Value-Passing Modal Interface Automata

Sascha Fendrich

July 12, 2017

Abstract

Interface theories based on de Alfaro and Henzinger's *Interface Automata* (IA) provide a compositional, refinement-based approach to the design of concurrent software systems. However, their uniform interaction model is unsuitable for representing data when transitioning towards implementation. We extend Holík et al.'s value-passing IA, where values of local variables may be passed between components, to the richer interface theory *Error-preserving Modal Interface Automata* (EMIA). The resulting interface theory vpEMIA supports nondeterminism and modal transitions in the spirit of Larsen's *Modal Transition Systems*. We also discuss the difficulties of supporting further standard interface-theoretic operations such as hiding, quotient, conjunction and alphabet extension and conclude that the current state-of-the-art of value-passing interface theories is insufficient for supporting the full expressiveness of recent interface theories.

1 Introduction and Related Work

Interface theories combining de Alfaro and Henzinger's *Interface Automata* (IA) [7] and Larsen's *Modal Transition Systems* (MTS) [13] are a well-studied family of specification theories providing a compositional, refinement-based approach to the design of concurrent software systems, e.g., [3, 4, 7, 9, 12, 14]. Such modal interface theories usually employ a uniform interaction model where an action is considered to be an atomic unit. However, when transitioning from design to implementation, one wishes to also model data.

In this report, we address data modelling by presenting a value-passing variant of the interface theory *Error-preserving Modal Interface Automata* (EMIA) [8, 9]. As a ground semantics, we enrich EMIA by transition labels that include data values yielding the theory *EMIA with data* (EMIAD). We also define the abstract theory *value-passing EMIA* (vpEMIA) in order to be able to represent EMIADs over infinite data domains finitely. In addition, parallel composition can be computed and refinement can be checked on the abstract representation.

Value-passing is not a new idea, e.g., Cleaveland and Yankelevich have presented a value-passing variant of CCS [5]. In contrast to our goal of specifying component interfaces, Cleaveland and Yankelevich focus on providing an effective operational semantics. Hence, their work could serve as a practical implementation of our ground semantics. Our work is based on Holík et al.'s [10] value-passing version of IA. We extend their ideas to the richer interface theory EMIA, where we support must- and may-transitions, disjunctive must-transitions and nondeterminism.

A different approach to supporting data is based on shared-memory communication. This has been investigated for MTS [2], as well as for interface theories such as IA [6] and Bauer et al.'s MIO [1]. Further, behavioural type theories similar to session types [11] may also be adapted to interface theories, as we have shown in [8] for IA. Such a behavioural type theory could also be grounded with a value-passing semantics.

2 Basic Definitions

In this section we define the ground semantics *Error-preserving Modal Interface Automata with Data* (EMIAD) and the abstract representation *value-passing EMIA* (vpEMIA). Throughout the remainder of this paper, we employ the following *notation*: given sets X, Y, we write $X + Y$ for the disjoint union $(\{0\} \times X, \{1\} \times Y)$ and Y^X for the set of all functions $f : X \to Y$. We denote the powerset of a set X by $\mathfrak{P}(X)$. Further, we write $t[x/y]$ for the substitution of x for a variable y in a term t. We recap the definition of EMIA:

Definition 1 (Error-preserving Modal Interface Automaton [8, 9])**.** Let $P := (S_P, I_P, O_P, \longrightarrow_P, \dashrightarrow_P, S_P^0, U_P, E_P)$ be a tuple, where S_P is a set of states, I_P and O_P are the alphabets of *input and output names* (we define $A_P := I_P \cup O_P$), $\longrightarrow_P \subseteq S_P \times A_P \times \mathfrak{P}(S_P)$ is the *disjunctive must-transition relation*, $\dashrightarrow \subseteq S_P \times A_P \times S_P$ the *may-transition relation*, $S_P^0 \subseteq S_P$ the set of *initial states*, $U_P \subseteq S_P$ the set of *universal states* and $E \subseteq S_P$ the set of *error-states*. We call P an *Error-preserving Modal Interface Automaton* (EMIA) if

1. I_P and O_P are disjoint,
2. E_P and U_P are disjoint,
3. states in $E_P \cup U_P$ have no outgoing transitions,
4. $p \xrightarrow{a}_P P'$ implies $\forall p' \in P'. p \overset{a}{\dashrightarrow}_P p'$.

