{j, r\}$) on both sides that remain open after composition. Of course the composition of I/O-predicates can lead to inconsistent specifications, in particular if there are mutual dependencies. A variant of our approach has been considered in [17] for the composition and (possibly mutually dependent) connection of "stateless input/output interfaces" which are similar to I/O-predicates.

The refinement notion for I/O-predicates follows the idea of predicate refinement by implication from the concrete to the abstract specification. We must, however, be careful with inputs; the concrete specification should accept any inputs that belong to the domain of the abstract specification. For this purpose we consider weakest preconditions. For an I/O-predicate $P(In_P, Out_P)$ the weakest precondition is given by the predicate $\exists Out_P : P$. An I/O-predicate $P'(In_{P'}, Out_{P'})$ is a *refinement* of an I/O-predicate $P(In_P, Out_P)$, denoted by $P'(In_{P'}, Out_{P'}) \leq^{\mathrm{p}} P(In_P, Out_P)$, if the following conditions are satisfied.

(r1) $In_{P'} = In_P$ and $Out_{P'} = Out_P$,

(r2) $\forall In_P : (\exists Out_P : P) \Rightarrow (\exists Out_P : P')$,



(a) I/O-predicate $P(\{i, x\}, \{y, o\})$ (b) I/O-predicate $Q(\{j, y\}, \{x, r\})$ (c) Composition

Figure 16: Composition of I/O-predicates

(r3) $\forall In_P \cup Out_P : P' \Rightarrow P$.

Obviously refinement of I/O-predicates is reflexive and transitive and satisfies the labeling condition (L3) in Def. 2.

An interesting question is how compatibility should be defined for I/O-predicates. The idea is that any possible output of one predicate should be accepted as input by the other predicate as expressed by the conditions (c1.1) and (c2.1) below. In the special case of sequential composition as discussed above this would be sufficient. However in the context of mutual dependencies we need an additional condition which requires that outputs fed back to the other component must be uniquely determined by the other variables. This is expressed by the conditions (c1.2) and (c2.2) below.

Two I/O-predicates $P(In_P, Out_P)$ and $Q(In_Q, Out_Q)$ are *compatible*, denoted by $P(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q(In_Q, Out_Q)$, if they are composable and if the following conditions are satisfied:

(c1) if $Out_P \cap In_Q \neq \emptyset$, then
(c1.1) $\forall \ell^{\mathrm{p}}(P) : (P \Rightarrow \exists \ell^{\mathrm{p}}(Q) \setminus Out_P : Q)$, and
if additionally $Out_Q \cap In_P \neq \emptyset$, then
(c1.2) $\forall \ell^{\mathrm{p}}(P) : (P \Rightarrow \mathbf{UniqueOut}(Q, Out_Q \cap In_P))$

(c2) if $Out_Q \cap In_P \neq \emptyset$, then
(c2.1) $\forall \ell^{\mathrm{p}}(Q) : (Q \Rightarrow \exists \ell^{\mathrm{p}}(P) \setminus Out_Q : P)$, and
if additionally $Out_P \cap In_Q \neq \emptyset$, then
(c2.2) $\forall \ell^{\mathrm{p}}(Q) : (Q \Rightarrow \mathbf{UniqueOut}(P, Out_P \cap In_Q))$.

where $\mathbf{UniqueOut}(Q, Out_Q \cap In_P) =$

$$\forall (\ell^{\mathrm{p}}(Q) \setminus Out_P)[X''/X] : (Q[X''/X] \Rightarrow X'' = X),$$

such that $X = Out_Q \cap In_P$, X'' is the set of the double primed variables of X, $[X''/X]$ denotes the substitution of all variables in X by their double primed versions, and $X'' = X$ denotes the conjunction of all equations $x'' = x$ with $x \in X$. $\mathbf{UniqueOut}(P, Out_P \cap In_Q)$ is defined analogously.

The compatibility relation is obviously symmetric. In some special cases the compatibility conditions are much simpler. If two I/O-predicates have no shared variables, then they are trivially compatible. If there are only shared variables in one direction, for instance if $Out_P = In_Q$ and $Out_Q \cap In_P = \emptyset$, then $P(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q(In_Q, Out_Q)$ holds if just condition (c1.1) is satisfied, which in this case reads as $\forall \ell^{\mathrm{p}}(P) : (P \Rightarrow \exists Out_Q : Q)$. This case expresses compatibility for sequential composition of predicates requiring that the outputs of $P(In_P, Out_P)$ belong to the input domain of $Q(In_Q, Out_Q)$ given by the weakest precondition of Q which is $\exists Out_Q : Q$.

