On Relational Interfaces

Stavros Tripakis
Ben Lickly
Thomas A. Henzinger
Edward A. Lee

May 10, 2009

On Relational Interfaces[*]

Stavros Tripakis, UC Berkeley
Ben Lickly, UC Berkeley
Thomas A. Henzinger, EPFL
Edward A. Lee, UC Berkeley

May 8, 2009

Abstract

In this paper we extend the work of De Alfaro, Henzinger et al, on interface theories for component-based design. Existing interface theories fail to capture functional relations between the inputs and outputs of an interface. For example, a simple interface that takes as input a number $n \geq 0$ and returns as output $n+1$, cannot be expressed in existing theories. In this paper we provide a theory of relational interfaces, where such input-output relations can be captured. Our theory supports both stateless and stateful interfaces, includes explicit notions of environments and pluggability, and satisfies fundamental properties such as preservation of refinement by composition, and characterization of pluggability by refinement. We achieve these properties by making reasonable restrictions on feedback loops in interface compositions.

1 Introduction

Component-based design has emerged as a significant challenge in building complex systems, such as embedded, cyber-physical systems, in an efficient and reliable manner. The size and complexity of such systems prohibits designing an entire system from scratch, or building it as a single unit. Instead, the system must be designed as a set of components, some built from scratch, some inherited by legacy. *Interfaces* play a key role in component-based design, as they provide the means to reason about components. An interface can be seen as an abstraction of a component: on one hand, such an abstraction captures information that is essential in order to use the component in a given context; on the other hand, the abstraction hides unnecessary information, making reasoning simpler and more efficient.

Significant progress has been made in the past several years toward the development of a comprehensive theory of interfaces for component-based design. Such a theory has been pioneered and developed in a series of papers by De Alfaro, Henzinger et al (see [4, 3, 2, 5] for a sample). What has been elusive, however, is a theory of *relational interfaces*, that is, interfaces that specify relations between inputs and outputs. Consider, for example, a component that is supposed to take as input a number $n \geq 0$ and return as output $n+1$. The interface for such a component can be described as a binary relation between the input and the output: the relation containing all pairs $(n, n+1)$, such that $n \geq 0$. Such a relation can be seen as a *contract* between the component and its environment: the contract specifies the *legal* inputs that the environment is allowed to provide to the component (in this case $n \geq 0$); and for every legal input, what are the legal outputs that the component may produce when fed with that input.

Existing interface theories, in particular those proposed in the aforementioned works, only partially capture relational interfaces. [4] defines *relational nets*, which are networks of processes that non-deterministically

[*]This work is supported by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CSR-EHS: PRET) and #0720841 (CSR-CPS)), the U.S. Army Research Office (ARO #W911NF-07-2-0019), the U.S. Air Force Office of Scientific Research (MURI #FA9550-06-0312), the Air Force Research Lab (AFRL), the State of California Micro Program, and the following companies: Agilent, Bosch, Lockheed-Martin, National Instruments, Thales and Toyota. Authors' emails: {stavros,blickly,eal}@eecs.berkeley.edu, tah@epfl.ch.

relate input values to output values (these form essentially the subclass of *stateless* relational interfaces). [4] does not provide an interface theory for the complete class of relational nets. Instead they provide interface theories for subclasses, in particular: *rectangular nets* which have no input-output dependencies; *total nets* which can have input-output dependencies but make no restrictions on the inputs (in this paper we call such interfaces *input-complete*); and *total and rectangular nets* which combine both restrictions above.

The interfaces provided in [4] for rectangular nets are called *stateless A/G interfaces.* These interfaces separate the assumptions on the inputs from the guarantees on the outputs, and as such cannot capture input-output relations.

The interface theory developed in [5] is also based on A/G interfaces (both stateless and *stateful*). In a Discussion section, [5] briefly discuss *extended interfaces* which relate input and outputs (these are the same as the relational interfaces that we study in this paper). [5] conclude that extended interfaces are not appropriate, because "the basic properties of stepwise refinement and independent implementability do not hold in the extended framework" [5].

[2] consider *Moore interfaces*, defined by a formula ϕ_i that specifies the legal values of the input variables at the *next* state, given the current state, and a formula ϕ_o that specifies the legal values of the output variables at the *next* state, given the current state. This formulation does not allow to describe relations between inputs and outputs at the *same* state, thus fails to capture general relational interfaces.

Both [4] and [5] can handle very general compositions of interfaces, that can be obtained via two operators, namely, *parallel composition* and *connection* (this is similar to the denotational framework of [6]). This allows, in particular, arbitrary *feedback loops* to be created.

Feedback loops are a major source of problems when studying relational interfaces. To illustrate some of the problems that arise, consider the following example, borrowed from [5]. Suppose I_{true} is an interface on input x and output y, with trivial contract true, making no assumptions on the inputs and no guarantees on the outputs. Suppose $I_{y\neq x}$ is another interface on x and y, with contract $y \neq x$, meaning that it guarantees the value of the output will be different from the value of the input. According to standard definitions of refinement, $I_{y\neq x}$ refines I_{true}: this is because $I_{y\neq x}$ is "more deterministic" than I_{true} (the output guarantees of $I_{y\neq x}$ are stronger). Now, consider the feedback connection $x = y$. This could be considered an allowed connection for I_{true}, since it does not contradict its contract. But the same connection contradicts the contract of $I_{y\neq x}$. As a result, even though $I_{y\neq x}$ refines I_{true}, the feedback composition of $I_{y\neq x}$ does not refine the feedback composition of I_{true}. This means that one of the fundamental properties that an interface theory should provide, namely, that composition preserves refinement, would not hold in this case.

In this paper we propose a theory of relational interfaces. Our theory relies on a notion of refinement which is *input-contravariant* like the relations proposed in [1, 4, 5], but not strictly *output-covariant.* We avoid problems created by feedback loops by restricting the cases in which feedback loops are allowed. In particular, we allow an output of an interface I to be connected to one of its inputs x only if I is *Moore with respect to* x, meaning that the contract of I does not depend on x(note that this definition is less restrictive than the one of [2]).