EMIAs may be extended by data values in the following way:

Definition 2 (Error-preserving Modal Interface Automaton with Data)**.** Let $\mathcal{S}$ be a set of *sorts* and $D := \sum_{\sigma \in \mathcal{S}} D_\sigma$ the disjoint union of a family $(D_\sigma)_{\sigma \in \mathcal{S}}$ of sets, called the *data domain*. Further, let I, O be disjoint sets of input and output actions. An *Error-preserving Modal Interface Automaton with Data* (EMIAD) over D, I and O is an EMIA P, where $I_P := I \times D$ and $O_P := O \times D$.

In general, the data domain D may be infinite. Therefore, we define an abstraction of EMIADs that may represent infinite EMIADs finitely. To this end, we need to define valuations and guards first.

Definition 3 (Valuations)**.** Given a data domain D and a finite set X of variables, a *valuation* is a function $v \colon X \to D$. We denote the set of valuations over D and X by D^X.

Definition 4 (Guards)**.** Let X be a finite set of variables and D a data domain. A *guard* over X and D is a Boolean expression over X, D. We write $v \vDash g$ if a guard g evaluates to true under valuation v.

Definition 5 (Value-passing EMIA)**.** Let D be a data domain, X a finite set of local variables and I, O be disjoint sets of input and output names. A *value-passing EMIA* (vpEMIA) is an EMIA P with actions of the form $[g]a\dagger\xi^\sigma; \vec{x} := \vec{e}$ where, for some $k, n \in \mathbf{N}$ and a formal parameter ξ of sort σ,

1. $a \in A$,
2. $\dagger = ?$ if $a \in I$, and $\dagger = !$ if $a \in O$,
3. g is a guard over X, D and the formal parameter ξ,
4. $\vec{x} := (x_1, \ldots, x_n)$ is a vector of (local) variables $x_1, \ldots, x_n \in X$,

Figure 1: Process P sending and process Q receiving a value on channel a.

5. $\vec{e} := (e_1, \ldots, e_n)$ is a vector of expressions over X, D and the formal parameter ξ.

We illustrate the intuition behind value-passing EMIAs by means of the example shown in Fig. 1; a formal definition is given below. Specification P in Fig. 1 models a process P that sends the value of its local variable x on channel a. The type of this value is σ. The guard $[x = \xi]$ requires that the local variable x and the formal parameter ξ evaluate to the same value; otherwise, the action $a!\xi^\sigma$ is not available. The assignment $x := x$ indicates that P's local variable x does not change when the action is performed. Specification Q models a process that is ready to receive a value of type σ on channel a and, if so, binds the received value to Q's local variable y. The guard [true] indicates that Q's readiness is unconstrained, the readiness itself is specified by action $a?\xi^\sigma$. With the assignment $y := \xi$, the received value is bound to variable y. Note that, in the tradition of interface automata, the action types input (?) and output (!) do not model the data flow direction. An output encodes that the component may actively engage in an action while an input specifies that a component is able to reactively participate in an action. The data flow direction is encoded in the guards and the assignments.