Example 5. We provide an example for compatible I/O-predicates with mutual dependencies. Consider Fig. 16 and let P be $y = x + 1$ and Q be $x = y - 1$. Then condition (c1.1) is: $\forall i, x, y, o : (y = x + 1 \Rightarrow \exists j, x, r : x = y - 1)$ which is trivially true.

Condition (c1.2) is: $\forall i, x, y, o : (y = x + 1 \Rightarrow \forall j, x'', r : (x'' = y - 1 \Rightarrow x'' = x))$ which can be easily verified. Conditions (c2.1) and (c2.2) hold analogously. Hence the two I/O-predicates are compatible.

Let us now change the predicate Q to $x = y+1$. Then (c1.1) remains valid but (c1.2) becomes false. Hence the two I/O-predicates would not be compatible in this case.

To get a labeled interface theory of I/O-predicates it remains to show that the properties of compositional refinement and preservation of compatibility are satisfied.

Proposition 9. *For all I/O-predicates $S, S', T, T' \in \mathfrak{S}^{\mathrm{p}}$ the following holds:*

1. Compositional Refinement: *If $S \otimes^{\mathrm{p}} T$ is defined, $S' \leq^{\mathrm{p}} S$ and $T' \leq^{\mathrm{p}} T$, then $S' \otimes^{\mathrm{p}} T'$ is defined and $S' \otimes^{\mathrm{p}} T' \leq^{\mathrm{p}} S \otimes^{\mathrm{p}} T$.*
2. Preservation of Compatibility: *If $S \rightleftarrows^{\mathrm{p}} T$, $S' \leq^{\mathrm{p}} S$ and $T' \leq^{\mathrm{p}} T$, then $S' \rightleftarrows^{\mathrm{p}} T'$.*

Proof. For better readability we perform the proof for two composable I/O-predicates $P(In_P, Out_P)$ and $Q(In_Q, Out_Q)$ as in Fig. 16 such that $In_P = \{i, x\}$, $Out_P = \{y, o\}$, $In_Q = \{j, y\}$, and $Out_Q = \{x, r\}$. Let $P'(In_P, Out_P) \leq^{\mathrm{p}} P(In_P, Out_P)$ and $Q'(In_Q, Out_Q) \leq^{\mathrm{p}} Q(In_Q, Out_Q)$ be two refinements.

Proof of (1): We have to show

$$P'(In_P, Out_P) \otimes^{\mathrm{p}} Q'(In_Q, Out_Q) \leq^{\mathrm{p}} P(In_P, Out_P) \otimes^{\mathrm{p}} Q(In_Q, Out_Q).$$

Condition (r1) is obviously satisfied. For (r2) we have to show

$$\forall i, j : (\exists o, r : \exists y, x : P \wedge Q) \Rightarrow (\exists o, r : \exists y, x : P' \wedge Q').$$

We know from the assumptions that

$$\forall i, x : (\exists y, o : P) \Rightarrow (\exists y, o : P') \text{ and } \forall j, y : (\exists x, r : Q) \Rightarrow (\exists x, r : Q').$$

This implies easily the claim (r2).

For condition (r3) we have to show

$$\forall i, j, o, r : (\exists y, x : P' \wedge Q') \Rightarrow (\exists y, x : P \wedge Q).$$

We know from the assumptions that

$$\forall i, x, y, o : P' \Rightarrow P \text{ and } \forall j, y, x, r : Q' \Rightarrow Q.$$

This implies easily the claim (r3).

Proof of (2): Assume $P(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q(In_Q, Out_Q)$. We have to show $P'(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q'(In_Q, Out_Q)$. For the I/O-variables we know $Out_P \cap In_Q = \{y\} \neq \emptyset$ and $Out_Q \cap In_P = \{x\} \neq \emptyset$. By symmetry, it is sufficient to show that condition (c1) is satisfied, i.e. we have to prove:

(c1.1) $\forall i, x, y, o : (P' \Rightarrow \exists j, x, r : Q')$, and

(c1.2) $\forall i, x, y, o : (P' \Rightarrow \mathbf{UniqueOut}(Q', \{x\}))$

with $\mathbf{UniqueOut}(Q', \{x\}) = \forall j, x'', r : (Q[x''/x] \Rightarrow x'' = x)$.