Arguably, this is a reasonable restriction in practice. After all, feedback loops generally create *causality cycles* that result in ambiguous semantics. In many languages and tools these problems are avoided by making restrictions similar to (in fact, stricter than) ours. For example, tools such as Simulink from The MathWorks [1] or SCADE from Esterel Technologies [2] often require a *unit-delay* block to be placed in every feedback loop.[3] This restriction does not appear to result in a significant loss of expressiveness in practice.

Using the above assumption, we are able to derive a comprehensive theory of relational interfaces that supports *stepwise refinement* (if I' refines I then I' can replace I in any context), *independent implementability* (if components I'_i refine components I_i then the composition of I'_i refines the composition of I_i) and *component reuse* (via *shared refinement*, see below) [5]. Other properties of our theory and contributions of this paper include the following:

[1] www.mathworks.com/products/simulink/

[2] www.esterel-technologies.com/products/scade-suite/

[3] Simulink provides the user with the option to ignore so-called *algebraic loops* but this results in ambiguous (non-deterministic) semantics.

We explicitly introduce the notions of *environments* and *pluggability* of interfaces to environments. We also introduce two notions of *equivalence* between interfaces: equivalence of their contracts, and equivalence with respect to environments (two interfaces are equivalent iff they can be plugged to the same set of environments). We show that these two equivalences coincide. We also prove that our notion of refinement characterizes pluggability in the following fundamental way: interface I' refines I iff I' can replace I in every context (i.e., environment). (Note that this is a stronger property than stepwise refinement: it is an *iff* instead of an *only if*.) We show by example that alternative definitions of refinement do not have this property.

We seamlessly handle stateless and stateful interfaces. Stateful interfaces associate a potentially different contract at every state. We model a state very simply and generally, as a history of input/output values. This allows us to handle finite as well as infinite-state interfaces. Stateless interfaces can be viewed as a special case of stateful interfaces where the contract is the same at all states.

In the stateful case, we distinguish between *well-formed* and *well-formable* interfaces (the two notions coincide for stateless interfaces). Well-formed interfaces are such that their contract can always be satisfied at every reachable state. Well-formable interfaces are not necessarily well-formed, but can be made well-formed by appropriately restricting their inputs. Refinement preserves well-formability always, but it only preserves well-formedness under certain conditions. As observed in [4], *controller-synthesis* type of procedures can be used to transform finite-state well-formable interfaces into well-formed ones.

We propose a *hiding* operator that allows removal of a subset of the output variables in a contract by projecting them out. This is useful when composing interfaces, where often many variables end up being equal. Hiding is always possible for stateless interfaces, and corresponds to existentially quantifying outputs in the contract. The situation is more subtle in the stateful case, where we need to ensure that the "hidden" variables do not influence the evolution of the contract.

Our theory supports *shared refinement* of two interfaces I and I', which is important for component reuse, as argued in [5]. A shared refinement operator $I \sqcap I'$ is proposed in the Discussion section of [5] for extended (i.e., relational) interfaces, and it is conjectured that this operator represents the greatest lower bound with respect to refinement. We show that this holds only if a *shared refinability* condition is imposed. This condition states that for every inputs that is legal in both I and I', the corresponding sets of outputs of I and I' must have a non-empty intersection.

2 Preliminaries, notation

In this paper we use first-order logic (FOL) as a language to describe contracts.[4] For an introduction to FOL, see, for instance, [8]. We use true and false for logical constants true and false, $\neg, \wedge, \vee, \rightarrow, \equiv$ for logical negation, conjunction, disjunction, implication, and equivalence, and $\exists$ and $\forall$ for existential and universal quantification, respectively. We will use $:=$ when defining concepts or introducing new notation, for instance, $x_0 := \max\{1, 2, 3\}$ defines x_0 to be the maximum of the set $\{1, 2, 3\}$.

Let V be a finite set of variables. A *property over* V is a FOL formula ϕ such that any free variable of ϕ is in V. The set of all properties over V is denoted $\mathcal{F}(V)$. Let ϕ be a property over V and V' be a finite subset of V, $V' = \{v_1, v_2, ..., v_n\}$. Then, $\exists V' : \phi$ is shorthand for $\exists v_1 : \exists v_2 : ... : \exists v_n : \phi$. Similarly, $\forall V' : \phi$ is shorthand for $\forall v_1 : \forall v_2 : ... : \forall v_n : \phi$.

We will implicitly assume that all variables are *typed*, meaning that every variable is associated with a certain *domain*. An *assignment* over a set of variables V is a (total) function mapping every variable in V to a certain value in the domain of that variable. The set of all assignments over V is denoted $\mathcal{A}(V)$. If a is an assignment over V_1 and b is an assignment over V_2, and V_1, V_2 are disjoint, we use (a, b) to denote the combined assignment over $V_1 \cup V_2$. A formula ϕ is *satisfiable* iff there exists an assignment a over the free variables of ϕ such that a satisfies ϕ, denoted $a \models \phi$. A formula ϕ is *valid* iff it is satisfied by every assignment.

[4] As mentioned in the introduction, contracts are essentially relations between inputs and outputs. Our theory holds for such relations, independently from how the relations are specified. FOL formulae is one possible language, but other languages could be used as well.

If S is a set, S^* denotes the set of all finite sequences made up of elements in S. S^* includes the empty sequence, denoted ε. If $s, s' \in S^*$, then $s \cdot s'$ is the concatenation of s and s'. $|s|$ denotes the *length* of $s \in S^*$, with $|\varepsilon| = 0$ and $|s \cdot a| = |s| + 1$, for $a \in S$. If $s = a_1 a_2 \cdots a_n$, then the i-th element of the sequence, a_i, is denoted s_i, for $i = 1, ..., n$.

