A Generalised Theory of Interface Automata, Component Compatibility and Error*

Sascha Fendrich and Gerald Lüttgen

Software Technologies Research Group, University of Bamberg, Germany
{sascha.fendrich,gerald.luettgen}@swt-bamberg.de

Abstract. Interface theories allow systems designers to reason about the composability and compatibility of concurrent system components. Such theories often extend both de Alfaro and Henzinger's *Interface Automata* and Larsen's *Modal Transition Systems*, which leads, however, to several issues that are undesirable in practice: an unintuitive treatment of specified unwanted behaviour, a binary compatibility concept that does not scale to multi-component assemblies, and compatibility guarantees that are insufficient for software product lines.

In this paper we show that communication mismatches are central to all these problems and, thus, the ability to represent such errors semantically is an important feature of an interface theory. Accordingly, we present the *error-aware* interface theory EMIA, where the above shortcomings are remedied by introducing explicit *fatal error states*. In addition, we prove via a Galois insertion that EMIA is a conservative generalisation of the established MIA (Modal Interface Automata) theory.

1 Introduction

Today's software systems are increasingly composed from off-the-shelf components. Hence, software developers desire to detect incompatibilities between components early. This is supported by *interface theories* [1,2,4,6,7,9,17,20,21], which may serve as specification theories for component-based design [2,4,8,15], software product lines [17], web services [5] and the Internet of Things [19]. Interface theories may also be employed as contract languages or behavioural type theories when transitioning from software design to implementation [3,13].

Many interface theories [4,6,17,20,21] extend de Alfaro and Henzinger's *Interface Automata* (IA) [1,2] and Larsen's *Modal Transition Systems* (MTS) [16,18]. In order to express compatibility assumptions of components on the communication behaviour of their environment, IA divides an interface's action alphabet into input ('?'), output ('!') and an internal action τ. A *communication mismatch*, or error, arises between parallelly composed components P and Q, if P may issue an output $a!$ while Q is not ready to receive the input $a?$ in its current state. Orthogonally, MTS permits one to specify required and optional behaviour. Taking stepwise decisions on the optional behaviour allows for a component-based, incremental design, which is supported by a compositional refinement preorder.

* Supported by the DFG (German Research Foundation) under grant LU-1748/3-1.

Unfortunately, interface theories combining IA and MTS have several issues that impact their practical use. *Issue (A):* Forbidden inputs are preserved by the resp. refinement preorder but are widely ignored by parallel composition, such that behaviour that is forbidden in one component may be re-introduced in the composed system if another component defies this prohibition. This unintuitive treatment of communication mismatches and, in particular, unwanted behaviour, is dangerous for safety-critical applications. *Issue (B):* Pairwise binary compatibility of multiple components does not guarantee their overall compatibility when being considered as a multi-component assembly, and vice versa, even if parallel composition is associative. To address this, Hennicker and Knapp [14] have introduced *assembly theories* that extend interface theories by a separate level of assemblies where multi-component compatibility is checked. However, these assemblies have to be re-interpreted as interfaces to be of further use. *Issue (C):* Optional behaviour, modelled via may-transitions as in MTS, may be employed to express variability inherent in software product lines. In current interface theories, two product families may be considered compatible only if all products of one family are compatible with all products of the other. However, one would prefer a more detailed set of guarantees, such that one may distinguish if all, some or none of the product lines' products are compatible [17]. *Issue (D):* MTS and MTS-based interface theories have some subtle differences wrt. modalities, resulting in different composition concepts: in MTS, components unanimously agree on transitions of their composition; in interface theories, an error arises if the components' requirements do not match. Each theory makes a global choice of a composition concept, which is tightly bound to a respective compatibility notion and does not allow one to mix different compatibility and composition concepts that are suitable for the application at hand.

This paper shows that communication mismatches are central to Issues (A)–(D) above. Hence, the ability to represent such errors semantically is an important feature that is missing in current interface theories. We illustrate this in Sec. 2 by an example wrt. Issue (A). In Sec. 3 we present our interface theory *Error-aware Modal Interface Automata* (EMIA), for which we remedy Issues (A)–(D) by making communication mismatches explicit in form of *fatal error states* and by employing an *error-aware refinement preorder*. In contrast, current interface theories [1,2,4,6,7,9,17,20,21] remove such information about the causes and possible resolutions of communication mismatches. As is typical for interface theories, EMIA also includes conjunction and disjunction operators, which enables systems designers to combine operational and declarative specification styles. In Sec. 4 we show that a Galois insertion [10] renders our refined semantics a conservative extension of the arguably most general interface theory to date, MIA (Modal Interface Automata) [6]. Sec. 5 revisits the example of Sec. 2 in terms of EMIA, and discusses how fatal error states solve Issues (A)–(D). The resulting specification theory tightly integrates MTS, interface theories and assembly theories, and allows systems designers to combine the different composition concepts of these theories within a single interface specification. Due to space constraints, the proofs of our results are included in a technical report [12].

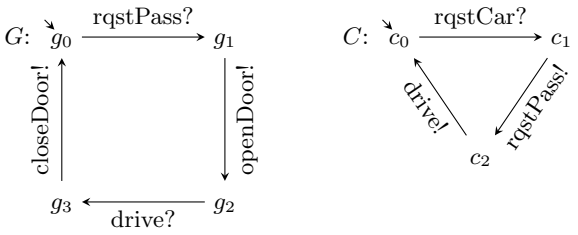

Fig. 1. Example of a driving assistant system including a garage G and a car C.

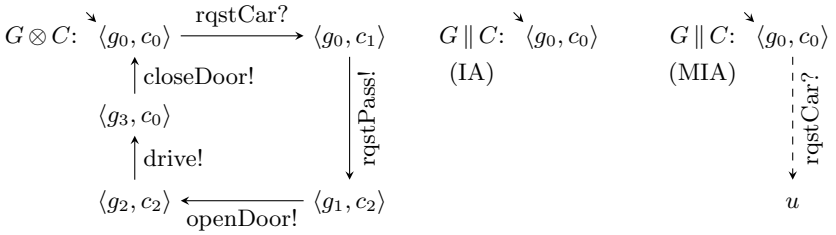

Fig. 2. Parallel product in IA or MIA (left), and parallel composition in IA (middle) and MIA (right) of the components depicted in Fig. 1.

2 Motivating Example

In this section we discuss compatibility problems of current interface theories by means of an illustrative example highlighting Issue (A). Consider a driving assistance system that enables a car to drive into and out of a garage autonomously. Such a system must communicate with the garage in order to make it open and close its door. In Fig. 1 we show specifications G and C of the garage's and the car's interfaces, resp. Starting in state g_0, the garage is ready to receive a passage request (rqstPass?). After such a request, the garage opens its door (openDoor!), waits for a car driving in or out (drive?) and, finally, closes the door (closeDoor!) again. The car starts in state c_0 waiting for a user's request (rqstCar?). Upon receiving such a request, the car requests passage from the garage (rqstPass!) and then drives into or out of the garage (drive!), reaching state c_0 again.