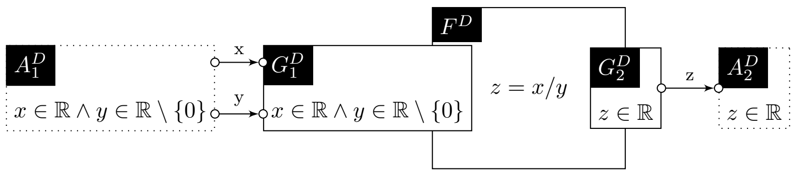
We perform the proof on the semantic level. First we prove (c1.1). Let i, x, y, o be arbitrary elements such that $P'(i, x, y, o)$ holds. We have to show that for the given y, there exist elements j, x'', r such that $Q'(j, y, x'', r)$. Since $P'(In_P, Out_P) \leq^{\mathrm{p}} P(In_P, Out_P)$, we get by (r3) that $P(i, x, y, o)$ holds. From $P(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q(In_Q, Out_Q)$ it follows that for the given y there exist elements j, x'', r such that $Q(j, y, x'', r)$. Let j be such an element. Hence for j, y (which is the input of Q) there exist outputs x'', r of Q such that $Q(j, y, x'', r)$. Then the refinement rule (r2) for $Q'(In_Q, Out_Q) \leq^{\mathrm{p}} Q(In_Q, Out_Q)$ implies that for j, y there exist outputs x'', r of Q' such that $Q'(j, y, x'', r)$. Hence there exist elements j, x'', r such that $Q'(j, y, x'', r)$.

Secondly, we prove (c1.2). Let i, x, y, o be arbitrary elements such that $P'(i, x, y, o)$ holds. We have to show that for the given y and for all elements j, x'', r, if $Q'(j, y, x'', r)$ then $x'' = x$. Let j, x'', r be arbitrary elements with $Q'(j, y, x'', r)$. By refinement rule (r3) for $Q'(In_Q, Out_Q) \leq^{\mathrm{p}} Q(In_Q, Out_Q)$ we get $Q(j, y, x'', r)$. Moreover, by (r3) for $P'(In_P, Out_P) \leq^{\mathrm{p}} P(In_P, Out_P)$ we get $P(i, x, y, o)$. Then, from the assumed compatibility $P(In_P, Out_P) \rightleftarrows^{\mathrm{p}} Q(In_Q, Out_Q)$ it follows that $x'' = x$. □

Theorem 10. *The tuple* $(\mathfrak{S}^{\mathrm{p}}, \otimes^{\mathrm{p}}, \leq^{\mathrm{p}}, \rightleftarrows^{\mathrm{p}}, \mathcal{L}^{\mathrm{p}}, \ell^{\mathrm{p}})$ *is a labeled interface theory.*

As a consequence of this theorem we can lift the theory of I/O-predicates to a theory of component interfaces such that all general results developed in the last sections can be applied.

Example 6. We consider the division of real numbers like in [17]. We have three variables x, y, z which are interpreted in $\mathbb{R} \cup \{\perp\}$. Division by zero yields $\perp$. We specify a component with the interface shown in Fig. 17.



F^D
A_1^D
$x \in \mathbb{R} \wedge y \in \mathbb{R} \setminus \{0\}$
x
y
G_1^D
$x \in \mathbb{R} \wedge y \in \mathbb{R} \setminus \{0\}$
$z = x/y$
G_2^D
$z \in \mathbb{R}$
z
A_2^D
$z \in \mathbb{R}$