3 Relational Interfaces

Definition 1 (Relational interface) *A* relational interface *(or simply* interface*) is a tuple $I = (X, Y, \xi)$ where X and Y are two finite and disjoint sets of* input *and* output variables*, respectively, and ξ is a total function*

$$\xi : \mathcal{A}(X \cup Y)^* \to \mathcal{F}(X \cup Y)$$

Recall that $\mathcal{A}(V)$ is the set of all assignments over set of variables V. Therefore, $\mathcal{A}(X \cup Y)^*$ is the set of all finite sequences of assignments over $X \cup Y$. Note that we allow X or Y to be empty: if X is empty then I is a *source* interface; if Y is empty then I is a *sink*. An element of $\mathcal{A}(X \cup Y)^*$ is called a *state*. The *initial state* is the empty sequence ε. Recall that $\mathcal{F}(X \cup Y)$ is the set of all properties over $X \cup Y$. Therefore, ξ associates with every state s a formula $\xi(s)$ over $X \cup Y$. This formula acts as the contract between I and its environment *at that state*. The contract changes dynamically, as the state of I changes.

Suppose the current state of I is s and the environment presents I with an assignment a_X over the input variables X, which satisfies the input assumptions $\mathsf{in}(\xi(s))$. I then chooses an assignment a_Y over the output variables Y, such that together the two assignments satisfy $\xi(s)$. The combined assignments yield an assignment over $X \cup Y$, $a := (a_X, a_Y)$. The new state of I is $s \cdot a$.

A *stateless* interface is one where the contract is independent of the state:

Definition 2 (Stateless interface) *An interface $I = (X, Y, \xi)$ is* stateless *if for all $s, s' \in \mathcal{A}(X \cup Y)^*$, $\xi(s) = \xi(s')$.*

For a stateless interface, we can treat ξ as a property, instead of as a function that maps states to properties. For clarify, we will write $I = (X, Y, \phi)$ for a stateless interface, where ϕ is a property over $X \cup Y$.

Example 1 *Consider a component which is supposed to take as input a positive number n and return n or $n+1$ as output. We can capture such a component in different ways. One way is by the following stateless interface:*

$$I_1 := (\{x\}, \{y\}, x > 0 \wedge (y = x \vee y = x + 1)\}).$$

Here, x is the input variable and y is the output variable. The contract of I_1 explicitly forbids zero or negative values for x. Another possible stateless interface for this component is:

$$I_2 := (\{x\}, \{y\}, x > 0 \to (y = x \vee y = x + 1)\}).$$

The contract of I_2 is different from that of I_1: it allows $x \leq 0$, but makes no guarantees about the output y in that case. I_2 is an input-complete *interface, in the sense that it accepts all inputs. Input-complete interfaces are discussed in detail in Section 9.*

In general, the state space of an interface is infinite. In some cases, however, only a finite set of states is needed to specify ξ. For example ξ may be specified by a finite-state automaton, as in [5]. Every state of the automaton is labeled with a contract. Every transition of the automaton is labeled with a *guard*, i.e., a condition on the input and output variables. Outgoing transitions from a state must have disjoint guards (for determinism) and the union of such guards must be `true` (for absence of deadlocks). An interface that can be specified as a finite-state automaton is called a *finite-state interface*.

Definition 3 (Assumptions, guarantees) *Given a contract $\phi \in \mathcal{F}(X \cup Y)$, the* input assumption *of ϕ is the formula $\mathsf{in}(\phi) := \exists Y : \phi$. The* output guarantee *of ϕ is the formula $\mathsf{out}(\phi) := \exists X : \phi$.*

Note that $\mathsf{in}(\phi)$ is a property over X and $\mathsf{out}(\phi)$ is a property over Y. When ϕ is the contract of a stateless interface I, we write $\mathsf{in}(I), \mathsf{out}(I)$ instead of $\mathsf{in}(\phi), \mathsf{out}(\phi)$. For example, for the interfaces I_1 and I_2 of Example 1, we have $\mathsf{in}(I_1) \equiv x > 0$ and $\mathsf{in}(I_2) \equiv \mathsf{true}$. Note that $\phi \to \mathsf{in}(\phi)$ and $\phi \to \mathsf{out}(\phi)$ are valid formulae for any ϕ.

The A/G interfaces considered in [3, 5] are a special case of the relational interfaces that we consider in this paper. A stateless A/G interface is a tuple (X, Y, ϕ_X, ϕ_Y), where ϕ_X is a property on X representing the input assumptions and ϕ_Y is a property on Y representing the output guarantees. This interface can simply be represented as the relational interface $(X, Y, \phi_X \wedge \phi_Y)$.

Definition 1 allows for the contract $\xi(s)$ at a certain state s to be an unsatisfiable property. On the other hand, not all such states may generally be *reachable*, because not all inputs or outputs are legal. We only care about states with unsatisfiable contracts when these states are reachable. Let us define reachable states formally.

A *run* of I is a state $s = a_1 \cdots a_k$, with $k \geq 0$ (if $k = 0$ then $s = \varepsilon$), such that $\forall i \in \{1, ..., k\} : a_i \models \xi(a_1 \cdots a_{i-1})$. A state is *reachable* iff it is a run. The set of reachable states of I is denoted $\mathcal{R}(I)$. By definition, $\varepsilon \in \mathcal{R}(I)$, for any I.

Definition 4 (Well-formed interface) *An interface $I = (X, Y, \xi)$ is* well-formed *iff for all $s \in \mathcal{R}(I)$, $\xi(s)$ is satisfiable.*

Some interfaces, even though they are not well-formed, can be turned into well-formed interfaces by appropriately restricting their inputs, as the following example shows.

Example 2 *Let $I := (\{x\}, \{y\}, \xi)$ where x, y are implicitly considered to be Booleans, and $\xi(\varepsilon) := \mathsf{true}$, $\xi((x, \_) \cdot s) := \mathsf{false}$, $\xi((\neg x, \_) \cdot s) := \mathsf{true}$, for all s. $(x, \_)$ denotes any assignment where x is* true *and $(\neg x, \_)$ denotes any assignment where x is* false. *I is not well-formed, because it has reachable states with contract* false *(all states starting with x being* true*). I can be transformed into a well-formed interface by restricting $\xi(\varepsilon)$ so that all reachable states with unsatisfiable contracts are avoided. In particular, setting $\xi(\varepsilon) := \neg x$, achieves this goal.*

Motivated by the above example, we introduce a weaker notion of well-formedness:

Definition 5 (Well-formable interface) *An interface $I = (X, Y, \xi)$ is* well-formable *if there exists a well-formed interface $I' = (X, Y, \xi')$ (called a* witness*) such that: for all $s \in \mathcal{R}(I')$, $\xi'(s) \equiv \xi(s) \wedge \phi_s$, where ϕ_s is some property over X.*

Clearly, every well-formed interface is well-formable, but the opposite is not true as Example 2 shows. For stateless interfaces, the two notions coincide, however.

