Modal Interface Automata

Gerald Lüttgen, Walter Vogler

Modal Interface Automata

Gerald Lüttgen[1] and Walter Vogler[2]

[1] Software Technologies Group, University of Bamberg, 96045 Bamberg, Germany
gerald.luettgen@swt-bamberg.de
[2] Institute for Computer Science, University of Augsburg, 86135 Augsburg, Germany
vogler@informatik.uni-augsburg.de

Abstract. De Alfaro and Henzinger's Interface Automata (IA) and Nyman et al.'s recent combination IOMTS of IA and Larsen's Modal Transition Systems (MTS) are established frameworks for specifying interfaces of system components. However, neither IA nor IOMTS consider conjunction that is needed in practice when a component satisfies multiple interfaces, while Larsen's MTS-conjunction is not closed. In addition, IOMTS-parallel composition exhibits a compositionality defect.
This paper defines conjunction on IA and MTS and proves the operators to be 'correct', i.e., the greatest lower bounds wrt. IA- and resp. MTS-refinement. As its main contribution, a novel interface theory called Modal Interface Automata (MIA) is introduced: MIA is a rich subset of IOMTS, is equipped with compositional parallel and conjunction operators, and allows a simpler embedding of IA than Nyman's. Thus, it fixes the shortcomings of related work, without restricting designers to deterministic interfaces as Raclet et al.'s modal interface theory does.

1 Introduction

Interfaces play an important role when designing complex software and hardware systems. Early interface theories that deal with types of data and operations only, have recently been extended to also capture protocol aspects of component interaction. One prominent example of such a rich interface theory is de Alfaro and Henzinger's *Interface Automata* (IA) [4, 5], which is based on labelled transition systems (LTS) but distinguishes a component's input and output actions. The theory comes with an asymmetric parallel composition operator, where a component may wait on inputs but never on outputs. Thus, a component output must be consumed immediately, or an error occurs. In case no potential system environment may restrict the system components' behaviour so that all errors are avoided, the components are deemed to be incompatible.

Semantically, IA employs a refinement notion based on an alternating simulation, such that a component satisfies an interface if (a) it implements all input behaviour prescribed by the interface and (b) the interface permits all output behaviour executed by the implementing component. Notably, this means that a component that consumes all inputs but never produces any output satisfies any interface. To be able to mandate output-transitions, Larsen, Nyman

and Wasowski have built their interface theory on Modal Transition Systems (MTS) [7] rather than LTS, which enables one to distinguish between may- and must-transitions and thus to express compulsory outputs. The resulting *IOMTS* interface theory [8], into which IA can be embedded, is equipped with an IA-style parallel composition and an MTS-style modal refinement. Unfortunately, IOMTS-modal refinement has a compositionality defect wrt. parallel composition, i.e., it is not a precongruence for parallel composition; a related result in [8] has already been shown incorrect by Raclet et al. in [12].

The present paper starts from the observation that the above interface theories are missing one important operator, namely conjunction on interfaces. Conjunction is needed in practice since components are often designed to satisfy multiple interfaces simultaneously, each of which specifies a particular aspect of component interaction. We thus start off by recalling the IA-setting and defining a conjunction operator $\wedge$ for IA; we prove that $\wedge$ is indeed conjunction, i.e., the greatest lower bound wrt. alternating simulation (cf. Sec. 2). Similarly, we do so for a slight extension of MTS (a subset of *Disjunctive MTS* [10], cf. Sec. 3), which paves us the way for our main contribution outlined below. Although Larsen has studied conjunction for MTS, his operator does – in contrast to ours – not preserve the MTS-property of syntactic consistency, i.e., a conjunction almost always has some required transitions (must-transitions) that are not allowed (missing may-transitions). An additional difficulty when compared to the IA-setting is that two MTS-interfaces may not have a common implementation; indeed, inconsistencies may arise when composing MTSs conjunctively. We handle inconsistencies by adapting ideas from our prior work on conjunction in a CSP-style process algebra [11] that uses, however, a very different parallel operator and refinement preorder. Note also that our setting employs event-based communication via handshake and thus differs significantly from the one of shared-memory communication studied by Abadi and Lamport in their paper on conjoining specifications [1].

Our paper's main contribution is a novel interface theory, called *Modal Interface Automata* (MIA), which is essentially a rich subset of IOMTS that still allows one to express output-must-transitions. It is equipped with an MTS-style conjunction $\wedge$ and an IOMTS-style parallel composition operator, as well as with a slight adaptation of IOMTS-refinement. We show that (i) MIA-refinement is a precongruence for both operators; (ii) $\wedge$ is indeed conjunction for this preorder; and (iii) IA can be embedded into MIA in a much cleaner, homomorphic fashion than into IOMTS [8] (cf. Sec. 4). Thereby, we remedy the shortcomings of related work while, unlike the language-based modal interface theory of [12], still permitting nondeterminism in interface specifications.

2 Conjunction for Interface Automata

Interface Automata (IA) were introduced by de Alfaro and Henzinger [4, 5] as a *reactive type* theory that abstractly describes the communication behaviour of software or hardware components in terms of their inputs and outputs. IAs are

labelled transition systems where visible actions are partitioned into inputs and outputs. The idea is that interfaces interact with their environment according to the following rules. An interface cannot block an incoming input in any state but, if an input arrives unexpectedly, it is treated as a catastrophic system failure. This means that, if a state does not enable an input, this is a requirement on the environment not to produce this input. Vice versa, an interface guarantees not to produce any unspecified outputs, which are in turn inputs to the environment.

This intuition is reflected in the specific refinement relation of *alternating simulation* between IA and in the *parallel composition* on IA, which have been defined in [5] and are recalled in this section. Most importantly, however, we introduce and study a *conjunction operator* on IA, which is needed in practice to reason about components that are expected to satisfy multiple interfaces.

Definition 1 (Interface Automata [5]). An *Interface Automaton* (IA) is a tuple $Q = (Q, I, O, \longrightarrow)$, where

1. Q is a set of states,
2. I and O are disjoint input and output alphabets, respectively, not containing the special (in contrast to [5] unique), silent action τ,
3. $\longrightarrow \subseteq Q \times (I \cup O \cup \{\tau\}) \times Q$ is the *transition relation.*

The transition relation is required to be *input-deterministic*, i.e., $a \in I$, $q \xrightarrow{a} q'$ and $q \xrightarrow{a} q''$ implies $q' = q''$. In the remainder, we write $q \xrightarrow{a}$ if $q \xrightarrow{a} q'$ for some q', as well as $q \not\xrightarrow{a}$ for its negation.