Specifications G and C have a communication mismatch due to the drive!-transition at state c_2 and the fact that no drive?-transition is specified at state g_1. Hence, in the parallel product $G \otimes C$ shown in Fig. 2 (left), state $\langle g_1, c_2 \rangle$ is considered illegal. In *pessimistic* theories, e.g., [4,20], the parallel composition of G and C is undefined, because the illegal state $\langle g_1, c_2 \rangle$ is reachable from the initial state $\langle g_0, c_0 \rangle$. *Optimistic* theories, e.g., [1,2,6,7,9,17,20,21], assume a helpful environment that tries to steer away from communication mismatches by controlling the composed system via its input transitions. A state is optimistically illegal if a communication mismatch is reachable via uncontrollable actions, i.e., output- or τ-transitions. Parallel composition $G \parallel C$ is obtained from $G \otimes C$ by

removing all illegal states. In our example, state $\langle g_1, c_2\rangle$ is illegal, just as state $\langle g_0, c_1\rangle$ from which $\langle g_1, c_2\rangle$ is reachable by an output (rqstPass!). This pruning leaves a single state $\langle g_0, c_0\rangle$ with no transitions; all other states are unreachable. The rqstCar?-transition at state $\langle g_0, c_0\rangle$, which would allow one to reach illegal states when triggered by the environment, is also removed. However, in order to ensure compositionality of refinement, rqstCar? must be permitted with arbitrary behaviour afterwards (cf. [6]); IA-based refinement [1,2,20] allows this implicitly for all unspecified inputs (Fig. 2, middle). In MTS-based interface theories, where unspecified transitions represent forbidden behaviour, compositionality is achieved by replacing pruned behaviour by an explicit optional transition to a special, universally refineable state u (Fig. 2, right) [6].

Due to this possibility of introducing arbitrary behaviour in case of a communication mismatch, stepwise refinement may re-introduce behaviour that has previously been removed due to the mismatch. Hence, optimistic theories accept a car driving into or out of the garage before the door is opened as a valid implementation of $G \parallel C$. This contradicts G's sensible constraint that driving in or out is only permitted after the door has been opened, i.e., the meaning of a car crashing into the door can simply be 'refined' to not being an error. In other words, the assumptions and guarantees expressible in current interface theories are insufficient for expressing unwanted behaviour.

Bujtor and Vogler [7] have shown that keeping or removing illegal states on a purely syntactic level are equivalent for IA wrt. preserving compatibility. In this spirit, current interface theories [1,2,4,6,7,17,20,21] eliminate erroneous behaviour either by regarding it as undefined (pessimistic) or by pruning (optimistic); all errors are treated semantically equivalent. Due to this equivalence, theories combining IA and MTS cannot remove illegal states completely but must replace them by a special, arbitrarily refinable behaviour as mentioned above. However, because optional transitions (i.e., may-transitions) allow for underspecification in MTS-based interface theories, one may distinguish potential errors that can be resolved by a suitable refinement from actual, unresolvable errors that arise when an output is required and the corresponding input is forbidden. That is, specifications based on MTS contain more information wrt. compatibility, which we make explicit in EMIA. EMIA guarantees that compatible specifications have only compatible implementations, potential errors have both compatible and erroneous implementations, and actual errors have only erroneous implementations (cf. Sec. 5, Issue (C)).

3 Error-aware Modal Interface Automata

Our interface theory *Error-aware Modal Interface Automata* (EMIA), which we present in this section, is equipped with a *parallel composition operator* modelling concurrency and communication, a *conjunction operator* permitting the specification of a component from different perspectives, and a *compositional refinement preorder* enabling the substitution of an interface by a more concrete version. In addition to these standard requirements on interface theories, EMIA

solves Issues (A)–(D) of Sec. 1. We achieve this by introducing *fatal error states*, which represent unresolvable incompatibilities between interfaces. This enables EMIA to deal with errors on a semantic level, since forbidden behaviour can be modelled by input transitions leading to a fatal error state.

Definition 1 (Error-aware Modal Interface Automata). *An* Error-aware Modal Interface Automaton *(EMIA) is a tuple* $P := (S_P, I_P, O_P, \longrightarrow_P, \dashrightarrow_P, S_P^0, D_P)$*, where* S_P *is the set of states,* I_P*,* O_P *are the disjoint alphabets of input and output actions not including the silent action* τ *(we define* $A_P := I_P \cup O_P$ *and* $\Omega_P := O_P \cup \{\tau\}$*),* $\longrightarrow_P \subseteq S_P \times (A_P \cup \{\tau\}) \times \mathfrak{P}(S_P)$ *is the disjunctive must-transition relation (*$\mathfrak{P}$ *denotes the power set operator),* $\dashrightarrow_P \subseteq S_P \times (A_P \cup \{\tau\}) \times S_P$ *is the may-transition relation,* $S_P^0 \subseteq S_P$ *is the set of initial states, and* $D_P \subseteq S_P$ *is the set of fatal error states. We also adopt* syntactic consistency *from MTS, i.e., for all* $\alpha \in A_P \cup \{\tau\}$ *and* $p \xrightarrow{\alpha} P'$*, we have* $\forall p' \in P.\, p \dashrightarrow^{\alpha} p'$*.*

Our definition of weak transitions that abstract from internal behaviour is adopted from the one in MIA [6]:

Definition 2 (Weak Transition Relations). *Let* P *be an EMIA. We define* weak *must- and may-transition relations,* $\Longrightarrow$ *and* $\Mapsto$ *resp., as the smallest relations satisfying the following conditions, where we use* $P' \stackrel{\hat{\alpha}}{\Longrightarrow} P''$ *as a shorthand for* $\forall p \in P'\, \exists P_p.\, p \stackrel{\hat{\alpha}}{\Longrightarrow} P_p$ *and* $P'' = \bigcup_{p \in P'} P_p$*:*

WT1. $p \stackrel{\epsilon}{\Longrightarrow} \{p\}$ *for all* $p \in P$*,*
WT2. $p \xrightarrow{\tau} P'$ *and* $P' \stackrel{\hat{\alpha}}{\Longrightarrow} P''$ *implies* $p \stackrel{\hat{\alpha}}{\Longrightarrow} P''$*,*
WT3. $p \xrightarrow{a} P'$ *and* $P' \stackrel{\epsilon}{\Longrightarrow} P''$ *implies* $p \stackrel{a}{\Longrightarrow} P''$*,*
WT4. $p \stackrel{\epsilon}{\Mapsto} p$*,*
WT5. $p \stackrel{\epsilon}{\Mapsto} p'' \dashrightarrow^{\tau} p'$ *implies* $p \stackrel{\epsilon}{\Mapsto} p'$*,*
WT6. $p \stackrel{\epsilon}{\Mapsto} p'' \dashrightarrow^{\alpha} p''' \stackrel{\epsilon}{\Mapsto} p'$ *implies* $p \stackrel{\alpha}{\Mapsto} p'$*.*

We write $\xrightarrow{a}\!\!\stackrel{\epsilon}{\Longrightarrow}$ *for transitions built up according to WT3 and call them* trailing-weak *must-transitions. Similarly,* $\dashrightarrow\!\!\stackrel{\epsilon}{\Mapsto}$ *stands for trailing-weak may-transitions.*

Our *error-aware modal refinement preorder* $\sqsubseteq_{\mathrm{EA}}$ corresponds to standard modal refinement from MTS [16,18] but reflects *and* preserves fatal error states. Intuitively, $P \sqsubseteq_{\mathrm{EA}} Q$ for an implementation P and a specification Q, enforces that P's may-transitions are permitted by Q while for any of Q's disjunctive must-transitions at least one of the branches is implemented by P. In contrast to DMTS [18], we require that all branches of a disjunctive transition have the same label. This is sufficient for our purposes and does away with technical complications of parallel composition in the presence of τ-transitions (cf. [12]).