We give vpEMIA the following EMIAD semantics:

Definition 6 (Semantics of a vpEMIA)**.** A vpEMIA P over D, X, I and O represents the EMIAD $[\![P]\!]$ given as follows:

1. $S_{[\![P]\!]} := S_P \times D^X$,
2. $I_{[\![P]\!]} := I \times D$,
3. $O_{[\![P]\!]} := O \times D$,
4. $S^0_{[\![P]\!]} := S^0_P \times D^X$,
5. $(p, v) \xrightarrow{a\dagger d}_{[\![P]\!]} \bar{P}$ iff $p \xrightarrow{[g]a\dagger\xi^\sigma;\vec{x}:=\vec{e}}_P P'$, with $d \in D_\sigma$, $\bar{P} = \{(p', v') \mid p' \in P', v'(x_i) = e_i[d/\xi], i = 1, \ldots, n\}$ and $v \vDash g[d/\xi]$.

Hence, the semantics of a vpEMIA P is an EMIAD, where a state (s, v) is a pair consisting of a location $s \in S_P$ expressing the current behavioural state and a valuation $v \in D^X$ expressing the state of P's local variables. A transition is present if and only if its guard evaluates to true and the valuation changes from the source to the target state according to the assignment $\vec{x} := \vec{e}$.

3 Refinement

Concerning the ground semantics EMIAD, we directly employ EMIA-refinement that is based on modal refinement [13], as is standard in modal interface theories. Intuitively, modal refinement requires one to preserve must-transitions, whereas a may-transition in a refined interface specification must already be permitted in the coarser specification.

Definition 7 (EMIA Refinement [8, 9])**.** Let $P := (S_P, I, O, \longrightarrow_P, \dashrightarrow_P, S^0_P, U_P, E_P)$ and $Q := (S_Q, I, O, \longrightarrow_Q, \dashrightarrow_Q, S^0_Q, U_Q, E_Q)$ be EMIAs sharing the same alphabets I and O. A relation $\mathcal{R} \subseteq S_P \times S_Q$ is an EMIA *refinement relation* if, for all $(p, q) \in \mathcal{R}$,

1. $p \in E_P$ iff $q \in E_Q$,
2. $p \in U_P$ implies $q \in U_Q$,
3. $q \xrightarrow{a}_Q Q'$ implies $\exists P' \subseteq S_P.\, p \xrightarrow{a}_P P' \wedge \forall p' \in P'.\, \exists q' \in Q'.\, (p', q') \in \mathcal{R}$,
4. $p \dashrightarrow^{a}_P p'$ implies $\exists q' \in S_Q.\, q \dashrightarrow^{a}_Q q' \wedge (p', q') \in \mathcal{R}$.

We write $P \sqsubseteq Q$ if there is an EMIA refinement relation $\mathcal{R}$ such that, for all $p \in S_P^0$, there is a $q \in S_Q^0$ with $(p, q) \in \mathcal{R}$.

Such a standard notion of refinement is not computable on infinite data domains, i.e., on the abstract level of vpEMIA a more abstract definition of refinement is required:

Definition 8 (vpEMIA Refinement)**.** Let $P := (S_P, I, O, \longrightarrow_P, \dashrightarrow_P, S_P^0, U_P, E_P)$ and $Q := (S_Q, I, O, \longrightarrow_Q, \dashrightarrow_Q, S_Q^0, U_Q, E_Q)$ be vpEMIAs with equal alphabets and sets of local variables X and Y, respectively. A relation $R \subseteq S_P \times S_Q$ is a *vpEMIA refinement* relation if, for all $(p, q) \in S_P \times S_Q$,

1. $p \in E_P$ iff $q \in E_Q$,
2. $p \in U_P$ implies $q \in U_Q$,
3. $q \xrightarrow{[h]a\dagger\xi^\sigma;\vec{y}:=\vec{f}} Q'$ implies some $P', g, \vec{x}, \vec{e}$ such that
   - $p \xrightarrow{[g]a\dagger\xi^\sigma;\vec{x}:=\vec{e}} P'$,
   - $\forall d \in D_\sigma, v \in D^X, w \in D^Y.\, h[d/\xi^\sigma, w(\vec{y})/\vec{y}] \implies g[d/\xi^\sigma, v(\vec{x})/\vec{x}]$,
   - $\forall p' \in P'.\, \exists q' \in Q'.\, (p', q') \in R$,
4. $p \overset{[g]a\dagger\xi^\sigma;\vec{x}:=\vec{e}}{\dashrightarrow} p'$ implies some q', h, y, f such that
   - $q \overset{[h]a\dagger\xi^\sigma;y:=f}{\dashrightarrow} q'$,
   - $\forall d \in D_\sigma, v \in D^X, w \in D^Y.\, g[d/\xi^\sigma, v(\vec{x})/\vec{x}] \implies h[d/\xi^\sigma, w(\vec{y})/\vec{y}]$,
   - $(p', q') \in R$.