We let A stand for $I \cup O$, let a (α) range over A ($A \cup \{\tau\}$), and introduce the following weak transition relations: $q \stackrel{\varepsilon}{\Longrightarrow} q'$ if $q(\xrightarrow{\tau})^* q'$, and $q \stackrel{o}{\Longrightarrow} q'$ for $o \in O$ if $\exists q''.\, q \stackrel{\varepsilon}{\Longrightarrow} q'' \xrightarrow{o} q'$; note that there are no τ-transitions after the o-transition. Moreover, we define $\hat{\alpha} = \varepsilon$ if $\alpha = \tau$, and $\hat{\alpha} = \alpha$ otherwise.

Definition 2 (Alternating Simulation [5]). Let P and Q be IAs with common input and output alphabets. Relation $\mathcal{R} \subseteq P \times Q$ is an *alternating simulation relation* if for any $(p, q) \in \mathcal{R}$:

(i) $q \xrightarrow{a} q'$ and $a \in I$ implies $\exists p'.\, p \xrightarrow{a} p'$ and $(p', q') \in \mathcal{R}$,
(ii) $p \xrightarrow{\alpha} p'$ and $\alpha \in O \cup \{\tau\}$ implies $\exists q'.\, q \stackrel{\hat{\alpha}}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{R}$.

We write $p \sqsubseteq_{\mathrm{IA}} q$ and say that p *IA-refines* q if there exists an alternating simulation relation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$.

According to the basic idea of IA, if specification Q in state q allows some input a delivered by the environment, then the related implementation state p of P must allow this input immediately in order to avoid system failure. Conversely, if P in state p produces output a to be consumed by the environment, this output must be expected by the environment even if $q \stackrel{a}{\Longrightarrow}$; this is because Q could have moved unobservedly from state q to some q' that enables a. Since inputs are not treated in Def. 2 (ii), they are always allowed for p.

It is easy to see that IA-refinement $\sqsubseteq_{\mathrm{IA}}$ is a preorder on IA. Given input and output alphabets I and O, respectively, the IA $\mathit{BlackHole}_{I,O} =_{\mathrm{df}} (\{\mathit{blackhole}\}, I, O, \{(\mathit{blackhole}, a, \mathit{blackhole}) \mid a \in I\})$ IA-refines any other IA over I and O.

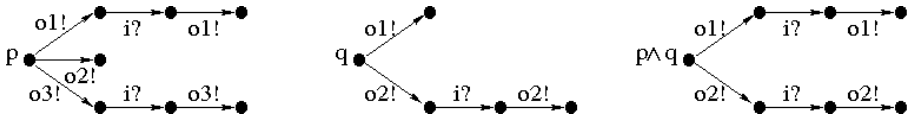

Fig. 1. Example illustrating IA-conjunction.

2.1 Conjunction on IA

Two IAs with common alphabets are always logically consistent in the sense that they have a common implementation, e.g., the respective blackhole IA as noted above. This makes the definition of conjunction on IA relatively straightforward. Here and similarly later, we index a transition by the system's name to make clear from where it originates, in case this is not obvious from the context.

Definition 3 (Conjunction on IA). Let $P = (P, I, O, \longrightarrow_P)$ and $Q = (Q, I, O, \longrightarrow_Q)$ be IAs with common input and output alphabets and disjoint state sets P and Q. The conjunction $P \wedge Q$ is defined by $(\{p \wedge q \,|\, p \in P,\, q \in Q\} \cup P \cup Q, I, O, \longrightarrow)$, where $\longrightarrow$ is the least set satisfying $\longrightarrow_P \subseteq \longrightarrow$, $\longrightarrow_Q \subseteq \longrightarrow$ and the following operational rules:

(I1)	$p \wedge q \xrightarrow{a} p'$	if	$p \xrightarrow{a}_P p'$, $q \not\xrightarrow{a}_Q$ and $a \in I$
(I2)	$p \wedge q \xrightarrow{a} q'$	if	$p \not\xrightarrow{a}_P$, $q \xrightarrow{a}_Q q'$ and $a \in I$
(I3)	$p \wedge q \xrightarrow{a} p' \wedge q'$	if	$p \xrightarrow{a}_P p'$, $q \xrightarrow{a}_Q q'$ and $a \in I$
(O)	$p \wedge q \xrightarrow{a} p' \wedge q'$	if	$p \xrightarrow{a}_P p'$, $q \xrightarrow{a}_Q q'$ and $a \in O$
(T1)	$p \wedge q \xrightarrow{\tau} p' \wedge q$	if	$p \xrightarrow{\tau}_P p'$
(T2)	$p \wedge q \xrightarrow{\tau} p \wedge q'$	if	$q \xrightarrow{\tau}_Q q'$

Intuitively, conjunction is the synchronous product over actions (cf. Rules (I3), (O), (T1) and (T2)). Since inputs are always implicitly present, this also explains Rules (I1) and (I2); for example, in Rule (I1), q does not impose any restrictions on the behaviour after input a and is therefore dropped from the target state. Moreover, the conjunction operator is commutative and associative. As an aside, note that the rules with digit 2 in their names are the symmetric cases of the respective rules with digit 1; this convention will hold true throughout this paper. Fig. 1 applies the rules above to an illustrating example; here and in the following figures, we write $a?$ for an input a and $a!$ for an output a.

Theorem 4 ($\wedge$ is And). *Let P, Q, R be IAs with states p, q and r, respectively. Then, $r \sqsubseteq_{IA} p$ and $r \sqsubseteq_{IA} q$ if and only if $r \sqsubseteq_{IA} p \wedge q$.*

Hence, $\wedge$ gives the greatest lower-bound wrt. $\sqsubseteq_{\mathrm{IA}}$, i.e., an implementation satisfies the conjunction of interfaces exactly if it satisfies each of them. This is a desired property in system design where each interface describes one aspect of the overall specification. The above theorem also implies compositional reasoning; from universal algebra one easily gets:

Corollary 5. For IAs P, Q, R with states p, q and r: $p \sqsubseteq_{\mathrm{IA}} q \implies p \wedge r \sqsubseteq_{\mathrm{IA}} q \wedge r$.

2.2 Parallel Composition on IA

We recall the parallel composition operator $|$ on IA of [5], which is defined in two stages: first a standard product $\otimes$ between two IAs is introduced, where common actions are synchronized and hidden. Then, error states are identified, and all states are pruned from which reaching an error state is unavoidable.