Definition 3 (Error-aware Modal Refinement). *Let* P *and* Q *be EMIAs with equal alphabets, i.e.,* $I_P = I_Q$ *and* $O_P = O_Q$*. A relation* $\mathcal{R} \subseteq S_P \times S_Q$ *is an* error-aware modal refinement *relation (EA-refinement) if, for all* $\langle p, q \rangle \in \mathcal{R} \setminus (D_P \times D_Q)$*, the following conditions hold:*

R1. $p \notin D_P$ *and* $q \notin D_Q$*,*

R2. $q \xrightarrow{i} Q'$ *implies* $\exists P'.\, p \overset{i}{\to}\overset{\epsilon}{\Rightarrow} P'$ *and* $\forall p' \in P'\ \exists q' \in Q'.\, \langle p', q' \rangle \in \mathcal{R}$,
R3. $q \xrightarrow{\omega} Q'$ *implies* $\exists P'.\, p \overset{\omega}{\Longrightarrow} P'$ *and* $\forall p' \in P'\ \exists q' \in Q'.\, \langle p', q' \rangle \in \mathcal{R}$,
R4. $p \mathrel{-\!\!\!\overset{i}{\to}} p'$ *implies* $\exists q'.\, q \overset{i}{\to}\overset{\epsilon}{\Rightarrow} q'$ *and* $\langle p', q' \rangle \in \mathcal{R}$,
R5. $p \mathrel{-\!\!\!\overset{\omega}{\to}} p'$ *implies* $\exists q'.\, q \overset{\omega}{\Rightarrow} q'$ *and* $\langle p', q' \rangle \in \mathcal{R}$.

We write $p \sqsubseteq_{\mathrm{EA}} q$ *if there is an EA-refinement* $\mathcal{R}$ *with* $\langle p, q \rangle \in \mathcal{R}$, *and* $P \sqsubseteq_{\mathrm{EA}} Q$ *if, for each* $p \in S^0_P$, *there is a* $q \in S^0_Q$ *with* $p \sqsubseteq_{\mathrm{EA}} q$. *If* $p \sqsubseteq_{\mathrm{EA}} q$ *and* $q \sqsubseteq_{\mathrm{EA}} p$, *we employ the symbol* $p \sqsupseteq\sqsubseteq_{\mathrm{EA}} q$, *and similar for EMIAs* P, Q.

The refinement relation $\sqsubseteq_{\mathrm{EA}}$ is reflexive and transitive and, hence, a preorder. Moreover, we have $p \in D_P$ iff $q \in D_Q$ for all $\langle p, q \rangle \in \mathcal{R}$ due to R1. Optional input-transitions, which may be refined to required or forbidden behaviour, are expressed as a disjunctive must-transition containing a fatal error state in its set of target states. For example, optional $a?$-transitions from a state p_0 to states p_1 and p_2 are modelled as $p_0 \xrightarrow{a?} \{p_1, p_2, p_3\}$ for some fatal error state $p_3 \in D_P$.

IA's parallel composition operator synchronises input and output transitions to τ-transitions. In contrast, we define a multicast parallel composition, where an output can synchronise with multiple input transitions, as in MI [21] and MIA [6]. We leave out MIA's separate hiding due to space constraints.

Definition 4 (Parallel Composition). *Let* P *and* Q *be EMIAs. We call* P *and* Q composable *if* $O_P \cap O_Q = \emptyset$. *If* P *and* Q *are composable, the* multicast parallel composition $P \parallel Q$ *is defined by* $S_{P \parallel Q} := S_P \times S_Q$, $I_{P \parallel Q} := (I_P \cup I_Q) \setminus O_{P \parallel Q}$, $O_{P \parallel Q} := O_P \cup O_Q$, $S^0_{P \parallel Q} := S^0_P \times S^0_Q$, $D_{P \parallel Q} := (D_P \times S_Q) \cup (S_P \times D_Q)$, *and the transition relations are given by the following rules:*

P1. $\langle p, q \rangle \xrightarrow{\alpha} P' \times \{q\}$ *if* $p \xrightarrow{\alpha} P'$ *and* $\alpha \notin A_Q$,
P2. $\langle p, q \rangle \xrightarrow{\alpha} \{p\} \times Q'$ *if* $\alpha \notin A_P$ *and* $q \xrightarrow{\alpha} Q'$,
P3. $\langle p, q \rangle \xrightarrow{a} P' \times Q'$ *if* $p \xrightarrow{a} P'$ *and* $q \xrightarrow{a} Q'$ *for some* $a \in A_P \cap A_Q$.
P4. $\langle p, q \rangle \mathrel{-\!\!\!\overset{\alpha}{\to}} \langle p', q \rangle$ *if* $p \mathrel{-\!\!\!\overset{\alpha}{\to}} p'$ *and* $\alpha \notin A_Q$,
P5. $\langle p, q \rangle \mathrel{-\!\!\!\overset{\alpha}{\to}} \langle p, q' \rangle$ *if* $\alpha \notin A_P$ *and* $q \mathrel{-\!\!\!\overset{\alpha}{\to}} q'$,
P6. $\langle p, q \rangle \mathrel{-\!\!\!\overset{a}{\to}} \langle p', q' \rangle$ *if* $p \mathrel{-\!\!\!\overset{a}{\to}} p'$ *and* $q \mathrel{-\!\!\!\overset{a}{\to}} q'$ *for some* $a \in A_P \cap A_Q$.

We also write $p \parallel q$ for $\langle p, q \rangle$. IA-based interface theories usually define a communication mismatch for p at q as a situation where an action $a \in O_P \cap I_Q$ is permitted at p and not required at q. In EMIA, such a situation is modelled with the help of an $a?$-must-transition from q to a target set Q' that includes some fatal error state $q' \in D_Q$, as explained above. Parallel composition is associative and commutative. Further, $\sqsubseteq_{\mathrm{EA}}$ is a precongruence wrt. $\parallel$:

Proposition 5 (Compositionality). *If* P_1, P_2, Q *are EMIAs s.t.* $P_1 \sqsubseteq_{\mathrm{EA}} P_2$ *and* P_2, Q *are composable, then* P_1 *and* Q *are composable and* $P_1 \parallel Q \sqsubseteq_{\mathrm{EA}} P_2 \parallel Q$.