Theorem 1 *A stateless interface I is well-formed iff it is well-formable.*

Proof: Suppose $I = (X, Y, \xi)$ is a stateless interface. Well-formed implies well-formable for all interfaces. Suppose $I = (X, Y, \xi)$ is well-formable. Then there exists $I' = (X, Y, \xi')$ such that I' is well-formed and $\xi'(\varepsilon) = \xi(\varepsilon) \wedge \phi_\varepsilon$, for some ϕ_ε. Since I' is well-formed, $\xi'(\varepsilon)$ is satisfiable. Therefore, $\xi(\varepsilon)$ is also satisfiable. Since I is stateless, $\xi(s) = \xi(\varepsilon)$ for any s, thus, $\xi(s)$ is satisfiable for any s. Therefore, I is well-formed. ■

For a finite-state interface, there exists a procedure to check whether it is well-formable, and if this is the case, transform it into a well-formed interface. Such a procedure essentially attempts to find a winning strategy in a *game*, as pointed out in [3]. Roughly speaking, the procedure consists in recursively marking states as *illegal*, until no more states can be marked. Initially, all states s such that $\xi(s)$ is unsatisfiable are marked as illegal. Then, repeatedly, a state s is marked illegal if there exists no *legal input assignment* at s. A legal input assignment at s is an assignment a to input variables, such that for any assignment b to output variables, if $(a, b) \models \xi(s)$ then the successor state $s \cdot (a, b)$ is legal. If at the end of this operation the initial state is marked illegal, then the interface is not well-formable, otherwise it is. During the procedure, the contract $\xi(s)$ of a legal state s can be restricted to allow only legal input assignments.

Definition 6 (Equivalence) *Interfaces $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$ are* equivalent*, denoted $I \equiv I'$, if $X = X'$, $Y = Y'$, and for all $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, the formula $\xi(s) \equiv \xi'(s)$ is valid.*

Lemma 1 *If $I \equiv I'$ then $\mathcal{R}(I) = \mathcal{R}(I')$.*

Proof: By induction on the length of states. The result holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose the result holds for a given state s. We prove it for $s \cdot a$. Let $s \cdot a \in \mathcal{R}(I)$. This means that the assignment a satisfies the contract of I at state s, that is, $a \models \xi(s)$. $s \cdot a \in \mathcal{R}(I)$ implies $s \in \mathcal{R}(I)$. By the induction hypothesis, $s \in \mathcal{R}(I')$, therefore, $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. This and the hypothesis $I \equiv I'$ imply $\xi(s) \equiv \xi'(s)$. Therefore, $a \models \xi'(s)$, thus, $s \cdot a \in \mathcal{R}(I')$. ■

4 Environments, pluggability

Definition 7 (Environment) *An* environment *is a tuple $E = (X, Y, h_X, h_Y)$, where X and Y are as in Definition 1, and h_X, h_Y are total functions*

$$h_X : \mathcal{A}(X \cup Y)^* \to \mathcal{F}(X), \qquad h_Y : \mathcal{A}(X \cup Y)^* \to \mathcal{F}(Y)$$

h_X represents the guarantees on the inputs X that the environment *provides* at a given state. h_Y represents the guarantees that the environment *expects* on the outputs Y. See Definition 8 that follows.

States are defined for environments in the same way as for interfaces. A *stateless environment* is one where $h_X(s)$ and $h_Y(s)$ are constant for all states s.

Definition 8 (Pluggability) *Interface $I = (X', Y', \xi)$ is* pluggable *to environment $E = (X, Y, h_X, h_Y)$, denoted $I \models E$, iff $X' = X$, $Y' = Y$, and for all $s \in \mathcal{R}(I_E)$, the following formula is valid:*

$$h_X(s) \to \big(\mathsf{in}(\xi(s)) \wedge (\xi(s) \to h_Y(s))\big) \tag{1}$$

where I_E is the interface defined as follows:

$$I_E \ := \ (X, Y, \xi_E) \tag{2}$$

$$\xi_E(s) \ := \ \xi(s) \wedge h_X(s), \ \textit{for any } s \in \mathcal{A}(X \cup Y)^* \tag{3}$$

Pluggability can be intuitively seen as a game between the interface and the environment [4]. Suppose s is the current state of I and E (initially, $s = \varepsilon$). If $h_X(s)$ is unsatisfiable, then E decides to stop the game. Otherwise, E chooses some input assignment a_X satisfying $h_X(s)$. If a_X violates $\mathsf{in}(\xi(s))$, then Condition (1) is violated, and I is not pluggable to E: the "blame" here is on E, which provides too weak guarantees on the inputs. Otherwise, I chooses an output assignment a_Y such that the input-output assignment $a := (a_X, a_Y)$ satisfies $\xi(s)$. If a_Y violates $h_Y(s)$), then Condition (1) is violated, which again means I is not pluggable to E: in this case the "blame" is on I, which provides too weak guarantees on the outputs. Otherwise, a round is complete, and the new state (for both I and E) is $s \cdot a$. The game continues in the same manner.

Example 3 *Consider stateless interfaces I_1 and I_2 from Example 1 and stateless environment $E_1 := (\{x\}, \{y\}, x > 0, y > 0)$. It can be seen that both I_1 and I_2 are pluggable to E_1. On the other hand, none of I_1, I_2 are pluggable to $E_2 := (\{x\}, \{y\}, x \geq 0, y > 0)$: the constraint $x \geq 0$ is not strong enough to meet the input assumption $x > 0$. Notice that I_2 does not explicitly impose this input assumption, however, it implicitly does, because it makes no guarantees on the outputs when $x > 0$ is violated. Finally, consider $E_3 := (\{x\}, \{y\}, \mathsf{true}, \mathsf{true})$. I_2 is pluggable to E_3: E_3 provides no guarantees on the inputs, but expects no guarantees on the outputs either. I_1 is not pluggable to E_3 because $\mathsf{true} \not\to x > 0$.*

The interface I_E defined by (2) and (3) is intended to capture the reachable states of the "closed-loop" composition of E and I. These reachable states are a subset of the reachable states of I, because the environment E may in general provide only a restricted set of inputs, among all possible legal inputs for I. The contract function ξ_E of I_E captures exactly that. The following lemma states that the reachable states of I_E are indeed a subset of those of I.