Definition 6 (Parallel Product on IA [5]). IAs P_1, P_2 are *composable* if $A_1 \cap A_2 = (I_1 \cap O_2) \cup (O_1 \cap I_2)$, i.e., each common action is input of one IA and output of the other IA. For such IAs we define the *product* $P_1 \otimes P_2 = (P_1 \times P_2, I, O, \longrightarrow)$, where $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$ and $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2)$ and where $\longrightarrow$ is given by the following operational rules:

(Par1) $(p_1, p_2) \xrightarrow{\alpha} (p_1', p_2)$ if $p_1 \xrightarrow{\alpha} p_1'$ and $\alpha \notin A_2$

(Par2) $(p_1, p_2) \xrightarrow{\alpha} (p_1, p_2')$ if $p_2 \xrightarrow{\alpha} p_2'$ and $\alpha \notin A_1$

(Par3) $(p_1, p_2) \xrightarrow{\tau} (p_1', p_2')$ if $p_1 \xrightarrow{a} p_1'$ and $p_2 \xrightarrow{a} p_2'$ for some a.

Note that, in case of synchronization and according to Rule (Par3), one only gets internal τ-transitions.

Definition 7 (Parallel Composition on IA [5]). A state (p_1, p_2) of a parallel product $P_1 \otimes P_2$ is an *error state* if there is some $a \in A_1 \cap A_2$ such that (a) $a \in O_1$, $p_1 \xrightarrow{a}$ and $p_2 \not\xrightarrow{a}$, or (b) $a \in O_2$, $p_2 \xrightarrow{a}$ and $p_1 \not\xrightarrow{a}$.

A state of $P_1 \otimes P_2$ is *incompatible* if it may reach an error state autonomously, i.e., only by output or internal actions that are, intuitively, locally controlled. Formally, the set $E \subseteq P_1 \times P_2$ of incompatible states is the least set such that $(p_1, p_2) \in E$ if (i) (p_1, p_2) is an error state or (ii) $(p_1, p_2) \xrightarrow{\alpha} (p_1', p_2')$ for some $\alpha \in O \cup \{\tau\}$ and $(p_1', p_2') \in E$.

The *parallel composition* $P_1|P_2$ of P_1, P_2 is obtained from $P_1 \otimes P_2$ by *pruning*, i.e., removing all states in E and all transitions involving such states as source or target. If $(p_1, p_2) \in P_1|P_2$, we write $p_1|p_2$ and call p_1 and p_2 *compatible*.

Parallel composition is well-defined since input-determinism is preserved.

Theorem 8 (Compositionality of IA-Parallel Composition [5]). *Let P_1, P_2 and Q_1 be IAs with $p_1 \in P_1$, $p_2 \in P_2$, $q_1 \in Q_1$ and $p_1 \sqsubseteq_{IA} q_1$. Assume that Q_1 and P_2 are composable; then, (a) P_1 and P_2 are composable and (b) if q_1 and p_2 are compatible, then so are p_1 and p_2 and $p_1|p_2 \sqsubseteq_{IA} q_1|p_2$.*

This result relies on the fact that IAs are input-deterministic. While the theorem is already stated in [5], its proof is only sketched therein. Here, it is a simple corollary of Thms. 23 in Sec. 4.2 and Thms. 25 and 26(b) in Sec. 4.3 below.

We conclude by presenting a small example of IA-parallel composition in Fig. 2, which is adapted from [5]. The client does not accept its input *retry*. Thus, if the environment of *Client* $\otimes$ *TryOnce* would produce *nack*, the system would autonomously produce *reset* and run into a catastrophic error. To avoid this, the environment of *Client*$|$*TryOnce* is required not to produce *nack*. This view is called optimistic: there exists an environment in which *Client* and *TryOnce* can cooperate without errors, and *Client*$|$*TryOnce* describes the necessary requirements for such an environment. In the pessimistic view as advocated in [2], *Client* and *TryOnce* are regarded as incompatible due to the potential error.

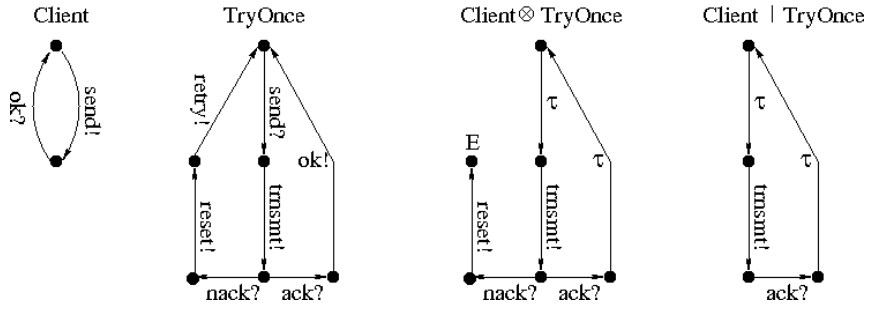

Fig. 2. Example illustrating IA-parallel composition, where IA *TryOnce* has inputs {*send, ack, nack*} and outputs {*trnsmt, ok, reset, retry*}, while IA *Client* has inputs {*ok, retry*} and outputs {*send*}.

3 Conjunction for Modal Transition Systems

Modal Transition Systems (MTS) were investigated by Larsen [7] as a specification framework based on labelled transition systems but with two kinds of transitions: must-transitions specify required behaviour, may-transitions specify allowed behavior, and absent transitions specify forbidden behaviour. Any refinement of an MTS-specification must preserve required and forbidden behaviour and may turn allowed behaviour into required or forbidden behaviour. Technically, this is achieved via an alternating-style simulation relation, called *modal refinement*, where any must-transition of the specification must be simulated by an implementation, while any may-transition of the implementation must be simulated by the specification.

Larsen [7] defined conjunction on MTS, but the resulting systems often violate syntactic consistency and are hard to understand. To improve this, we allow an a-must-transition to have several alternative target states, i.e., we work with Disjunctive MTS (DMTS). Larsen and Xinxin also generalized Larsen's construction to DMTS [10], but again ignoring syntactic consistency. We will thus define conjunction on a syntactically consistent subclass of DMTS, called *dMTS*, but more generally in a setting with internal τ-actions as defined in [5, 8].

3.1 Disjunctive Modal Transition Systems

We extend standard MTS only as far as needed for defining conjunction, by introducing disjunctive must-transitions that are disjunctive wrt. exit states only (see Fig. 4). The following extension also has no τ-must-transitions since these are not considered in the definition of the observational modal refinement of [8].