Perspective-based specification is concerned with specifying a system component from separate perspectives s.t. the component satisfies each of these perspective specifications; for example, each requirement for a component might describe a perspective. The component's overall specification is the most general specification refining all perspective specifications, i.e., it is the greatest lower bound wrt. the refinement preorder. This conjunction operator is defined in two stages:

Definition 6 (Conjunctive Product). *Let P, Q be EMIAs with equal alphabets. The* conjunctive product *of P and Q is $P \,\&\, Q := (S_{P\,\&\,Q}, I, O, \longrightarrow_{P\,\&\,Q}, \dashrightarrow_{P\,\&\,Q}, S^0_{P\,\&\,Q}, D_{P\,\&\,Q})$ with $S_{P\,\&\,Q} := S_P \times S_Q$, $S^0_{P\,\&\,Q} := S^0_P \times S^0_Q$, $D_{P\,\&\,Q} := D_P \times D_Q$, and the transition relations are given by the following rules:*

C1. $\langle p,q\rangle \xrightarrow{i} \{\langle p',q'\rangle \mid p' \in P',\ q \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon} q'\}$ *if* $p \xrightarrow{i} P'$ *and* $q \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon}$,

C2. $\langle p,q\rangle \xrightarrow{i} \{\langle p',q'\rangle \mid p \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon} p',\ q' \in Q'\}$ *if* $p \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon}$ *and* $q \xrightarrow{i} Q'$,

C3. $\langle p,q\rangle \xrightarrow{\omega} \{\langle p',q'\rangle \mid p' \in P',\ q \overset{\omega}{=\!\!=\!\!\Rightarrow} q'\}$ *if* $p \xrightarrow{\omega} P'$ *and* $q \overset{\omega}{=\!\!=\!\!\Rightarrow}$,

C4. $\langle p,q\rangle \xrightarrow{\omega} \{\langle p',q'\rangle \mid p \overset{\omega}{=\!\!=\!\!\Rightarrow} p',\ q' \in Q'\}$ *if* $p \overset{\omega}{=\!\!=\!\!\Rightarrow}$ *and* $q \xrightarrow{\omega} Q'$,

C5. $\langle p,q\rangle \overset{i}{\dashrightarrow} \langle p',q'\rangle$ *if* $p \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon} p'$ *and* $q \overset{i}{\dashrightarrow}\!\!\Rightarrow^{\epsilon} q'$,

C6. $\langle p,q\rangle \overset{\omega}{\dashrightarrow} \langle p',q'\rangle$ *if* $p \overset{\omega}{=\!\!=\!\!\Rightarrow} p'$ *and* $p \overset{\omega}{=\!\!=\!\!\Rightarrow} q'$,

C7. $\langle p,q\rangle \overset{\tau}{\dashrightarrow} \langle p',q\rangle$ *if* $p \overset{\tau}{=\!\!=\!\!\Rightarrow} p'$,

C8. $\langle p,q\rangle \overset{\tau}{\dashrightarrow} \langle p,q'\rangle$ *if* $q \overset{\tau}{=\!\!=\!\!\Rightarrow} q'$.

A state $\langle p,q\rangle$ of $P \,\&\, Q$ is a candidate for refining both p and q. Because $\langle p,q\rangle$ cannot require and forbid the same action a or be at once fatal and non-fatal, some states p and q do not have a common refinement. In such cases, $\langle p,q\rangle$ is called *inconsistent* and has to be removed from the candidates, including the removal of all states that require transitions leading to inconsistent states.

Definition 7 (Conjunction). *The set $F \subseteq S_{P\,\&\,Q}$ of* logically inconsistent *states is defined as the smallest set satisfying the following rules:*

F1. $\langle p,q\rangle \in (D_P \times (S_Q \setminus D_Q)) \cup ((S_P \setminus D_P) \times D_Q)$ *implies* $\langle p,q\rangle \in F$,

F2. $\langle p,q\rangle \notin D_{P\,\&\,Q}$, $p \xrightarrow{i}$ *and* $q \overset{i}{\not\dashrightarrow}$ *implies* $\langle p,q\rangle \in F$,

F3. $\langle p,q\rangle \notin D_{P\,\&\,Q}$, $p \overset{i}{\not\dashrightarrow}$ *and* $q \xrightarrow{i}$ *implies* $\langle p,q\rangle \in F$,

F4. $\langle p,q\rangle \notin D_{P\,\&\,Q}$, $p \xrightarrow{\omega}$ *and* $q \overset{\omega}{\not=\!\!=\!\!\Rightarrow}$ *implies* $\langle p,q\rangle \in F$,

F5. $\langle p,q\rangle \notin D_{P\,\&\,Q}$, $p \overset{\omega}{\not=\!\!=\!\!\Rightarrow}$ *and* $q \xrightarrow{\omega}$ *implies* $\langle p,q\rangle \in F$,

F6. $\langle p,q\rangle \xrightarrow{\alpha} R$ *and* $R \subseteq F$ *implies* $\langle p,q\rangle \in F$.

The conjunction *$P \wedge Q$ is obtained from $P \,\&\, Q$ by deleting all states in F. This deletes all transitions exiting deleted states and removes all deleted states from targets of must-transitions. If $S^0_{P\wedge Q} = \emptyset$, then P and Q are called inconsistent.*

Fatal states are excluded in Rules F2 through F5 because we do not care about consistency for fatal error states. Note that the states in D and F are different in nature: D-states represent states with possible but unwanted behaviour. F-states represent contradictory specifications that are impossible to implement. Conjunction is the greatest lower bound wrt. the refinement preorder $\sqsubseteq_{\mathrm{EA}}$:

Proposition 8 ($\wedge$ is And). *If P and Q are EMIAs with equal alphabets, then (i) $\exists R.\, R \sqsubseteq_{\mathrm{EA}} P$ and $R \sqsubseteq_{\mathrm{EA}} Q$ iff P and Q are consistent. Further, if P and Q are consistent, then, for any R, (ii) $R \sqsubseteq_{\mathrm{EA}} P$ and $R \sqsubseteq_{\mathrm{EA}} Q$ iff $R \sqsubseteq_{\mathrm{EA}} P \wedge Q$.*

As a standard category theoretic result, Prop. 8 implies that $\wedge$ is associative:

Corollary 9 (Associativity of $\wedge$). *Conjunction is strongly associative, i.e., for all EMIAs P, Q, and R, if one of $P \wedge (Q \wedge R)$ and $(P \wedge Q) \wedge R$ is defined, then both are defined and $P \wedge (Q \wedge R) \sqsupseteq\sqsubseteq_{\mathrm{EA}} (P \wedge Q) \wedge R$.*

We close this section with a remark on alphabet extension. Conjunction, disjunction and refinement are defined for EMIAs with equal alphabets. For perspective-based specification, it is of interest to consider EMIAs with different alphabets [6]. Following the lines of MI and MIA, the operations on EMIAs can be lifted to different alphabets by extending the alphabets of the operands by their mutually foreign actions. When a specification's alphabet is extended, the least possible assumptions should be made on a new action a, while the same specification wrt. known actions should hold before and after a. This can be achieved by adding an optional a-loop to each state. For output actions this is straightforward, but the exact meaning of optional input transitions depends on the desired composition concept (cf. Sec. 1, Issue (D)). Therefore, a separate alphabet extension operator has to be defined for unanimous, broadcast and error-sensitive parallel composition. Besides this, there is nothing surprising to expect from alphabet extension, and we leave out the formal definition here for brevity.