Figure 17: Component interface for the division of real numbers

The frame F^D has input variables x, y and the output variable z. The frame predicate says that the component computes the division of x by y and delivers the result on z. The component interface has two ports. The guarantee G_2^D on the right port has the output variable z, indicated on the outgoing arrow, and no input variable. It guarantees that the component will deliver a proper real value (i.e. different from $\perp$). The assumption predicate A_2^D on this port has the input variable z and requires that any environment connected to that port must accept a proper real number as input. Obviously assumption and guarantee are compatible. Hence, the pair (A_2^D, G_2^D) forms a port contract. For the construction of the port contract on the other side we apply the second design method for component interfaces explained in Sect. 6.1. First we compute the guarantee G_1^D by constructing the composition $A_2^D \otimes^{\mathrm{P}} F^D$ which gives us literally the I/O-predicate $(\exists z : z \in \mathbb{R} \wedge z = x/y)(\{x, y\}, \emptyset)$. This I/O-predicate is equivalent to G_1^D which guarantees that the component will accept as inputs on that port any proper real numbers x, y with $y \neq 0$. Then we must design A_1^D in such a way that

$$A_1^D \rightleftarrows^{\mathrm{P}} G_1^D \text{ and } A_1^D \otimes^{\mathrm{P}} F^D \leq^{\mathrm{P}} G_2^D.$$

For A_1^D we take the most liberal I/O-predicate compatible with G_1^D which has the same underlying predicate as G_1^D but inputs and outputs are reversed. A_1^D requires that the environment connected to that port can only deliver (as outputs) proper real values x, y with $y \neq 0$. Finally we have to check that $A_1^D \otimes^{\mathrm{P}} F^D \leq^{\mathrm{P}} G_2^D$. Since both I/O-predicates have no input variables and the same output variable z it is enough to prove the refinement condition (r3) which reads in this case

$$\forall z : (\exists x, y : x \in \mathbb{R} \wedge y \in \mathbb{R} \setminus \{0\} \wedge z = x/y) \Rightarrow z \in \mathbb{R}.$$

Obviously this is true.

One may wonder whether the above component interface specification doesn't look over-complicated. Indeed one could use syntactic sugar to express situations in which assumptions and guarantees consist of the same predicate but with reversed input and output variables. One could also implicitly assume that whenever a guarantee is given on outputs, like in G_2^D, then the environment is assumed to take that output as an input and when an assumption is formulated on outputs of the environment, like in A_1^D, then the component guarantees to take that output as an input. This implicit assumption is usually behind pre/postcondition style specifications who don't explicitly state that a precondition is not only a requirement for the user to provide the right data but also a requirement for the implementor to accept that data, and similarly for postconditions.

8. Conclusion

We have presented a generic framework to construct a theory of component interfaces with port contracts on top of any arbitrary labeled interface theory. A simple extension of interface theories with labels was enough to develop the full theory of component interfaces. We have shown how the approach can be instantiated to obtain modal component interfaces on the basis of modal I/O-transition systems. MIOs are particularly suited for presenting port contracts since they allow to specify loose assumptions utilizing the expressiveness of the modalities. We have also provided methodological guidelines how to construct reliable component interfaces and we have discussed an adaptation scenario.

As a second instantiation of our approach we have considered I/O-predicates which led to component interfaces in the style of pre/postcondition specifications. Of course it is appealing to study more instantiations of our abstract component theory. In principle it will work whenever a concrete framework satisfies the rules of a labeled interface theory. For instance, this would be the case for transition systems with buffered communication and with weak asynchronous compatibility considered in [6], for modal I/O-automata with data constraints ([8, 4]), which combine pre/postconditions on data states with the specification of dynamic behavior, and for modal Petri nets, in which compatibility is expressed by a consumption requirement for the channels of asynchronously composed nets [24]. Another candidate for instantiation would be CSP processes with failures (or failures-divergence) refinement [41] if we define compatibility of processes by deadlock freedom of their parallel composition. Since failures refinement preserves deadlock freedom and is also compositional this would form an appropriate basis for our approach. A candidate from coordination languages are REO connectors [34] equipped with a semantics in terms of UTP designs [26]. Another example could be contract-based components in the sense of [33] with provided and required interfaces for offered and used methods. In this case one could consider predicates on object-oriented state models as a basic interface theory and one could attach to a component one port for each method such that the precondition is the assumption and the postcondition is the guarantee on that port.