Definition 9 (disjunctive Modal Transition System). A *disjunctive Modal Transition System* (dMTS) is a tuple $Q = (Q, A, \longrightarrow, \dashrightarrow)$, where

1. Q is a set of states,

2. A is an alphabet not containing the special, silent action τ,
3. $\longrightarrow \subseteq Q \times A \times (\mathcal{P}(Q) \setminus \emptyset)$ is the *must-transition* relation,
4. $\dashrightarrow \subseteq Q \times (A \cup \{\tau\}) \times Q$ is the *may-transition* relation.

We require *syntactic consistency*, i.e., $q \xrightarrow{a} Q'$ implies $\forall q' \in Q'.\, q \overset{a}{\dashrightarrow} q'$.

More generally, the must-transition relation in a standard DMTS [10] may be a subset of $Q \times (\mathcal{P}(A \times Q) \setminus \emptyset)$. For notational convenience, we write $q \xrightarrow{a} q'$ whenever $q \xrightarrow{a} \{q'\}$; all must-transitions in standard MTS have this form.

Our refinement relation on dMTS abstracts from internal computation steps in the same way as [8], i.e., by considering the following *weak may-transitions* for $\alpha \in A \cup \{\tau\}$: $q \overset{\varepsilon}{=\!=\!\!\blacktriangleright} q'$ if $q \overset{\tau}{\dashrightarrow}^{*} q'$, and $q \overset{\alpha}{=\!=\!\!\blacktriangleright} q'$ if $\exists q''.\, q \overset{\varepsilon}{=\!=\!\!\blacktriangleright} q'' \overset{\alpha}{\dashrightarrow} q'$.

Definition 10 (Observational Modal Refinement, see [8]). Let P, Q be dMTSs with common alphabet. Relation $\mathcal{R} \subseteq P \times Q$ is an *(observational) modal refinement relation* if for any $(p, q) \in \mathcal{R}$:

(i) $q \xrightarrow{a} Q'$ implies $\exists P'.\, p \xrightarrow{a} P'$ and $\forall p' \in P'\, \exists q' \in Q'.\, (p', q') \in \mathcal{R}$,
(ii) $p \overset{\alpha}{\dashrightarrow} p'$ implies $\exists q'.\, q \overset{\hat{\alpha}}{=\!=\!\!\blacktriangleright} q'$ and $(p', q') \in \mathcal{R}$.

We write $p \sqsubseteq_{\mathrm{dMTS}} q$ and say that p *dMTS-refines* q if there exists an observational modal refinement relation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$.

Except for disjunctiveness, dMTS-refinement is exactly defined as for MTS in [8], i.e., the τ-must-transitions allowed in their variant of MTS are not treated in Cond. (i) of observational modal refinement. Thus, they are treated as only may-transitions and not included in our setting.

3.2 Conjunction on dMTS

Similarly to parallel composition for IA, conjunction will be defined in two stages. State pairs can be logically inconsistent due to unsatisfiable must-transitions; in the second stage, we remove such pairs incrementally.

Definition 11 (Conjunctive Product on dMTS). Let $P = (P, A, \longrightarrow_P, \dashrightarrow_P)$ and $Q = (Q, A, \longrightarrow_Q, \dashrightarrow_Q)$ be dMTSs with common alphabet. The conjunctive product $P \& Q =_{\mathrm{df}} (P \times Q, A, \longrightarrow, \dashrightarrow)$ is defined by its operational transition rules as follows:

(Must1)	$(p, q) \xrightarrow{a} \{(p', q') \mid p' \in P',\, q \overset{a}{=\!=\!\!\blacktriangleright}_Q q'\}$	if $p \xrightarrow{a}_P P'$ and $q \overset{a}{=\!=\!\!\blacktriangleright}_Q$
(Must2)	$(p, q) \xrightarrow{a} \{(p', q') \mid p \overset{a}{=\!=\!\!\blacktriangleright}_P p',\, q' \in Q'\}$	if $p \overset{a}{=\!=\!\!\blacktriangleright}_P$ and $q \xrightarrow{a}_Q Q'$
(May1)	$(p, q) \overset{\tau}{\dashrightarrow} (p', q)$	if $p \overset{\tau}{=\!=\!\!\blacktriangleright}_P p'$
(May2)	$(p, q) \overset{\tau}{\dashrightarrow} (p, q')$	if $q \overset{\tau}{=\!=\!\!\blacktriangleright}_Q q'$
(May3)	$(p, q) \overset{\alpha}{\dashrightarrow} (p', q')$	if $p \overset{\alpha}{=\!=\!\!\blacktriangleright}_P p'$ and $q \overset{\alpha}{=\!=\!\!\blacktriangleright}_Q q'$

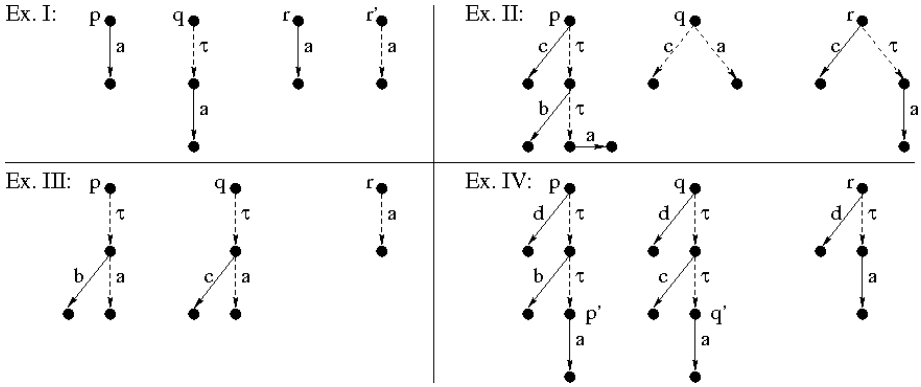

Fig. 3. Examples motivating the rules of Def. 11.

It might be surprising that a single transition in the product might stem from a transition sequence in one of the components (cf. the first four items above) and that the components can also synchronize on τ (cf. Rule (May3)). The necessity of this is discussed below; we only note here that conjunction is inherently different from parallel composition.