4 Relation to other Interface Theories

The majority of IA-based interface theories prune errors. Therefore, it is important to investigate the relation between such error-pruning interface theories and our non-pruning EMIA theory. We do this for MIA [6] because it is the most general IA-based interface theory to date in that it is nondeterministic rather than deterministic and optimistic rather than pessimistic, thus subsuming MI [21] and MIO [4] (wrt. strong compatibility), resp. We establish here a Galois insertion between MIA and EMIA, i.e., a Galois connection $\langle \gamma, \alpha \rangle$ for which $\alpha \circ \gamma = \mathrm{id}_{\mathsf{MIA}}$ [10] (up to $\sqsupseteq \sqsubseteq_{\mathrm{MIA}}$). Recall that states from which a communication mismatch is reachable via output- or τ-transitions are called illegal. Intuitively, α abstracts from EMIAs by considering all illegal states to be equivalent, and γ concretises MIAs as EMIAs without any loss of information. Note that γ is different from the error-completion presented in [22] that is motivated by algorithmic considerations only. Error-completion preserves an interface's semantics when replacing missing inputs by transitions to an error state. In contrast, EMIA refines the semantics of MIA by retaining error states.

Definition 10 (MIA [6]). Modal Interface Automata *(MIA) are defined like EMIAs (cf. Def. 1), except that, instead of D_P, there is a* universal state u_P *that is only permitted as target of input may-transitions.*

An important difference between fatal error states and u_P is revealed in the different notion of refinement. While EMIA employs a variant of modal refinement [18] that preserves and reflects fatal error states, MIA adopts (ordinary) modal refinement in general but provides the possibility to employ IA-refinement where necessary. This is achieved by state u_P, which may be refined arbitrarily.

Definition 11 (MIA-Refinement [6]). *Let P, Q be MIAs with equal alphabets. $\mathcal{R} \subseteq S_P \times S_Q$ is a* MIA-refinement *relation if, for all $\langle p, q \rangle \in \mathcal{R} \setminus (S_P \times \{u_Q\})$, the rules of Def. 3 hold when replacing R1 by: MR1. $p \neq u_P$.*

Parallel composition of MIAs is defined through reachability of illegal states:

Definition 12 (Backward Closure). *Let P be a MIA or EMIA and $S \subseteq S_P$. The* Ω-backward closure of S in P *is the smallest set* $\mathrm{bcl}_P^\Omega(S) \subseteq S_P$ *s.t.* $S \subseteq \mathrm{bcl}_P^\Omega(S)$ *and, for all* $\omega \in \Omega_P$ *and* $p' \in \mathrm{bcl}_P^\Omega(S)$*, if* $p \dashrightarrow^{\omega} p'$*, then* $p \in \mathrm{bcl}_P^\Omega(S)$.

Definition 13 (MIA-Parallel Composition [6]). *For composable MIAs P, Q, the* parallel product $P \otimes Q$ *is defined by ignoring fatal error states in Def. 4. We say that there is a* communication mismatch *for p at q, in symbols* $\mathrm{mis}(p,q)$*, if there is an* $a \in O_P \cap I_Q$ *with* $p \dashrightarrow^{a}$ *and* $q \not\dashrightarrow^{a}$*. The set of* illegal states *is defined as* $E_{P\otimes Q} := \mathrm{bcl}^\Omega_{P\otimes Q}(\{\langle p,q\rangle \mid \mathrm{mis}(p,q) \text{ or } \mathrm{mis}(q,p)\} \cup (S_P \times \{u_Q\}) \cup (\{u_P\} \times S_Q))$*. The* parallel composition $P \parallel Q$ *is the MIA given by the state set* $S_{P\parallel Q} := (S_{P\otimes Q} \setminus E_{P\otimes Q}) \cup \{u_{P\parallel Q}\}$*, the alphabets* $I_{P\parallel Q} := I_{P\otimes Q}$ *and* $O_{P\parallel Q} := O_{P\otimes Q}$*, and the transition relations obtained from $P \otimes Q$ by replacing all $i?$-transitions of states $\langle p,q\rangle$ having an $i?$-transition to $E_{P\otimes Q}$ by a transition* $\langle p,q\rangle \dashrightarrow^{i} u_{P\parallel Q}$*. If* $S^0_{P\otimes Q} \subseteq E_{P\otimes Q}$*, then* $S^0_{P\parallel Q} := \{u_{P\parallel Q}\}$*, else* $S^0_{P\parallel Q} := S^0_{P\otimes Q} \setminus E_{P\otimes Q}$.

The set $\mathrm{bcl}_P^\Omega(D_P) \setminus D_P$ of an EMIA P corresponds roughly to the set of illegal states in IA, EIO, MI and MIA. In contrast to these theories, EMIA requires one to match transitions of such states during refinement. The resulting refinement relation is comparable to other refinement preorders for error-free interfaces, but is more detailed for erroneous ones. Indeed, MIA can be seen as an abstraction of EMIA, where all states in $\mathrm{bcl}_P^\Omega(D_P) \setminus D_P$ are deemed equivalent (cf. Thm. 19).

Definition 14 (MIA-Conjunction [6]). *Let P and Q be MIAs with equal alphabets. The* MIA-conjunctive product *is defined by ignoring fatal error states in Def. 6 and adding the following rules for u:*

CE1. $\langle p, u_Q\rangle \xrightarrow{\alpha} P' \times \{u_Q\}$ *if* $p \xrightarrow{\alpha} P'$,
CE2. $\langle u_P, q\rangle \xrightarrow{\alpha} \{u_P\} \times Q'$ *if* $q \xrightarrow{\alpha} Q'$,
CE3. $\langle p, u_Q\rangle \dashrightarrow^{\alpha} \langle p', u_Q\rangle$ *if* $p \dashrightarrow^{\alpha} p'$,
CE4. $\langle u_P, q\rangle \dashrightarrow^{\alpha} \langle u_P, q'\rangle$ *if* $q \dashrightarrow^{\alpha} q'$.