For our modal component interface theory we plan to provide tool support by extending the MIO Workbench [9]. This tool is already very useful to check weak modal refinement and weak modal compatibility but it lacks explicit support for the structure of component interfaces with contracts on ports. Another issue concerns a next refinement stage of our approach by introducing a more detailed, but still generic framework that integrates architectural concepts like connectors, assemblies and hierarchies and then to study the applicability to well established architectural design languages like, e.g., Wright [1]. This would be particularly interesting since Wright emphasizes also the role of communication compatibility which there is achieved by relating component ports and connector roles (whose behavior is specified in terms of CSP) by failures-divergence refinement. This idea is close to our compatibility relation between component interfaces which relates port guarantees of one component with the assumptions of the other by refinement (which could be instantiated with failures-divergence refinement as discussed above).

Acknowledgment. We would like to thank the reviewers of previous versions of this work for valuable comments and suggestions.

References

[1] Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, 1997.

[2] Ralph-Johan Back and Joakim von Wright. *Refinement calculus - a systematic introduction*. Undergraduate texts in computer science. Springer, 1999.

[3] Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis. Rigorous component-based system design using the BIP framework. *IEEE Software*, 28(3):41–48, 2011.

[4] Sebastian S. Bauer. *Modal Specification Theories for Component-Based Design*. PhD thesis, Institut für Informatik, Ludwig-Maximilians-Universität München, Germany, 2012.

[5] Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Moving from specifications to contracts in component-based design. In Juan de Lara and Andrea Zisman, editors, *FASE*, volume 7212 of *Lecture Notes in Computer Science*, pages 43–58. Springer, 2012.

[6] Sebastian S. Bauer, Rolf Hennicker, and Stephan Janisch. Interface Theories for (A)synchronously Communicating Modal I/O-Transition Systems. In Axel Legay and Benoît Caillaud, editors, *FIT*, volume 46 of *EPTCS*, pages 1–8, 2010.

[7] Sebastian S. Bauer, Rolf Hennicker, and Axel Legay. Component interfaces with contracts on ports. In Corina S. Pasareanu and Gwen Salaün, editors, *FACS*, volume 7684 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2012.

[8] Sebastian S. Bauer, Rolf Hennicker, and Martin Wirsing. Interface theories for concurrency and data. *Theor. Comput. Sci.*, 412(28):3101–3121, 2011.

[9] Sebastian S. Bauer, Philip Mayer, Andreas Schroeder, and Rolf Hennicker. On weak modal compatibility, refinement, and the MIO Workbench. In *Proc. TACAS 2010*, volume 6015 of *Lect. Notes Comp. Sci.*, pages 175–189. Springer, 2010.

[10] Saddek Bensalem, Marius Bozga, Axel Legay, Thanh-Hung Nguyen, Joseph Sifakis, and Rongjie Yan. Incremental component-based construction and verification using invariants. In *FMCAD*, pages 257–256. IEEE, 2010.

[11] Saddek Bensalem, Marius Bozga, Joseph Sifakis, and Thanh-Hung Nguyen. Compositional verification for component-based systems and application. In *ATVA*, volume 5311 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2008.

[12] Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, and Christos Sofronis. Multiple viewpoint contract-based specification and design. In *FMCO*, volume 5382 of *LNCS*, pages 200–225. Springer, 2007.

[13] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. Constraint markov chains. *Theor. Comput. Sci.*, 412(34):4373–4404, 2011.

[14] Antonio Cau and Pierre Collette. Parallel composition of assumption-commitment specifications: A unifying approach for shared variable and distributed message passing concurrency. *Acta Inf.*, 33(2):153–176, 1996.

[15] Jamieson M. Cobleigh, Dimitra Giannakopoulou, and Corina S. Pasareanu. Learning assumptions for compositional verification. In *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2003.

[16] Luca de Alfaro and Thomas A. Henzinger. Interface automata. *Software Engineering Notes*, pages 109–120, 2001.

[17] Luca de Alfaro and Thomas A. Henzinger. Interface Theories for Component-Based Design. In *EMSOFT 2001*, pages 148–165, 2001.

[18] Luca de Alfaro and Thomas A. Henzinger. Interface-based Design. In Manfred Broy, Johannes Grünbauer, David Harel, and C. A. R. Hoare, editors, *Engineering Theories of Software-intensive Systems*, volume 195 of *NATO Science Series: Mathematics, Physics, and Chemistry*, pages 83–104. Springer, 2005.