Definition 12 (Conjunction on dMTS). Given a conjunctive product $P\&Q$, the set $F \subseteq P \times Q$ of *(logically) inconsistent states* is defined as the least set satisfying the following rules:

(F1)	$p \xrightarrow{a}_P,\ q \not\xrightarrow{a}_Q$	implies	$(p,q) \in F$
(F2)	$p \not\xrightarrow{a}_P,\ q \xrightarrow{a}_Q$	implies	$(p,q) \in F$
(F3)	$(p,q) \xrightarrow{a} R'$ and $R' \subseteq F$	implies	$(p,q) \in F$

The conjunction $P \wedge Q$ of dMTSs P, Q is obtained by deleting all states $(p,q) \in F$ from $P\&Q$. This also removes any may- or must-transition exiting a deleted state and any may-transition entering a deleted state; in addition, deleted states are removed from targets of disjunctive must-transitions. We write $p \wedge q$ for the state (p,q) of $P \wedge Q$; these are the consistent states by construction, and $p \wedge q$ is only defined for such a state.

Regarding well-definedness, first observe that $P\&Q$ is a dMTS, where syntactic consistency follows from Rule (May3). Now, $P \wedge Q$ is a dMTS, too: if R' becomes empty for some $(p,q) \xrightarrow{a} R'$, then also (p,q) is deleted when constructing $P \wedge Q$ from $P\&Q$ according to (F3).

Before we formally state that operator $\wedge$ is indeed conjunction on dMTS, we present several examples depicted in Fig. 3, which motivate the rules of Def. 11. Note that, in this figure and the following figures, any (disjunctive) must-transition drawn also represents implicitly the respective may-transition(s), unless stated otherwise. In each example in Fig. 3, r is a common implementation of p and q (but not r' in Ex. I), whence these must be logically consistent.

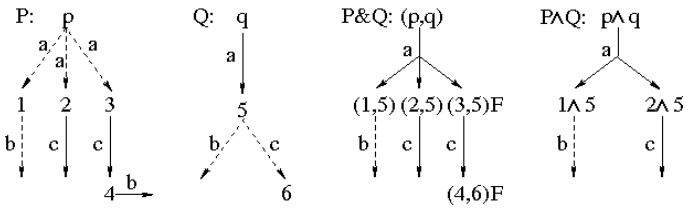

Fig. 4. Example illustrating dMTS-conjunction.

Thus, Ex. I explains Rule (Must1). If we only had $\overset{\tau}{\dashrightarrow}$ in the precondition of Rule (May1), $p \wedge q$ of Ex. II would just consist of a c-must- and an a-may-transition; the only τ-transition would lead to a state in F due to b. This would not allow the τ-transition of r, explaining Rule (May1). In Ex. III and with only $\overset{\alpha}{\dashrightarrow}$ in the preconditions of Rule (May3), $p \wedge q$ would just have three τ-transitions to inconsistent states (due to b, c, respectively). This explains the weak transitions for $\alpha \neq \tau$ in Rule (May3). According to Rules (May1) and (May2), $p \wedge q$ in Ex. IV has four τ-transitions to states in F (due to d). With preconditions based on at least one $\overset{\tau}{\dashrightarrow}$ instead of $\overset{\tau}{=\!=\!\twoheadrightarrow}$ in the τ-case of Rule (May3), there would be three more τ-transitions to states in F (due to b or c). Thus, it is essential that Rule (May3) also allows the synchronization of two weak τ-transitions, which in this case gives $p \wedge q \overset{\tau}{\dashrightarrow} p' \wedge q'$.

Fig. 4 shows a small example illustrating the treatment of disjunctive must-transitions in the presence of inconsistency. In $P\&Q$, the a-must-transition of Q combines with the three a-transitions of P to a truly disjunctive must-transition with a three-element target set. The inconsistency of state $(4, 6)$ due to b propagates back to state $(3, 5)$. The inconsistent states are removed in $P \wedge Q$.

Theorem 13 ($\wedge$ is And). *Let P, Q, R be dMTSs. Then, (i) $(\exists r \in R.\, r \sqsubseteq_{dMTS} p$ and $r \sqsubseteq_{dMTS} q)$ if and only if $p \wedge q$ is defined. Further, in case $p \wedge q$ is defined: (ii) $r \sqsubseteq_{dMTS} p$ and $r \sqsubseteq_{dMTS} q$ if and only if $r \sqsubseteq_{dMTS} p \wedge q$.*

This key theorem states in Item (ii) that conjunction behaves as it should, i.e., $\wedge$ on dMTSs is the greatest lower bound wrt. $\sqsubseteq_{\text{dMTS}}$. Item (i) concerns the intuition that two specifications p and q are logically inconsistent if they do not have a common implementation; formally, $p \wedge q$ is undefined in this case. Alternatively, we could have added an explicit inconsistent element *ff* to our setting, so that $p \wedge q = ff$. This element *ff* would be defined to be a refinement of every p' and equivalent to any $(p', q') \in F$ of some $P\&Q$. Additionally, $ff \wedge p'$ and $p' \wedge ff$ would be defined as *ff*, for any p'. The following corollary of Thm. 13 now follows from universal algebra, as above:

Corollary 14. dMTS-refinement is compositional wrt. conjunction.

Thus, we have succeeded in our ambition to define a syntactically consistent conjunction for MTS, albeit for an MTS variant with disjunctive must-transitions.

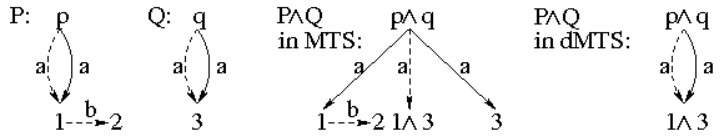

Fig. 5. Example illustrating Larsen's MTS-conjunction; $\overset{a}{\dashrightarrow}$ drawn separately.

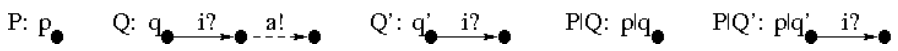

Fig. 6. Example demonstrating the compositionality flaw of IOMTS.

Larsen [7] also defines a conjunction operator on MTS, but almost always the result violates syntactic consistency. A simple example is shown in Fig. 5, where q refines p in Larsen's setting as well as in our dMTS-setting. Since Larsen's $p \wedge q$ is not syntactically consistent, this $p \wedge q$ and q are, contrary to the first impression, equivalent. In our dMTS-setting $P \wedge Q$ is isomorphic to Q, which will also hold for our MIA-setting below (with action b read as output and where a could be either an input or an output).

The above shortcoming has been avoided by Larsen et al. in [9] by limiting conjunction to so-called *independent* specifications that make inconsistencies obsolete. This restriction also excludes the above example. Another MTS-inspired theory including a conjunction operator has been introduced by Raclet et al. [12]. While their approach yields the desired $p \wedge q$ as in our dMTS-setting, it is language-based and thus deals with deterministic systems only.