The MIA-conjunction *is obtained from the* MIA-conjunctive product *by pruning logically inconsistent states according to Rules F2 through F6 of Def. 7.*

An input i forbidden at state p is modelled as a missing transition in MIA and, equivalently, as an i-must-transition from p to a fatal error state in EMIA. Hence, a MIA's behaviour can be modelled by an EMIA where non-fatal states are input-enabled. We write EMIA' for the collection of such EMIAs.

The Galois insertion between MIA and EMIA consists of a concretisation $\gamma : \mathsf{MIA} \to \mathsf{EMIA}'$ and an abstraction $\alpha : \mathsf{EMIA}' \to \mathsf{MIA}$ s.t. $\langle \gamma, \alpha\rangle$ is a Galois connection and $(\alpha \circ \gamma)(Q) \sqsupseteq \sqsubseteq_{\mathrm{MIA}} Q$. The main idea behind α is to consider the states $\mathrm{bcl}_P^\Omega(D_P) \setminus D_P$ as equivalent, yielding equivalence classes of EMIAs; α assigns a MIA to each of these equivalence classes. Vice versa, γ assigns to each MIA the disjunction of an equivalence class of EMIAs.

Definition 15 (Abstraction Function from EMIA' to MIA). *Let $P \in \mathsf{EMIA}'$ and $C_P := \mathrm{bcl}^{\Omega}_P(D_P) \setminus D_P$. The* MIA-abstraction of P *is the MIA $\alpha(P) := (S_{\alpha(P)}, I_P, O_P, \longrightarrow_{\alpha(P)}, \dashrightarrow_{\alpha(P)}, S^0_{\alpha(P)}, u_{\alpha(P)})$ with the state sets $S_{\alpha(P)} := (S_P \setminus (C_P \cup D_P)) \dot\cup \{u_{\alpha(P)}\}$ and $S^0_{\alpha(P)} := S^0_P \cap S_{\alpha(P)}$. The transitions of $\alpha(P)$ are obtained from P by replacing all $i?$-transitions leading from a state p to states in C_P by $p \overset{i?}{\dashrightarrow} u_{\alpha(P)}$. The kernel equivalence $\equiv_\alpha \subseteq \mathsf{EMIA}' \times \mathsf{EMIA}'$, which is defined by $P \equiv_\alpha Q$ iff $\alpha(P) \sqsupseteq\sqsubseteq_{\mathrm{MIA}} \alpha(Q)$ and has equivalence classes $[P]_\alpha$, yields a canonical bijection $\bar\alpha : \mathsf{EMIA}'/{\equiv_\alpha} \to \mathsf{MIA}$.*

To define the concretisation function γ we need a disjunction operator:

Definition 16 (Disjunction). *For a family of EMIAs $\mathcal{P} := (P_j)_{j\in J}$ with equal alphabets, we define the* disjunction *of $\mathcal{P}$ as the EMIA: $\bigvee_{j\in J} P_j := (\dot\bigcup_{j\in J} S_{P_j}, I, O, \dot\bigcup_{j\in J} \longrightarrow_{P_j}, \dot\bigcup_{j\in J} \dashrightarrow_{P_j}, \dot\bigcup_{j\in J} S^0_{P_j}, \dot\bigcup_{j\in J} D_{P_j})$.*

Proposition 17 ($\vee$ is Or). *If P_j, for $j \in J$, and R are EMIAs with equal alphabets, then $\bigvee_{j\in J} P_j \sqsubseteq_{\mathrm{EA}} R$ iff $P_j \sqsubseteq_{\mathrm{EA}} R$ for all $j \in J$.*

Disjunction on MIAs is defined analogously by ignoring fatal error states and replacing u_P and u_Q by $u_{P\vee Q}$. Obviously, α is homomorphic wrt. disjunction.

Definition 18 (Concretisation Function from MIA to EMIA'). *The concretisation function $\gamma : \mathsf{MIA} \to \mathsf{EMIA}'$ is defined as $\gamma(P) := \bigvee \bar\alpha^{-1}(P)$.*

The mappings α and γ defined in Defs. 15 and 18 are monotonic, which is key to the proof of our main result that α and γ form a Galois insertion:

Theorem 19 (Galois Insertion). *The maps $\alpha\colon \mathsf{EMIA}' \to \mathsf{MIA}$ and $\gamma\colon \mathsf{MIA} \to \mathsf{EMIA}'$ defined in Defs. 15 and 18 form a Galois insertion between MIA and EMIA' up to $\sqsupseteq\sqsubseteq_{\mathrm{MIA}}$, i.e., $P \sqsubseteq_{\mathrm{EA}} \gamma(Q)$ iff $\alpha(P) \sqsubseteq_{\mathrm{MIA}} Q$ and $(\alpha\circ\gamma)(Q) \sqsupseteq\sqsubseteq_{\mathrm{MIA}} Q$.*

Proof (sketch). $\alpha\circ\gamma = \mathrm{id}_{\mathrm{MIA}}$ by homomorphicity of α wrt. $\bigvee$; standard monotonicity and extensivity arguments establish $\langle\gamma, \alpha\rangle$ as a Galois connection. □

α is homomorphic wrt. parallel composition but not wrt. conjunction: although $\alpha(P \wedge Q) \sqsubseteq_{\mathrm{MIA}} \alpha(P) \wedge \alpha(Q)$ holds for $P, Q \in \mathsf{EMIA}'$ because α is monotonic, the converse direction "$\sqsupseteq_{\mathrm{MIA}}$" does not hold in general, because MIA's replacement of illegal states by u—which must be reproduced by α—is a non-continuous operation. For the same reason, γ is not homomorphic wrt. parallel composition; however, γ satisfies the inequality $\gamma(P \parallel Q) \sqsupseteq_{\mathrm{EA}} \gamma(P) \parallel \gamma(Q)$ for MIAs P, Q.

5 Discussion

In this section we illustrate how the fatal error states employed in EMIA solve Issues (A)–(D) criticised in Sec. 1. In particular, we establish that EMIA treats unwanted behaviour more intuitively (Issue (A)), that EMIA, in contrast to MIA, is an assembly theory (Issue (B)), that EMIA provides better support for

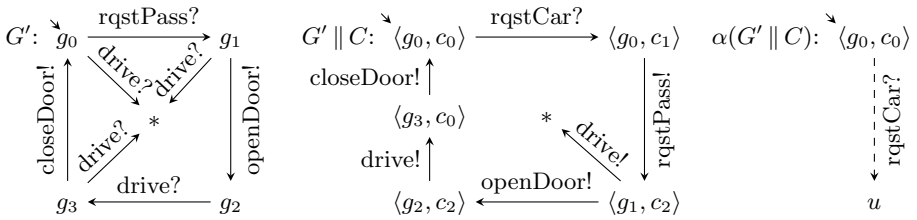

Fig. 3. Driving assistant system in EMIA and its Galois abstraction.