Lemma 2 *Let I be an interface, E be an environment, and I_E be defined as in Definition 8. Then, $\mathcal{R}(I_E) \subseteq \mathcal{R}(I)$.*

Proof: Follows from the fact that ξ_E strengthens ξ. ■

Lemma 3 *Let I, I' be interfaces, E be an environment, and I_E, I'_E be defined as in Definition 8. If $I \equiv I'$ then $\mathcal{R}(I_E) = \mathcal{R}(I'_E)$.*

Proof: By induction on the length of states. The result holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose the result holds for a given state s. We prove it for $s \cdot a$. Let $s \cdot a \in \mathcal{R}(I_E)$. This means that $s \in \mathcal{R}(I_E)$ and the assignment a satisfies the contract of I_E at state s, that is, $a \models \xi(s) \wedge h_X(s)$. By Lemma 2, $\mathcal{R}(I_E) \subseteq \mathcal{R}(I)$, therefore, $s \in \mathcal{R}(I)$. By Lemma 1, $\mathcal{R}(I) = \mathcal{R}(I')$, therefore, $s \in \mathcal{R}(I')$. This and $I \equiv I'$ imply $\xi(s) \equiv \xi'(s)$. Therefore, $a \models \xi'(s) \wedge h_X(s)$. By the induction hypothesis, $s \in \mathcal{R}(I'_E)$. The latter two facts imply $s \cdot a \in \mathcal{R}(I'_E)$. ■

Theorem 2 *If an interface I is well-formable then there exists an environment E such that $I \models E$.*

Proof: Let $I = (X, Y, \xi)$ and suppose I is well-formable. Then there exists $I' = (X, Y, \xi')$ such that I' is well-formed, and for all $s \in \mathcal{R}(I')$, $\xi'(s) \equiv \xi(s) \wedge \phi_s$, where ϕ_s is some property over X. Define $\psi_s := \xi(s) \wedge \phi_s$. and $E := (X, Y, h_X, h_Y)$ such that $h_X(s) := \mathsf{in}(\xi(s)) \wedge \phi_s$ and $h_Y(s) := \psi_s$, for all $s \in \mathcal{A}(X \cup Y)^*$. We claim that $I \models E$.

To prove the claim, we need to show that for all $s \in \mathcal{R}(I_E)$, Formula (1) is valid. Clearly, $h_X(s) \rightarrow \mathsf{in}(\xi(s))$. We need to prove $h_X(s) \rightarrow (\xi(s) \rightarrow \psi_s)$, i.e., $(h_X(s) \wedge \xi(s)) \rightarrow (\xi(s) \wedge \phi_s)$, which holds by definition of $h_X(s)$. ■

Note that since well-formed implies well-formable, a corollary of Theorem 2 is that every well-formed interface can be plugged to some environment. The converse does not generally hold. That is, there exist non-well-formed interfaces that can be plugged to environments: these environments restrict the inputs, so states with unsatisfiable contracts are never reached. In fact, there exist non-well-formable interfaces that can be plugged to environments as well: these environments "stop" after some point, i.e., are such that $h_X(s) \equiv \mathsf{false}$ for some state s. This holds even for non-well-formed stateless interfaces, which can be plugged into the trivial environment that stops immediately (i.e., such that $h_X(\varepsilon) \equiv \mathsf{false}$).

Definition 9 (Equivalence w.r.t. environments) *Two interfaces I and I' are* equivalent w.r.t. environments, *denoted $I \equiv_e I'$, if for any environment E, I is pluggable to E iff I' is pluggable to E.*

Theorem 3 *For any interfaces I, I', $I \equiv I'$ iff $I \equiv_e I'$.*

Proof: Let $I = (X, Y, \xi)$ and suppose $I \equiv I'$. Then $I' = (X, Y, \xi')$. Consider an environment E such that $I \models E$. Then $E = (X, Y, h_X, h_Y)$. Suppose $s \in \mathcal{R}(I_E)$. Since $I \models E$, the formula $h_X(s) \rightarrow (\mathsf{in}(\xi(s)) \wedge (\xi(s) \rightarrow h_Y(s)))$ is valid. By Lemma 3, $s \in \mathcal{R}(I'_E)$. By Lemma 2, $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. This and $I \equiv I'$ imply $\xi(s) \equiv \xi'(s)$. Therefore, the formula $h_X(s) \rightarrow (\mathsf{in}(\xi'(s)) \wedge (\xi'(s) \rightarrow h_Y(s)))$ is also valid, which means $I' \models E$.

In the opposite direction, we suppose $I \equiv_e I'$ and we need to show that for all $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, the formula $\xi(s) \equiv \xi'(s)$ is valid. Suppose not, that is, suppose that there exists an assignment a such that $a \models \xi(s)$ but $a \not\models \xi'(s)$. Let a_X and a_Y be the restrictions of a to X and Y, respectively. Let ϕ_{a_X} and ϕ_{a_Y} be properties on X and Y, respectively, such that the only assignment satisfying ϕ_{a_X} is a_X and the only assignment satisfying ϕ_{a_Y} is a_Y. Since there are a finite number of input and output variables, we can build such properties using the equality symbol $=$.

We use induction on the length of s.