4 Modal Interface Automata

An essential point of Larsen, Nyman and Wasowski's seminal paper [8] is to enrich IA with modalities to get a flexible specification framework where inputs and outputs can be prescribed, allowed or prohibited. To do so, they consider IOMTS, i.e., MTS where visible actions are partitioned into inputs and outputs, and define parallel composition in IA-style.

Our example of Fig. 6 shows that their approach has a serious flaw, namely observational modal refinement is not a precongruence for the parallel composition of [8]. In this example, the IOMTS P has input alphabet $\{a\}$ and empty output alphabet, while Q and Q' have input alphabet $\{i\}$ and output alphabet $\{a\}$. Obviously, $q' \sqsubseteq_{\mathrm{dMTS}} q$. When composing P and Q in parallel, $p|q$ would reach an error state after an i-must-transition in [8] since the potential output a of Q is not expected by P. In contrast, $p|q'$ has an i-must- and i-may-transition not allowed by $P|Q$, so that $p|q' \not\sqsubseteq_{\mathrm{dMTS}} p|q$. This counterexample also holds for (strong) modal refinement as defined in [8] and is particularly severe since all systems are deterministic. The problem is that $p|q$ forbids input i.

In [8], precongruence of parallel composition is not mentioned. Instead, a theorem relates the parallel composition of two IOMTSs to a different composition

on two refining implementations, where an implementation in [8] is an IOMTS in which may- and must-transitions coincide. This theorem is incorrect as is pointed out in [12] and repaired in the deterministic setting of that paper; the repair is still not a precongruence result, but compares the results of two different operators. However, a natural solution to the precongruence problem can be adopted from the IA-framework [5] where inputs are always allowed implicitly. Consequently, if an input is specified, it will always be a must.

In the remainder, we thus define and study a new specification framework, called *Modal Interface Automata* (MIA), that takes the dMTS-setting for an alphabet consisting of input and output actions, requires input-determinism, and demands that every input-may-transition is also an input-must-transition. The advantage over IA is that outputs can be prescribed via output-must-transitions, which precludes trivial implementations like *BlackHole* discussed in Sec. 2.

Definition 15 (Modal Interface Automaton). A *Modal Interface Automaton* (MIA) is a tuple $Q = (Q, I, O, \longrightarrow, \dashrightarrow)$, where $(Q, I \cup O, \longrightarrow, \dashrightarrow)$ is a dMTS with disjoint alphabets I and O for inputs and outputs and where for all $i \in I$: (a) $q \overset{i}{\dashrightarrow} q'$ and $q \overset{i}{\dashrightarrow} q''$ implies $q' = q''$; (b) $q \overset{i}{\dashrightarrow} q'$ implies $q \overset{i}{\longrightarrow} q'$.

Observe that syntactic consistency and input determinism imply that input-must-transitions always have the form $q \overset{i}{\longrightarrow} \{q'\}$. Thus, only output-must-transitions can be truly disjunctive.

Definition 16 (MIA-Refinement). Let P, Q be MIAs with common input and output alphabets. Relation $\mathcal{R} \subseteq P \times Q$ is an *(observational) MIA-refinement relation* if for any $(p, q) \in \mathcal{R}$:

(i) $q \overset{a}{\longrightarrow} Q'$ implies $\exists P'.\, p \overset{a}{\longrightarrow} P'$ and $\forall p' \in P'\, \exists q' \in Q'.\, (p', q') \in \mathcal{R}$,
(ii) $p \overset{\alpha}{\dashrightarrow} p'$ with $\alpha \in O \cup \{\tau\}$ implies $\exists q'.\, q \overset{\hat{\alpha}}{=\!=\!\twoheadrightarrow} q'$ and $(p', q') \in \mathcal{R}$.

We write $p \sqsubseteq_{\text{MIA}} q$ and say that p *MIA-refines* q if there exists an observational MIA-refinement relation $\mathcal{R}$ such that $(p, q) \in \mathcal{R}$. Moreover, we also write $p \sqsupseteq\!\sqsubseteq_{\text{MIA}} q$ in case $p \sqsubseteq_{\text{MIA}} q$ and $q \sqsubseteq_{\text{MIA}} p$ (which is an equivalence weaker than 'bisimulation').

One can easily check that $\sqsubseteq_{\text{MIA}}$ is a preorder and the largest observational MIA-refinement relation. Its definition coincides with dMTS-refinement except that Cond. (ii) is restricted to outputs and the silent action τ. Thus, inputs are always allowed implicitly and, in effect, treated just like in IA-refinement. Due to the output-must-transitions in the MIA-setting, MIA-refinement can model, e.g., STG-bisimilarity [13] for digital circuits.

4.1 Conjunction on MIA

Similar to conjunction on dMTS, we define conjunction on MIA by first constructing a conjunctive product and then eliminating all inconsistent states.

Definition 17 (Conjunctive Product on MIA). Let $P = (P, I, O, \longrightarrow_P, \dashrightarrow_P)$ and $Q = (Q, I, O, \longrightarrow_Q, \dashrightarrow_Q)$ be MIAs with common input and output alphabets and disjoint state sets P and Q. The conjunctive product $P\&Q =_{\text{df}} ((P \times Q) \cup P \cup Q, I, O, \longrightarrow, \dashrightarrow)$ inherits the transitions of P and Q and has additional transitions as follows, where $i \in I$, $o \in O$ and $\alpha \in O \cup \{\tau\}$:

(OMust1)	$(p,q) \xrightarrow{o} \{(p',q') \mid p' \in P',\ q \overset{o}{=\!=\!\not\rightarrow}_Q q'\}$	if	$p \xrightarrow{o}_P P'$ and $q \overset{o}{=\!=\!\not\rightarrow}_Q$
(OMust2)	$(p,q) \xrightarrow{o} \{(p',q') \mid p \overset{o}{=\!=\!\not\rightarrow}_P p',\ q' \in Q'\}$	if	$p \overset{o}{=\!=\!\not\rightarrow}_P$ and $q \xrightarrow{o}_Q Q'$
(IMust1)	$(p,q) \xrightarrow{i} p'$	if	$p \xrightarrow{i}_P p'$ and $q \not\xrightarrow{i}_Q$
(IMust2)	$(p,q) \xrightarrow{i} q'$	if	$p \not\xrightarrow{i}_P$ and $q \xrightarrow{i}_Q q'$
(IMust3)	$(p,q) \xrightarrow{i} (p',q')$	if	$p \xrightarrow{i}_P p'$ and $q \xrightarrow{i}_Q q'$
(May1)	$(p,q) \overset{\tau}{\dashrightarrow} (p',q)$	if	$p \overset{\tau}{=\!=\!\not\rightarrow}_P p'$
(May2)	$(p,q) \overset{\tau}{\dashrightarrow} (p,q')$	if	$q \overset{\tau}{=\!=\!\not\rightarrow}_Q q'$
(May3)	$(p,q) \overset{\alpha}{\dashrightarrow} (p,q')$	if	$p \overset{\alpha}{=\!=\!\not\rightarrow}_P p'$ and $q \overset{\alpha}{=\!=\!\not\rightarrow}_Q q'$

(Plus the may-rules corresponding to Rules (IMust1)–(IMust3) above.)