We write $P \sqsubseteq_{\mathsf{vp}} Q$ if there is a vpEMIA refinement relation R such that, for all $p \in S_P^0$, there is a $q \in S_Q^0$ with $(p, q) \in R$.

Intuitively, vpEMIA refinement mimics EMIA refinement on the abstract level by requiring the implications between guards in rules 3 and 4. We justify this intuition by proving the following proposition:

Proposition 9 (Monotonicity of Interpretation)**.** *If P, Q are vpEMIAs, then $P \sqsubseteq_{\mathsf{vp}} Q$ implies $[\![P]\!] \sqsubseteq [\![Q]\!]$.*

Proof. Must-transitions: Let $(q, v) \xrightarrow{a\dagger d}_{[\![Q]\!]} \bar{Q}$. By Def. 6, we have $q \xrightarrow{[h]a\dagger\sigma;\vec{y}:=\vec{f}} Q'$ with $d \in D_\sigma$, $v \vDash h[d/\xi^\sigma]$. Due to $P \sqsubseteq_{\mathsf{vp}} Q$ (Def. 8), there are P', g, $\vec{x}$, $\vec{e}$ such that $p \xrightarrow{[g]a\dagger\xi^\sigma;\vec{x}:=\vec{e}} P'$ and $\forall d, v, w.\, h[d/\xi^\sigma, w(\vec{y})/\vec{y}] \implies g[d/\xi^\sigma, v(\vec{x})/\vec{x}]$. Hence, $v \vDash g[d/\xi^\sigma]$ and $(p, v) \xrightarrow{a\dagger d} \bar{P} := \{(p', v') \mid p' \in P', v'(x_i) = e_i[d/\xi], i = 1, \ldots, n\}$. May-transitions: analogously, but easier due to the absence of disjunctive transitions. □

4 Parallel Composition

Interface theories permit one to compose more complex interface specifications from simpler ones by means of a parallel composition operator that represents concurrent interaction between components. As parallel composition for our ground semantics, we employ EMIA's parallel composition [8, 9], except that we merge the valuations as follows: given two sets of variables X, Y and valuations $v \in D^X$, $w \in D^Y$, the merge $v \cdot w \in D^{X \uplus Y}$ is defined as

$$v \cdot w(z) = \begin{cases} v(x) & \text{if } z = (0, x), \\ w(y) & \text{if } z = (1, y). \end{cases}$$

The basic intuition behind parallel composition is that components synchronise on shared actions while foreign action may be interleaved.

Definition 10 (Parallel Composition of EMIADs with Local Variables)**.** Let P and Q be EMIADs over variable sets X and Y, respectively. The parallel composition $P \otimes Q$ over the set of variables $X + Y$ is defined by $S_{P\otimes Q} := \{(p, q, v \cdot w) \mid (p, v) \in S_P, (q, w) \in S_Q\}$, $I_{P\otimes Q} := (I_{\mathcal{P}} \cup I_{\mathcal{Q}}) \setminus O_{P\otimes Q}$, $O_{P\otimes Q} := O_{\mathcal{P}} \cup O_{\mathcal{Q}}$, $(p, q, v) \in E_{P\otimes Q}$ iff $p \in E_P$ or $q \in E_Q$, $(p, q, v) \in U_{P\otimes Q}$ iff $p \in U_P \setminus E_P$ or $q \in U_Q \setminus E_Q$, where the transition relations are defined as follows:

1. $p \xrightarrow{a} P'$ and $a \notin A_Q$ implies $(p, q) \xrightarrow{a} P' \times \{q\}$ (and symm.)
2. $p \xrightarrow{a} P'$ and $q \xrightarrow{a} Q'$ implies $(p, q) \xrightarrow{a} P' \times Q'$ (and symm.)
3. $p \dashrightarrow^{a} p'$ and $a \notin A_Q$ implies $(p, q) \dashrightarrow^{a} (p', q)$ (and symm.)
4. $p \dashrightarrow^{a} p'$ and $q \dashrightarrow^{a} q'$ implies $(p, q) \dashrightarrow^{a} (p', q')$ (and symm.)

At the abstract level of vpEMIAs, parallel composition is defined analogously but with the following rules:

Definition 11 (Parallel Composition of vpEMIAs)**.**

1. $p \xrightarrow{[g];a!\sigma;\vec{x}:=\vec{e}} P'$ and $a \notin A_Q$ implies $(p, q) \xrightarrow{[g];a!\sigma;\vec{x},\vec{y}:=\vec{e},\vec{y}} P' \times \{q\}$ (and symm.)
2. $p \xrightarrow{[g];a!\sigma;\vec{x}:=\vec{e}} P'$ and $q \xrightarrow{[h];a?\sigma;\vec{y}:=\vec{f}} Q'$ implies $(p, q) \xrightarrow{[g\wedge h];a!\sigma;\vec{x},\vec{y}:=\vec{e},\vec{f}} P' \times Q'$ (and symm.)
3. $p \overset{[g];a!\sigma;\vec{x}:=\vec{e}}{- - - \Rightarrow} p'$ and $a \notin A_Q$ implies $(p, q) \overset{[g];a!\sigma;\vec{x},\vec{y}:=\vec{e},\vec{y}}{- - - - - \Rightarrow} (p', q)$ (and symm.)
4. $p \overset{[g];a!\sigma;\vec{x}:=\vec{e}}{- - - \Rightarrow} p'$ and $q \overset{[h];a?\sigma;\vec{y}:=\vec{f}}{- - - - \Rightarrow} q'$ implies $(p, q) \overset{[g\wedge h];a!\sigma;\vec{x},\vec{y}:=\vec{e},\vec{f}}{- - - - - - \Rightarrow} (p', q')$ (and symm.)

It is easy to see that interpretation is homomorphic wrt. parallel composition:

Proposition 12 (Homomorphicity of Interpretation)**.** *If P and Q are composable vpEMIAs, then $[\![P \otimes Q]\!] = [\![P]\!] \otimes [\![Q]\!]$.*

Proof sketch. A transition exists in P if g evaluates to true, and in Q if h evaluates to true. Hence, two transitions synchronise if and only if $g \wedge h$ evaluates to true, i.e., the abstract parallel composition corresponds to parallel composition of EMIADs. □

As a consequence, parallel composition is compositional.

Proposition 13 (Compositionality)**.** *Let P_1, P_2, Q be vpEMIAs with $P_1 \sqsubseteq_{\mathsf{vp}} P_2$. If $P_2 \otimes Q$ is defined, then $P_1 \otimes Q$ is defined and $P_1 \otimes Q \sqsubseteq_{\mathsf{vp}} P_2 \otimes Q$.*

Proof. Let P_1, P_2, Q be vpEMIAs such that $P_2 \otimes Q$ is defined and $P_1 \sqsubseteq_{\mathsf{vp}} P_2$. By Prop. 9, $[\![P_1]\!] \sqsubseteq [\![P_2]\!]$. Prop. 12 and compositionality of parallel composition for EMIA [8, 9] implies $[\![P_1 \otimes Q]\!] = [\![P_1]\!] \otimes [\![Q]\!] \sqsubseteq [\![P_2]\!] \otimes [\![Q]\!] = [\![P_2 \otimes Q]\!]$. □