Basis: $\varepsilon \in \mathcal{R}(I) \cap \mathcal{R}(I')$, therefore we need to show that $\xi(\varepsilon) \equiv \xi'(\varepsilon)$. Suppose not, that is, suppose that there exists an assignment a such that $a \models \xi(\varepsilon)$ but $a \not\models \xi'(\varepsilon)$. Let a_X and a_Y be the restrictions of a to X

and Y, respectively. Let ϕ_{a_X} and ϕ_{a_Y} be properties on X and Y, respectively, such that the only assignment satisfying ϕ_{a_X} is a_X and the only assignment satisfying ϕ_{a_Y} is a_Y. Since there is a finite number of input and output variables, we can build such properties using the equality symbol $=$. We reason by cases:

Case 1: $a_X \not\models \mathsf{in}(\xi'(\varepsilon))$. In this case we define environment $E := (X, Y, h_X, h_Y)$ such that $h_Y(s) := \mathsf{true}$ for all s, and h_X is defined as follows: $h_X(\varepsilon) := \phi_{a_X}$ and $h_X(s) := \mathsf{false}$ for all $s \neq \varepsilon$. In other words, E issues input a_X initially, and then stops. We claim that $I \models E$ but $I' \not\models E$, which contradicts the hypothesis $I \equiv_e I'$.

To show that $I \models E$, we need to show that for all $s \in \mathcal{R}(I_E)$, Formula (1) is valid. Observe that, by definition of E, $\mathcal{R}(I_E) = \{\varepsilon\}$, that is, only the empty state is reachable in the closed loop system of I and E: this is because $h_X(s) \equiv \mathsf{false}$ for all $s \neq \varepsilon$. Therefore, we need to show that Formula (1) is valid for $s = \varepsilon$. This means that every assignment on $X \cup Y$ that satisfies ϕ_{a_X} also satisfies $\mathsf{in}(\xi(\varepsilon)) \wedge (\xi(\varepsilon) \to \mathsf{true})$, i.e., $\mathsf{in}(\xi(\varepsilon))$. Let (a, b) be an assignment on $X \cup Y$ such that $(a, b) \models \phi_{a_X}$, and suppose a is an assignment on X and b an assignment on Y. Then $(a, b) \models \phi_{a_X}$ is equivalent to $a \models \phi_{a_X}$. By definition, there is only one assignment that satisfies ϕ_{a_X}, namely, a_X, therefore, $a = a_X$. a_X satisfies $\mathsf{in}(\xi(\varepsilon))$, since (a_X, a_Y) satisfies $\xi(\varepsilon)$. Since $\mathsf{in}(\xi(\varepsilon))$ does not contain variables in Y, $(a, b) \models \mathsf{in}(\xi(\varepsilon))$. This proves that $I \models E$. By hypothesis of Case 1, $a_X \not\models \mathsf{in}(\xi'(\varepsilon))$. Therefore, $(a, b) = (a_X, b)$ does not satisfy $\mathsf{in}(\xi'(\varepsilon)) \wedge (\xi'(\varepsilon) \to \mathsf{true})$ even though it satisfies ϕ_{a_X}. This proves that $I' \not\models E$. Contradiction. This completes Case 1.

Case 2: $a_X \models \mathsf{in}(\xi'(\varepsilon))$. Let $\psi := \mathsf{out}(\phi_{a_X} \wedge \xi(\varepsilon))$ and $\psi' := \mathsf{out}(\phi_{a_X} \wedge \xi'(\varepsilon))$. Notice that ψ and ψ' are properties on Y. From $(a_X, a_Y) \models \xi(\varepsilon)$, we get $a_Y \models \psi$. Also, from $(a_X, a_Y) \not\models \xi'(\varepsilon)$, we get $a_Y \not\models \psi'$. We define environment $E := (X, Y, h_X, h_Y)$ such that $h_Y(s) := \psi'$ for all s, and h_X is defined as follows: $h_X(\varepsilon) := \phi_{a_X}$ and $h_X(s) := \mathsf{false}$ for all $s \neq \varepsilon$. We claim that $I' \models E$ but $I \not\models E$, which contradicts the hypothesis $I \equiv_e I'$.

To show that $I' \models E$ we need to show that for all $s \in \mathcal{R}(I'_E)$, Formula (1) is valid. As in Case 1, we observe that $\mathcal{R}(I'_E) = \{\varepsilon\}$. Therefore, we need to show that Formula (1) is valid for $s = \varepsilon$. Let (a_X, b) be an assignment on $X \cup Y$ such that $(a_X, b) \models \phi_{a_X}$, where a_X is the single assignment on X that satisfies ϕ_{a_X} and b is some assignment on Y. We must show $(a_X, b) \models \mathsf{in}(\xi'(\varepsilon)) \wedge (\xi'(\varepsilon) \to \psi')$. a_X satisfies $\mathsf{in}(\xi'(\varepsilon))$ by hypothesis of Case 2. Suppose $(a_X, b) \models \xi'(\varepsilon)$. Then $b \models \mathsf{out}(\phi_{a_X} \wedge \xi'(\varepsilon)) \equiv \psi'$, i.e., $(a_X, b) \models \psi'$. This proves $I' \models E$.

We next show that $I \not\models E$ by showing that Formula (1) is invalid for the initial state ε. In particular, we show that $(a_X, a_Y) \not\models \xi(\varepsilon) \to \psi'$, even though $a_X \models \phi_{a_X}$. By definition, $(a_X, a_Y) = a \models \xi(\varepsilon)$. But $a_Y \not\models \psi'$, therefore, $(a_X, a_Y) \not\models \psi'$ (ψ' is a property on Y). This completes Case 2 and the Basis.

Induction step: Consider $s \cdot a \in \mathcal{R}(I) \cap \mathcal{R}(I')$. We must prove that $\xi(s \cdot a) \equiv \xi'(s \cdot a)$ is valid. The proof follows the same lines as in the Basis. We suppose that the result does not hold and reach a contradiction. The difference is how we construct the environments in Cases 1 and 2.

In Case 1, we define E so that $h_X(s') := \xi(s')$ for all states $s' \leq s$, that is, states that are prefixes of s. We define $h_X(s \cdot a) := \phi_{a_X}$ and $h_X(s'') := \mathsf{false}$ for all other s''. As in the Basis, we define $h_Y(s) := \mathsf{true}$ for all s'. From the induction hypothesis we have $\xi(s') \equiv \xi'(s')$ for all $s' \leq s$. Using this, we can show that for all $s \in \mathcal{R}(I_E)$, Formula (1) is valid. Similarly to the Basis, we can show that I' is not pluggable to E because $a_X \not\models \mathsf{in}(\xi'(s \cdot a))$.