This product is defined analogously to IA-conjunction for inputs (plus the corresponding 'may' rules) and to the dMTS-product for outputs and τ. It thus combines the effects shown in Fig. 1 (where all outputs are treated as may) and Fig. 4 (where all actions are outputs).

Definition 18 (Conjunction on MIA). Given a conjunctive product $P\&Q$, the set $F \subseteq P \times Q$ of (logically) *inconsistent states* is defined as the least set satisfying the following rules:

(F1)	$p \xrightarrow{o}_P,\ q \overset{o}{=\!=\!\not\rightarrow}_Q,\ o \in O$	implies	$(p,q) \in F$
(F2)	$p \overset{o}{=\!=\!\not\rightarrow}_P,\ q \xrightarrow{o}_Q,\ o \in O$	implies	$(p,q) \in F$
(F3)	$(p,q) \xrightarrow{a} R'$ and $R' \subseteq F$	implies	$(p,q) \in F$

The conjunction $P \wedge Q$ of MIAs P, Q with common input and output alphabets is obtained by deleting all states $(p,q) \in F$ from $P\&Q$. We write $p \wedge q$ for state (p,q) of $P \wedge Q$; all such states are defined – and consistent – by construction.

The conjunction $P \wedge Q$ is a MIA and is thus well-defined. This can be seen by a similar argument as we have used above in the context of dMTS-conjunction, while input-determinism can be established by an argument similar to that in the IA-setting. Note that, in contrast to the dMTS situation, Rules (F1) and (F2) only apply to outputs. Fig. 4 is also an example for conjunction in the MIA-setting if all actions are read as outputs.

Theorem 19 ($\wedge$ is And). *Let P, Q, R be MIAs. We have (i) $(\exists r \in R.\, r \sqsubseteq_{MIA} p$ and $r \sqsubseteq_{MIA} q)$ if and only if $p \wedge q$ is defined. Further, in case $p \wedge q$ is defined: (ii) $r \sqsubseteq_{MIA} p$ and $r \sqsubseteq_{MIA} q$ if and only if $r \sqsubseteq_{MIA} p \wedge q$.*

Corollary 20. MIA-refinement is compositional wrt. conjunction.

4.2 Parallel Composition on MIA

In analogy to the IA-setting [5], we provide a parallel operator on MIA. Here, error states are identified, and all states are removed from which reaching an error state is unavoidable in some implementation, as is done for IOMTS in [8].

Definition 21 (Parallel Product on MIA). MIAs P_1, P_2 are *composable* if $A_1 \cap A_2 = (I_1 \cap O_2) \cup (O_1 \cap I_2)$, as in IA. For such MIAs we define the *product* $P_1 \otimes P_2 = (P_1 \times P_2, I, O, \longrightarrow, \dashrightarrow)$, where $I = (I_1 \cup I_2) \setminus (O_1 \cup O_2)$ and $O = (O_1 \cup O_2) \setminus (I_1 \cup I_2)$ and where $\longrightarrow$ and $\dashrightarrow$ are defined as follows:

(Must1)	$(p_1, p_2) \xrightarrow{a} P_1' \times \{p_2\}$	if	$p_1 \xrightarrow{a} P_1'$ and $a \notin A_2$
(Must2)	$(p_1, p_2) \xrightarrow{a} \{p_1\} \times P_2'$	if	$p_2 \xrightarrow{a} P_2'$ and $a \notin A_1$
(May1)	$(p_1, p_2) \overset{\alpha}{\dashrightarrow} (p_1', p_2)$	if	$p_1 \overset{\alpha}{\dashrightarrow} p_1'$ and $\alpha \notin A_2$
(May2)	$(p_1, p_2) \overset{\alpha}{\dashrightarrow} (p_1, p_2')$	if	$p_2 \overset{\alpha}{\dashrightarrow} p_2'$ and $\alpha \notin A_1$
(May3)	$(p_1, p_2) \overset{\tau}{\dashrightarrow} (p_1', p_2')$	if	$p_1 \overset{a}{\dashrightarrow} p_1'$ and $p_2 \overset{a}{\dashrightarrow} p_2'$ for some a.

Recall that there are no τ-must-transitions as they are irrelevant for refinement.

Definition 22 (Parallel Composition on MIA). Given a parallel product $P_1 \otimes P_2$, a state (p_1, p_2) is an *error state* if there is some $a \in A_1 \cap A_2$ such that (a) $a \in O_1$, $p_1 \overset{a}{\dashrightarrow}$ and $p_2 \overset{a}{\not\longrightarrow}$, or (b) $a \in O_2$, $p_2 \overset{a}{\dashrightarrow}$ and $p_1 \overset{a}{\not\longrightarrow}$.

Again we define the set $E \subseteq P_1 \times P_2$ of *incompatible* states as the least set such that $(p_1, p_2) \in E$ if (i) (p_1, p_2) is an error state or (ii) $(p_1, p_2) \overset{\alpha}{\dashrightarrow} (p_1', p_2')$ for some $\alpha \in O \cup \{\tau\}$ and $(p_1', p_2') \in E$. The *parallel composition* $P_1|P_2$ of P_1 and P_2 is now obtained from $P_1 \otimes P_2$ by *pruning*, as in IA.

Parallel products and parallel compositions are well-defined MIAs. Syntactic consistency is preserved, as is input-determinism since input-transitions are directly inherited from one of the *composable* systems. In addition, targets of disjunctive must-transitions are never empty since all must-transitions that remain after pruning are taken from the product without modification.