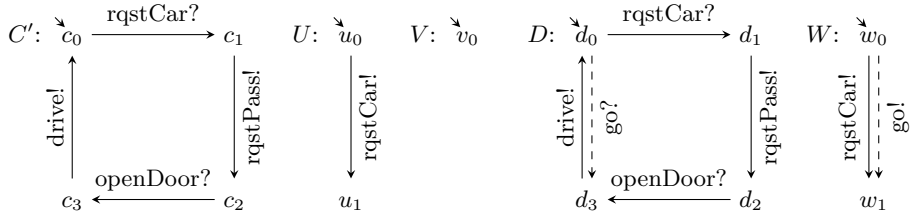

Fig. 4. Corrected car C', user interfaces U and V, and product families D and W.

specifying product families (Issue (C)), and that EMIA unifies the composition concepts of MTS and interface theories (Issue (D)). We do this mostly along the example of Sec. 2 and also use this example to demonstrate the Galois abstraction from EMIA to MIA.

Issue (A): In EMIA, the garage's constraint that a car shall not drive in or out in state g_1 would be specified by a drive?-transition to a fatal error state $*$, which represents an unresolvable error as is illustrated in specification G' in Fig. 3. In the resulting parallel composition $G' \parallel C$, also shown in Fig. 3, driving in or out too early in state $\langle g_1, c_2 \rangle$, when the door is still closed, leads to the fatal error state $*$, where the car crashes into the door. This information is not removed and cannot be redefined to not being an accident by refining $G' \parallel C$. Keeping this information is essential for pinning down the location and the cause of the error within the specification. Because G' forbids action drive? between rqstPass? and openDoor! but allows drive? after openDoor!, we can infer that specification C must be aware of action openDoor! in order to be compatible with G'. This way, a software design tool based on EMIA can propose possible specification changes to the designer. For example, the tool may propose to add action openDoor? to the car's alphabet and to insert an openDoor?-transition between rqstPass! and drive!, so as to avoid the fatal error state $*$ that is reachable from $\langle g_1, c_2 \rangle$. The resulting specification is shown as C' in Fig 4.

Galois abstraction: Fig. 3 (right) illustrates the abstraction function α of the Galois insertion between MIA and EMIA. We have $C_{G' \parallel C} := \mathrm{bcl}^{\Omega}_{G' \parallel C}(D_{G' \parallel C}) \setminus D_{G' \parallel C} = \{\langle g_1, c_2 \rangle, \langle g_0, c_1 \rangle\}$ (cf. Sec. 4). The rqstCar?-must-transition at $\langle g_0, c_0 \rangle$ leading to $C_{G' \parallel C}$ is replaced by a rqstCar?-may-transition to $u_{\alpha(G' \parallel C)}$. Due to

α being a homomorphism wrt. $\|$, this result corresponds exactly to the MIA shown in Fig. 2 (right).

Issue (B): When adding the specification of a simple user interface, shown as U in Fig. 4, as a third component to the specifications G and C of Fig. 1, the three components G, C and U are pairwise optimistically compatible. However, the composed system $G \parallel C \parallel U$ is incompatible, because the mismatch for action drive! is reachable from the initial state $\langle g_0, c_0, u_0 \rangle$. In other words, MIA is not by itself an assembly theory. A different but related problem arises in pessimistic theories: the user interface specification V in Fig. 4 promises to never request a car. The components G and C are pessimistically incompatible and $(G \parallel C) \parallel V$ is undefined. However, $G \parallel (C \parallel V)$ is a perfectly valid composition.

To lift their interface theory MIO to an assembly theory, Hennicker and Knapp propose an enrichment EMIO of MIO by error states similar to our fatal errors [14]. However, they do not develop EMIO into a full interface theory: EMIOs are only employed to describe the result of a multi-component parallel composition and to check the communication safety of such an assembly. In addition, refinement is lifted to assemblies by providing an error-preserving refinement relation for EMIOs, which is similar to EA-refinement. However, no further operations like parallel composition or conjunction are defined for assemblies; instead, EMIO forms a second layer on top of MIO, and an EMIO is re-interpreted as MIO via an encapsulation function that removes all error-information. In contrast to this loose integration, EMIA provides a uniform and tight integration of interfaces and assemblies by directly including its canonical assembly theory in the sense of [14]. In particular, EMIA does not need two separate refinement relations for interfaces and assemblies.

Theorem 20 (Assembly Theory). *EMIA induces a* canonical *assembly theory (i.e., where encapsulation is equivalent to parallel composition).*

The proof is straightforward by checking the conditions of the definitions in [14]i (cf. [12]). Because encapsulation corresponds to $\|$ and the assembly refinement preorder to $\sqsubseteq_{\mathrm{EA}}$, EMIA directly includes its canonical assembly theory.

Translating the above examples of assemblies with U and V into EMIA, the composition $G' \parallel C \parallel U$ resembles $G' \parallel C$ (Fig. 3), except that action rqstCar is an output instead of an input. Further, $(G' \parallel C) \parallel V$ and $G' \parallel (C \parallel V)$ are equivalent in EMIA. In both examples, compatibility is checked via reachability of fatal error states. However, it is up to the system designer to decide which error behaviour yields an incompatibility, i.e., compatibility is not necessarily a global concept as is the case for optimistic and pessimistic compatibility.

Issue (C): Consider specifications D and W of a car and a user interface product family, resp., both of which are shown in Fig. 4. These specifications allow product variations of a car and a user interface, which enable drivers to initiate the automatic driving assistance manually (go!), e.g., when parking in a different garage that is not equipped with an automatic door opener. Obviously, a user interface that provides this feature is incompatible with a car that does not, i.e., although some product combinations of D and W are compatible, some of

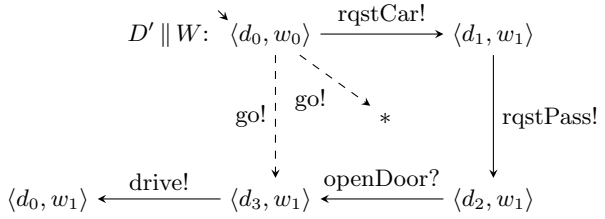

Fig. 5. Composition of product lines D' and W in EMIA.

them are not. Hence, D and W are incompatible, and no information that might help finding compatible product combinations is provided in current interface theories (see also the discussion about actual and potential errors in Sec. 2). In EMIA, the optional go?-transition at state d_0 would be modelled as a disjunctive go?-must-transition from d_0 to $\{d_3, *\}$, for a fatal error state $*$. We refer to this specification as D'. The specified error information is still present in the parallel composition of D' and W, so that one may derive additional conditions on the go-transitions. These conditions result in compatible refinements of D' and W, which describe compatible sub-families of the original product families. For example, refining the optional go?-transition into a mandatory one in D', or removing the optional go!-transition in W; both result in appropriate restrictions to sub-families. The necessary error information is present in the EMIA parallel composition of D' and W (cf. Fig. 5).