In Case 2, again, we define $h_X(s') := \xi(s')$ for all $s' \leq s$. The rest is again similar as in the Basis. ■

5 Composition

We define two types of composition. First, we can compose two interfaces I_1 and I_2 by *connecting* some of the output variables of I_1 to some of the input variables of I_2. One output can be connected to many inputs, but an input can be connected to at most one output. Parallel composition is a special case of composition by connection, where the connection is empty. The connections define a new stateless interface. Thus, the composition process can be repeated to yield arbitrary (acyclic) interface diagrams. Composition by connection is associative (Theorem 5), so the order in which interfaces are composed does not matter.

Two interfaces $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$ are *disjoint* if they have disjoint sets of input and output variables: $(X \cup Y) \cap (X' \cup Y') = \emptyset$.

Definition 10 (Composition by connection) *Let* $I_i = (X_i, Y_i, \xi_i)$, *for* $i = 1, 2$, *be two disjoint interfaces. A* connection θ between I_1, I_2, *is a finite set of pairs of variables,* $\theta = \{(y_i, x_i) \mid i = 1, ..., m\}$, *such that: (1)* $\forall (y, x) \in \theta : y \in Y_1 \wedge x \in X_2$, *and (2)* $\forall (y, x), (y', x') \in \theta : x = x' \to y = y'$. *The* external input and output variables *are the sets of variables* $X_{\theta(I_1,I_2)}$ *and* $Y_{\theta(I_1,I_2)}$, *respectively, defined as follows (where* $X_\theta := \{x \mid \exists (y, x) \in \theta\}$*):*

$$
\begin{aligned}
X_{\theta(I_1,I_2)} &:= (X_1 \cup X_2) \setminus X_\theta \\
Y_{\theta(I_1,I_2)} &:= Y_1 \cup Y_2 \cup X_\theta
\end{aligned}
$$

The connection θ *defines the* composite interface $\theta(I_1, I_2) := (X_{\theta(I_1,I_2)}, Y_{\theta(I_1,I_2)}, \xi)$, *where, for every* $s \in \mathcal{A}(X_{\theta(I_1,I_2)} \cup Y_{\theta(I_1,I_2)})^*$*:*

$$
\begin{aligned}
\xi(s) &:= \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta \wedge \forall Y_{\theta(I_1,I_2)} : \Phi \\
\Phi &:= (\xi_1(s_1) \wedge \rho_\theta) \to \mathsf{in}(\xi_2(s_2)) \\
\rho_\theta &:= \bigwedge_{(y,x)\in\theta} y = x
\end{aligned}
\tag{4}
$$

and, for $i = 1, 2$, s_i *is defined to be the projection of* s *to variables in* $X_i \cup Y_i$.

Definition 10 may seem unnecessarily complex at first sight. In particular, the reader may doubt the necessity of the term $\forall Y_{\theta(I_1,I_2)} : \Phi$ in the definition of $\xi(s)$. Informally speaking, this term states that, no matter which outputs I_1 chooses to produce for a given input, all such outputs are legal inputs for I_2 (when connected). This condition is essential for the validity of our interface theory. Omitting this condition would result in a fundamental property of the theory (Theorem 12) not being true, as will be explained in Example 11.

Notice that, by definition of θ, $X_\theta \subseteq X_2$. This implies $X_1 \subseteq X_{\theta(I_1,I_2)}$, that is, every input variable of I_1 is also an input variable of $\theta(I_1, I_2)$.

A connection θ is allowed to be empty. In that case, $\rho_\theta \equiv \mathsf{true}$, and the composition can be viewed as the *parallel composition* of two interfaces. If θ is empty, we write $I_1 \| I_2$ instead of $\theta(I_1, I_2)$. As may be expected, the contract of the parallel composition at a given global state is the conjunction of the original contracts at the corresponding local states, which implies that parallel composition is commutative:

Lemma 4 *Consider two disjoint interfaces,* $I_i = (X_i, Y_i, \xi_i)$, $i = 1, 2$. *Then* $I_1 \| I_2 = (X_1 \cup X_2, Y_1 \cup Y_2, \xi)$, *where* ξ *is such that for all* $s \in \mathcal{A}(X_1 \cup X_2 \cup Y_1 \cup Y_2)^*$, $\xi(s) \equiv \xi_1(s_1) \wedge \xi_2(s_2)$, *where, for* $i = 1, 2$, s_i *is the projection of* s *to* $X_i \cup Y_i$.

Proof: Following Definition 10, we have:

$$I_1 \| I_2 \quad = \quad (X_1 \cup X_2, Y_1 \cup Y_2, \xi)$$

where for all $s \in \mathcal{A}(X_1 \cup X_2 \cup Y_1 \cup Y_2)^*$

$$\xi(s) \quad = \quad \xi_1(s_1) \wedge \xi_2(s_2) \wedge \big(\forall Y_1 \cup Y_2 : \xi_1(s_1) \to \mathsf{in}(\xi_2(s_2))\big)$$

Observe that $\mathsf{in}(\xi_2(s_2))$ is a formula over X_2, that is, does not depend on $Y_1 \cup Y_2$. Therefore,

$$
\begin{gathered}
\big(\forall Y_1 \cup Y_2 : \xi_1(s_1) \to \mathsf{in}(\xi_2(s_2))\big) \equiv \neg(\exists Y_1 \cup Y_2 : \xi_1(s_1) \wedge \neg\mathsf{in}(\xi_2(s_2))) \equiv \\
\neg(\neg\mathsf{in}(\xi_2(s_2)) \wedge \exists Y_1 \cup Y_2 : \xi_1(s_1)) \equiv \big(\mathsf{in}(\xi_2(s_2)) \vee \neg\exists Y_1 \cup Y_2 : \xi_1(s_1)\big)
\end{gathered}
$$