[19] Luca de Alfaro, Thomas A. Henzinger, and Marielle I. A. Stoelinga. Timed interfaces. In Alberto L. Sangiovanni-Vincentelli and Joseph Sifakis, editors, *EMSOFT*, volume 2491 of *Lect. Notes Comp. Sci.*, pages 108–122. Springer, 2002.

[20] Edsger W. Dijkstra. Guarded commands, non-determinancy and a calculus for the derivation of programs. In Friedrich L. Bauer and Klaus Samelson, editors, *Language Hierarchies and Interfaces*, volume 46 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 1975.

[21] Michael Emmi, Dimitra Giannakopoulou, and Corina S. Pasareanu. Assume-guarantee verification for interface automata. In Jorge Cuéllar, T. S. E. Maibaum, and Kaisa Sere, editors, *FM*, volume 5014 of *Lecture Notes in Computer Science*, pages 116–131. Springer, 2008.

[22] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of markov decision processes. In *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2007.

[23] Gregor Goessler and Jean-Baptiste Raclet. Modal contracts for component-based design. In *SEFM*, pages 295–303. IEEE Computer Society, 2009.

[24] S. Haddad, R. Hennicker, and M. H. Møller. Specification of asynchronous component sytems with modal I/O-Petri nets. In *Truthworthy Global Computing*, Lecture Notes in Computer Science. Springer, 2013. To appear.

[25] Rolf Hennicker, Stephan Janisch, and Alexander Knapp. On the observable behaviour of composite components. *Electr. Notes Theor. Comput. Sci.*, 260:125–153, 2010.

[26] C.A.R. Hoare and J. He. *Unifying Theories of Programming.* Prentice Hall International, 1998.

[27] Hans Hüttel and Kim Guldstrand Larsen. The Use of Static Constructs in A Modal Process Logic. In *Logic at Botik 1989*, pages 163–180, 1989.

[28] Cliff B. Jones. *Development methods for computer programs including a notion of interference.* PhD thesis, Oxford University Computing Laboratory, 1981.

[29] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-guarantee verification for probabilistic systems. In *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2010.

[30] Leslie Lamport. *win* and *sin*: Predicate transformers for concurrency. *ACM Trans. Program. Lang. Syst.*, 12(3):396–428, 1990.

[31] Kim Guldstrand Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal I/O Automata for Interface and Product Line Theories. In *ESOP 2007*, volume 4421 of *Lect. Notes Comp. Sci.*, pages 64–79.

Springer, 2007.

[32] Kim Guldstrand Larsen and Bent Thomsen. A Modal Process Logic. In *3rd Annual Symp. Logic in Computer Science, LICS 1988*, pages 203–210. IEEE Computer Society, 1988.

[33] Zhiming Liu, Jifeng He, and Xiaoshan Li. Contract oriented development of component software. In Jean-Jacques Lévy, Ernst W. Mayr, and John C. Mitchell, editors, *IFIP TCS*, pages 349–366. Kluwer, 2004.

[34] Sun Meng, Farhad Arbab, Bernhard K. Aichernig, Lacramioara Astefanoaei, Frank S. de Boer, and Jan J. M. M. Rutten. Connectors as designs: Modeling, refinement and test case generation. *Sci. Comput. Program.*, 77(7-8):799–822, 2012.

[35] Bertrand Meyer. Applying "design by contract". *IEEE Computer*, 25(10):40–51, 1992.

[36] Jayadev Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Trans. Software Eng.*, 7(4):417–426, 1981.

[37] Pavel Parizek and Frantisek Plasil. Modeling environment for component model checking from hierarchical architecture. *Electr. Notes Theor. Comput. Sci.*, 182:139–153, 2007.

[38] Frantisek Plasil and Stanislav Visnovsky. Behavior protocols for software components. *IEEE Trans. Software Eng.*, 28(11):1056–1076, 2002.

[39] Sophie Quinton and Susanne Graf. Contract-based verification of hierarchical systems of components. In *SEFM*, pages 377–381. IEEE Computer Society, 2008.

[40] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. A modal interface theory for component-based design. *Fundam. Inform.*, 108(1-2):119–149, 2011.

[41] A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, 1997.

[42] Qiwen Xu, Antonio Cau, and Pierre Collette. On unifying assumption-commitment style proof rules for concurrency. In Bengt Jonsson and Joachim Parrow, editors, *CONCUR*, volume 836 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 1994.