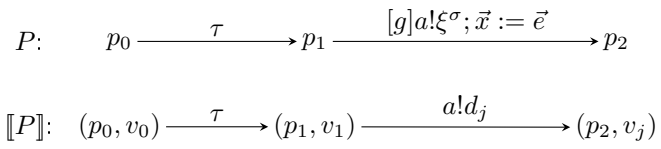

Figure 2: How to define weak transitions for value-passing interfaces?

5 Discussion and Future Work

With parallel composition and refinement we only discussed the two core concepts of interface theories. However, modern interface theories such as EMIA [8, 9] support several additional features and operators, which we briefly discuss in this section without giving a formal definition, namely internal transitions, hiding, conjunction, quotient and alphabet extension.

Following the related work [1, 2, 5, 10] we have not included internal transitions so far. Interface theories supporting internal behaviour include an additional internal action τ and a hiding operator that permits one to internalise transitions. The refinement preorder enables one to abstract from such internal behaviour by applying the simulation rules to weak transitions $\stackrel{a}{\Rightarrow}$ that abstract from leading and trailing τs. The example given in Fig. 2 shows that it is not obvious how to define weak transitions for value-passing interfaces. The figure depicts a vpEMIA P and its EMIAD semantics $[\![P]\!]$, where we have transitions $(p_1, v_1) \xrightarrow{a!d_j} (p_2, v_j)$ for some index set J with $j \in J$ if and only if $v_1 \vDash g[d_j/\xi]$ and $v_j = v_1[\vec{e}[d_j/\xi]/\vec{x}]$. Standard definitions of weak transitions would permit one to abstract the two transitions in P to a weak transition $p_0 \xRightarrow{[g]a!\xi^\sigma;\, \vec{x}:=\vec{e}} p_2$. However, the semantics $[\![P]\!]$ depends on the valuation v_1 of the intermediate state that is abstracted away in the weak transition. In particular, an arbitrary change is possible when following an internal transition, e.g., when transitioning from v_0 to v_1. It is unclear how to include this additional information in a weak transition. Similarly, when hiding action a in P, all information about guards and updates of a-transitions is lost in the resulting specification $P/\{a\}$. As a consequence, the EMIAD semantics $[\![P/\{a\}]\!]$ of this result is unclear. Due to the complications described in this subsection, we leave hiding and internal behaviour to future work.

Interface theories support component reuse and synthesis by means of a quotient operator. Given a global specification G and an already implemented component C, the quotient $G /\!\!/ C$ is the coarsest specification that complements C wrt. parallel composition, i.e., $X \sqsubseteq G /\!\!/ C \iff X \otimes C \sqsubseteq G$. Holík et al. [10] present a quotient for their value-passing interface automata. At the current state of our vpEMIA, we did not adapt their work but leave quotienting for future investigations due to the following reasons. Firstly, Holík et al. do not prove that their quotient is in fact an adjoint of parallel composition. Secondly, the quotient for value-passing IA is significantly simpler than for vpEMIA, because vpEMIA supports modalities and nondeterminism. In particular, nondeterministic quotients supporting internal behaviour are still an open problem in interface theories [8].

Conjunction is an important operation of interface theories, which permits one to express that a component shall implement several interfaces: if C is required to implement the interfaces P and Q, the conjunction $P \sqcap Q$ specifies an overall interface expressing this requirement. Hence, conjunction must be the greatest lower bound wrt. the refinement preorder. It is obvious that the EMIAD semantics is finer than vpEMIA, i.e., although the interpretation function $[\![\cdot]\!]$ is monotonic, it is not continuous in general. As a consequence, conjunction operators for vpEMIA and EMIAD do not coincide, and it is debatable whether such an operator may really be considered as conjunction. For future work, we wish to investigate the relation between vpEMIA refinement

and EMIAD refinement in order to get a better understanding of conjunction for value-passing.