Observe that pruning is different from removing inconsistent states in conjunction. For truly disjunctive transitions $(p_1, p_2) \xrightarrow{a} P'$ (i.e., $a \in O$) of the product $P_1 \otimes P_2$, the state (p_1, p_2) is removed if $P' \cap E \neq \emptyset$. Technically, this follows from syntactic consistency and Cond. (ii) above. Intuitively, this is because P' has w.l.o.g. the form $P_1' \times \{p_2\}$ in the product of P_1 and P_2, with some $(p_1', p_2') \in P' \cap E$; the implementor of P_1 might choose to implement $p_1 \xrightarrow{a} p_1'$ such that – when P_1's implementation is composed with P_2's – the error state is reached. This cannot be reasonably prevented by altering the above definition while preserving the precongruence property for parallel composition:

Theorem 23 (Compositionality of MIA-Parallel Composition). *Let P_1, P_2 and Q_1 be MIAs with $p_1 \in P_1$, $p_2 \in P_2$, $q_1 \in Q_1$ and $p_1 \sqsubseteq_{MIA} q_1$. Assume that Q_1 and P_2 are composable; then, (a) P_1 and P_2 are composable and (b) if q_1 and p_2 are compatible, then so are p_1 and p_2 and $p_1|p_2 \sqsubseteq_{MIA} q_1|p_2$.*

Fig. 7. Example illustrating the need of input-determinism for MIA.

This precongruence property of MIA-refinement would not hold if we would do away with input-determinism in MIA. To see this, consider the example of Fig. 7 for which $p \sqsubseteq_{\text{MIA}} q$; however, $p|r \sqsubseteq_{\text{MIA}} q|r$ does not hold since q and r are compatible while p and r are not. An analogue reasoning applies to IA, although we do not know of a reference in the IA literature where this has been observed.

4.3 Embedding of IA into MIA

To conclude, we provide an embedding of IA into MIA in the line of [8]:

Definition 24 (IA-Embedding). Let P be an IA. The embedding $[P]_{\text{MIA}}$ of P into MIA is defined as the MIA $(P, I, O, \longrightarrow, \dashrightarrow)$, where (i) $p \xrightarrow{i} p'$ if $p \xrightarrow{i}_P p'$ and $i \in I$, and (ii) $p \overset{\alpha}{\dashrightarrow} p'$ if $p \xrightarrow{\alpha}_P p'$ and $\alpha \in I \cup O \cup \{\tau\}$.

This embedding is much simpler than the one of [8] since MIA more closely resembles IA than IOMTS does. In particular, the following theorem is obvious:

Theorem 25 (IA-Embedding Respects Refinement). *For IAs P, Q with $p \in P$, $q \in Q$: $p \sqsubseteq_{IA} q$ in P and Q iff $p \sqsubseteq_{MIA} q$ in $[P]_{MIA}$ and $[Q]_{MIA}$.*

Our embedding respects operators $|$ and $\wedge$, unlike the one in [8]:

Theorem 26 (IA-Embedding is a Homomorphism). *For IAs P, Q with $p \in P$, $q \in Q$: (a) $p \wedge q$ (in $[P]_{MIA} \wedge [Q]_{MIA}$) $\sqsupseteq\sqsubseteq_{MIA}$ $p \wedge q$ (in $[P \wedge Q]_{MIA}$); (b) $p|q$ (in $[P]_{MIA}|[Q]_{MIA}$) $\sqsupseteq\sqsubseteq_{MIA}$ $p|q$ (in $[P|Q]_{MIA}$).*

5 Conclusions & Future Work

We introduced *Modal Interface Automata* (MIA), an interface theory that is more expressive than *Interface Automata* (IA) [5]: it allows one to mandate that a specification's refinement must implement some output, thus excluding trivial implementations, e.g., one that accepts all inputs but never emits any output. This was also the motivation behind *IOMTS* [8] that extends *Modal Transition Systems* (MTS) [7] by inputs and outputs; however, the IOMTS-parallel operator in the style of IA is not compositional. MIA is a subset of IOMTS, but it has a different refinement relation that is a precongruence for parallel composition.

Most importantly and in contrast to IA and IOMTS, the MIA theory is equipped with a conjunction operator for reasoning about components that satisfy multiple interfaces simultaneously. Along the way, we also introduced conjunction on IA and (a disjunctive extension of) MTS, and proved these operators to be the desired greatest lower bounds and thus compositional. Compared to

the language-based modal interface theory of [12], our formalism supports nondeterministic specifications (wrt. outputs). Hence, MIA establishes a theoretically clean and practical interface theory that fixes the shortcomings of related work.

Regarding future work, we plan to study the algorithmic complexity implied by MIA-refinement [12], and MIA's expressiveness in comparison to other theories via thoroughness [6]. On the practical side, we wish to adapt existing tool support for interface theories, e.g., the *MIO Workbench* [3], to MIA.

References

[1] M. Abadi and L. Lamport. Conjoining specifications. *ACM TOPLAS*, 1(3):507–534, 1995.

[2] S. Bauer, R. Hennicker, and M. Wirsing. Interface theories for concurrency and data. *Theoret. Comp. Sc.*, 412(28):3101–3121, 2011.

[3] S. Bauer, P. Mayer, A. Schroeder, and R. Hennicker. On weak modal compatibility, refinement, and the MIO Workbench. In *TACAS*, volume 6015 of *LNCS*, pages 175–189. Springer, 2010.

[4] L. de Alfaro and T.A. Henzinger. Interface automata. In *FSE*, pages 109–120. ACM, 2001.

[5] L. de Alfaro and T.A. Henzinger. Interface-based design. In *Engineering Theories of Software-Intensive Systems*, volume 195 of *NATO Science Series*. Springer, 2005.

[6] H. Fecher, D. de Frutos-Escrig, G. Lüttgen, and H. Schmidt. On the expressiveness of refinement settings. In *FSEN*, volume 5961 of *LNCS*, pages 276–291. Springer, 2009.

[7] K.G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 232–246. Springer, 1990.

[8] K.G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *ESOP*, volume 4421 of *LNCS*, pages 64–79. Springer, 2007.

[9] K.G. Larsen, B. Steffen, and C. Weise. A constraint oriented proof methodology based on modal transition systems. In *TACAS*, volume 1019 of *LNCS*, pages 17–40. Springer, 1995.

[10] K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *J. Logic Comput.*, 1(6):761–795, 1991.

[11] G. Lüttgen and W. Vogler. Ready simulation for concurrency: It's logical! *Inform. and Comput.*, 208:845–867, 2010.

[12] J. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone. A modal interface theory for component-based design. *Fund. Inform.*, 107:1–32, 2011.

[13] M. Schäfer and W. Vogler. Component refinement and CSC-solving for STG decomposition. *Theoret. Comp. Sc.*, 388(1-3):243–266, 2007.