Issue (D): MTS and interface theories combining IA with MTS share many aspects of the modality semantics wrt. refinement. However, the meaning of may- and must-modalities differs wrt. parallel composition. Required and forbidden actions never cause an error in a parallel composition in MTS: either all components *unanimously* agree on implementing an action, or the action is forbidden in the composed system. The possibility to disagree on transitions enables an environment to control all transitions of an MTS, such that they may be interpreted as input-transitions from an interface theoretic view. However, the MTS parallel composition does not directly scale to output actions, because these cannot be controlled by the environment. Consequently, previous interface theories have adopted an IA-like *error-aware* parallel composition that is tightly bound to a global compatibility concept. In contrast, EMIA's explicit error representation allows for a *local* description of compatibility that is independent of composition. Thus, EMIA unifies unanimous and error-aware parallel composition, i.e., it permits the mixing of these composition concepts within a specification. As an aside, note that EMIA collapses to MTS when considering input actions only.

6 Conclusions

Our interface theory EMIA is a uniformly integrated specification framework that is applicable at different levels of abstraction, e.g., component-based de-

sign, product line specification and programming with behavioural types. EMIA bridges the gaps between MTS [18], interface theories [1,2,4,6,7,9,17,20,21] and assembly theories [14]. It is based on a concept of *error-awareness*, whereby EMIA's refinement preorder reflects *and* preserves fatal error states. While recent interface theories [6,21] considered the problem of how to enforce required behaviour, our finer-grained error semantics also solves the dual and previously open problem of how to forbid unwanted behaviour.

We proved that EMIA is related to the IA-based interface theory MIA [6] via a Galois insertion, rendering MIA into an abstraction of EMIA. In the abstract theory, errors may be considered as models of unknown behaviour for which no guarantees can be made, while in EMIA errors model unwanted behaviour for which we know that it must not be implemented. This difference between EMIA and related interface theories can be captured in a more concise way when considering error states axiomatically. In related theories [6,21], an error state e satisfies the laws $e \parallel q = e$, meaning that a composed system is in an erroneous state if a component is, and $e \sqsubseteq p \;\Rightarrow\; p = e$, meaning that an error cannot be introduced when refining an ordinary state. In EMIA, the additional law $p \sqsubseteq e \;\Rightarrow\; p = e$ is satisfied, i.e., refining cannot redefine an erroneous situation to be non-erroneous.

Regarding future work we intend to add alphabet extension and quotienting, and wish to capture differences and commonalities of different interface theories via axiomatisations. We also plan to implement EMIA in a formal methods tool, e.g., Mica [8], the MIO-Workbench [4] or MoTraS [15], and to adapt EMIA as a behavioural type theory for the Go Programming Language [13]. Such tools would enable us to evaluate EMIA on larger, more realistic examples, e.g., the docking system studied in the context of IA in [11].

Acknowledgements. We are grateful to Ferenc Bujtor, Walter Vogler and the anonymous reviewers for their helpful suggestions.

References

1. de Alfaro, L., Henzinger, T.A.: Interface automata. In: Foundations of Software Engineering (FSE). pp. 109–120. ACM (2001)
2. de Alfaro, L., Henzinger, T.A.: Interface-based design. In: Engineering Theories of Software-Intensive Systems, NATO Science, vol. 195, pp. 83–104. Springer (2005)
3. Bauer, S.S., David, A., Hennicker, R., Larsen, K.G., Legay, A., Nyman, U., Wasowski, A.: Moving from specifications to contracts in component-based design. In: Fundamental Approaches to Software Engineering (FASE). LNCS, vol. 7212, pp. 43–58. Springer (2012)
4. Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On weak modal compatibility, refinement, and the MIO Workbench. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 6015, pp. 175–189. Springer (2010)
5. Beyer, D., Chakrabarti, A., Henzinger, T.A., Seshia, S.A.: An application of web-service interfaces. In: Intl. Conf. on Web Services (ICWS). pp. 831–838. IEEE (2007)

6. Bujtor, F., Fendrich, S., Lüttgen, G., Vogler, W.: Nondeterministic modal interfaces. In: Theory and Practice of Computer Science (SOFSEM), LNCS, vol. 8939, pp. 152–163. Springer (2015), An extended version of this paper, with a corrected definition of weak refinement, has been submitted to the TCS journal.
7. Bujtor, F., Vogler, W.: Error-pruning in interface automata. In: Theory and Practice of Computer Science (SOFSEM). LNCS, vol. 8327, pp. 162–173. Springer (2014)
8. Caillaud, B.: Mica: A modal interface compositional analysis library (2011), `http://www.irisa.fr/s4/tools/mica/`, online, accessed 2 Dec. 2015
9. Chen, T., Chilton, C., Jonsson, B., Kwiatkowska, M.Z.: A compositional specification theory for component behaviours. In: Programming Languages and Systems (ESOP). LNCS, vol. 7211, pp. 148–168. Springer (2012)
10. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Principles of Programming Languages (POPL). pp. 238–252. ACM (1977)
11. Emmi, M., Giannakopoulou, D., Păsăreanu, C.S.: Assume-guarantee verification for interface automata. In: Formal Methods (FM). LNCS, vol. 5014, pp. 116–131. Springer (2008)
12. Fendrich, S., Lüttgen, G.: A generalised theory of interface automata, component compatibility and error. Tech. Rep. Bamberger Beiträge zur Wirtschaftsinformatik und angewandten Informatik 98, Bamberg University (2016)
13. Gareis, J.: Prototypical Integration of the Modal Interface Automata Theory in Google Go. Master's thesis, Bamberg University, Germany (2015)
14. Hennicker, R., Knapp, A.: Moving from interface theories to assembly theories. Acta Informatica 52(2-3), 235–268 (2015)
15. Křetínský, J., Sickert, S.: MoTraS: A tool for modal transition systems and their extensions. In: Automated Technology for Verification and Analysis (ATVA), LNCS, vol. 8172, pp. 487–491. Springer (2013)
16. Larsen, K.G.: Modal specifications. In: Automatic Verification Methods for Finite State Systems. LNCS, vol. 407, pp. 232–246. Springer (1989)
17. Larsen, K.G., Nyman, U., Wasowski, A.: Modal I/O automata for interface and product line theories. In: Programming Languages and Systems (ESOP). LNCS, vol. 4421, pp. 64–79. Springer (2007)
18. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: Logic in Computer Scienc (LICS). pp. 108–117. IEEE (1990)
19. Lohstroh, M., Lee, E.A.: An interface theory for the Internet of Things. In: Software Engineering and Formal Methods (SEFM), LNCS, vol. 9276, pp. 20–34. Springer (2015)
20. Lüttgen, G., Vogler, W., Fendrich, S.: Richer interface automata with optimistic and pessimistic compatibility. Acta Informatica 52(4-5), 305–336 (2015)
21. Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fund. Inform. 108(1-2), 119–149 (2011)
22. Tripakis, S., Stergiou, C., Broy, M., Lee, E.A.: Error-completion in interface theories. In: Model Checking Software (SPIN). LNCS, vol. 7976, pp. 358–375. Springer (2013)