Alphabet extension enables one to adapt a specification to new operational environments by permitting the implementation of new features. This is particularly useful in combination with conjunction for not having to require the conjuncts to share the same alphabets, when alphabet extension is available. The intuition behind alphabet extension is that a specification is unspecific wrt. new actions but must satisfy the same requirements before and after a new action is performed. Hence, the alphabet extension of a specification P by a set of new actions A with $A \cap A_P = \emptyset$ is obtained from P by adding a may-loop for each $a \in A$ to each state $p \in S_P$. In a value-passing interface theory, such a loop must be added for every type σ and every possible assignment $\vec{x} := \vec{e}$; this results in an infinite number of new loops, which is impractical. Hence, a different representation of value-passing interfaces is required if one wishes to support alphabet extension.

In addition, we plan to compare value-passing interface theories with shared memory communication as is employed, e.g., in [1, 2]. Further, we have developed interface automata into a behavioural type theory [8], providing a different approach to modelling data. Such a behavioural type theory may also be grounded with a semantics similar to EMIAD and, thus, compared with value-passing interface theories. From such a comparison, we expect to advance both the value-passing and the behavioural type theories and to maybe find a suitable combination that solves some of the above mentioned problems.

References

[1] Sebastian S. Bauer, Rolf Hennicker, and Martin Wirsing. Interface theories for concurrency and data. *Theoretical Computer Science*, 412(28):3101–3121, 2011.

[2] Sebastian S. Bauer, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wąsowski. A modal specification theory for components with data. In *Formal Aspects of Component Software (FACS)*, LNCS, pages 61–78. Springer, 2012.

[3] Sebastian S. Bauer, Philip Mayer, Andreas Schroeder, and Rolf Hennicker. On weak modal compatibility, refinement, and the MIO Workbench. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6015 of *LNCS*, pages 175–189. Springer, 2010.

[4] Ferenc Bujtor, Sascha Fendrich, Gerald Lüttgen, and Walter Vogler. Nondeterministic modal interfaces. *Theoretical Computer Science*, 642:24–53, 2016.

[5] Rance Cleaveland and Daniel Yankelevich. An operational framework for value-passing processes. In *Principles of Programming Languages (POPL)*, pages 326–338. ACM, 1994.

[6] Luca de Alfaro, Leandro D. da Silva, Marco Faella, Axel Legay, Pritam Roy, and Maria Sorea. Sociable interfaces. In *Frontiers of Combining Systems (FroCoS)*, volume 3717 of *LNAI*, pages 81–105. Springer, 2005.

[7] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Foundations of Software Engineering (FSE)*, ESEC/FSE-9, pages 109–120. ACM, 2001.

[8] Sascha Fendrich. *Modal Interface Theories for Specifying Component-based Systems.* PhD thesis, Univ. Bamberg, Germany, 2017.

[9] Sascha Fendrich and Gerald Lüttgen. A generalised theory of interface automata, component compatibility and error. In *Integrated Formal Methods (iFM)*, volume 9681 of *LNCS*, pages 160–175. Springer, 2016.

[10] Lukáš Holík, Malte Isberner, and Bengt Jonsson. Mediator synthesis in a component algebra with data. In *Correct System Design*, volume 9360 of *LNCS*, pages 238–259. Springer, 2015.

[11] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM*, 63(1):9:1–9:67, 2016.

[12] Kim G. Larsen, Ulrik Nyman, and Andrzej Wąsowski. Modal I/O automata for interface and product line theories. In *Programming Languages and Systems (ESOP)*, volume 4421 of *LNCS*, pages 64–79. Springer, 2007.

[13] Kim G. Larsen and Bent Thomsen. A modal process logic. In *Logic in Computer Science (LICS)*, pages 203–210. IEEE, 1988.

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