Now, observe that $\phi \to \mathsf{in}(\phi)$ is a valid formula for any ϕ. Therefore, $\xi_2(s_2) \to \mathsf{in}(\xi_2(s_2)) \to \mathsf{in}(\xi_2(s_2)) \vee \neg\exists Y_1 \cup Y_2 : \xi_1(s_1)$, which gives

$$\Big(\xi_1(s_1) \wedge \xi_2(s_2) \wedge \forall Y_1 \cup Y_2 : \xi_1(s_1) \to \mathsf{in}(\xi_2(s_2))\Big) \equiv (\xi_1(s_1) \wedge \xi_2(s_2))$$

■

Theorem 4 (Commutativity of parallel composition) *For two disjoint interfaces, I_1 and I_2, $I_1 \| I_2 \equiv I_2 \| I_1$.*

Proof: Follows from Lemma 4. ■

Theorem 5 (Associativity of connection) *Let I_1, I_2, I_3 be interfaces. Let θ_{12} be a connection between I_1, I_2, θ_{13} a connection between I_1, I_3, and θ_{23} a connection between I_2, I_3. Then:*

$$(\theta_{12} \cup \theta_{13})\,(I_1, \theta_{23}(I_2, I_3)) \equiv (\theta_{13} \cup \theta_{23})\,(\theta_{12}(I_1, I_2), I_3)$$

Proof: For simplicity of notation, we conduct the proof assuming the interfaces are stateless. The proof is almost identical for general interfaces, except that $\xi(s)$ replaces ϕ, $\xi'(s)$ replaces ϕ', and so on.

Suppose the setting is as illustrated in Figure 1. That is, $I_1 = (X_1, Y_1 \cup Y_{12} \cup Y_{13}, \phi_1)$; $I_2 = (X_2 \cup X_{12}, Y_2 \cup Y_{23}, \phi_2)$; $I_3 = (X_3 \cup X_{13} \cup X_{23}, Y_3, \phi_3)$; and θ_{12} connects X_{11} and Y_{12}; θ_{13} connects X_{13} and Y_{13}; θ_{23} connects X_{23} and Y_{23}.

Our first step is to clearly express what the definitions tell us about $I := (\theta_{12} \cup \theta_{13})\,(I_1, \theta_{23}(I_2, I_3))$ and $I' := (\theta_{13} \cup \theta_{23})\,(\theta_{12}(I_1, I_2), I_3)$.

For simplicity, we will use the notation ρ_θ to refer to $\bigwedge_{(y,x)\in\theta} y = x$. We also refer to the outputs of $\theta_{12}(I_1, I_2)$ as $P = Y_1 \cup Y_{12} \cup Y_{13} \cup X_{12} \cup Y_2 \cup Y_{23}$ and the outputs of $\theta_{23}(I_2, I_3)$ as $Q = Y_2 \cup Y_{23} \cup X_{23} \cup Y_3$ and the overall outputs as $O = Y_1 \cup Y_2 \cup Y_3 \cup Y_{12} \cup Y_{13} \cup Y_{23} \cup X_{12} \cup X_{13} \cup X_{23}$.

The definitions are as follows:

$$\begin{aligned}
\theta_{12}(I_1, I_2) &= (X_1 \cup X_2, P, \phi_1 \wedge \phi_2 \wedge \rho_{\theta_{12}} \wedge \forall P : \phi_1 \wedge \rho_{\theta_{12}} \to \mathsf{in}(\phi_2)) \\
\theta_{23}(I_2, I_3) &= (X_2 \cup X_{12} \cup X_3 \cup X_{13}, Q, \phi_2 \wedge \phi_3 \wedge \rho_{\theta_{23}} \wedge \forall Q : \phi_2 \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3))
\end{aligned}$$

Let ϕ_{12} and ϕ_{23} be the contracts of $\theta_{12}(I_1, I_2)$ and $\theta_{23}(I_2, I_3)$, respectively. Then:

$$\begin{aligned}
I &= (X_1 \cup X_2 \cup X_3, O, \phi_{12} \wedge \phi_3 \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}} \wedge \forall O : \phi_{12} \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3)) \\
I' &= (X_1 \cup X_2 \cup X_3, O, \phi_1 \wedge \phi_{23} \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}} \wedge \forall O : \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}} \to \mathsf{in}(\phi_{23}))
\end{aligned}$$

Let ϕ and ϕ' be the contracts of I and I', respectively. Simplifying, we get:

$$\begin{aligned}
\phi &\equiv \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \rho_\theta \wedge (\forall P : \phi_1 \wedge \rho_{\theta_{12}} \to \mathsf{in}(\phi_2)) \wedge (\forall O : \phi_{12} \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3)) \\
\phi' &\equiv \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \rho_\theta \wedge (\forall Q : \phi_2 \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3)) \wedge (\forall O : \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}} \to \mathsf{in}(\phi_{23}))
\end{aligned}$$

In order to simplify discussion, we will name the subformulae as follows:

$$\begin{aligned}
C &:= \forall P : \phi_1 \wedge \rho_{\theta_{12}} \to \mathsf{in}(\phi_2) \\
D &:= \forall O : \phi_{12} \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3) \\
E &:= \forall Q : \phi_2 \wedge \rho_{\theta_{23}} \to \mathsf{in}(\phi_3) \\
F &:= \forall O : \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}} \to \mathsf{in}(\phi_{23})
\end{aligned}$$

In order to prove equivalence of I and I', we need to prove that the following four formulae are valid:

$$\phi \to E, \quad \phi \to F, \quad \phi' \to C, \text{ and } \phi' \to D$$

Proof of $\phi \to E$: Let (x, q, o) be an arbitrary assignment such that $(x, q, o) \models \phi$, where $x \in X_1 \cup X_2 \cup X_3$, $q \in Q$, and $o \in O \setminus Q$. We want to show that $(x, q, o) \models E$ (i.e. $(x, o) \models E$).

Let q' be an arbitrary assignment over Q such that $(x, q', o) \models \phi_2 \wedge \rho_{\theta_{23}}$. We want to show

$$(x, q', o) \models \phi_1 \wedge \phi_2 \wedge \rho_\theta \wedge (\forall P : \phi_1 \wedge \rho_{\theta_{12}} \rightarrow \mathsf{in}(\phi_2)).$$

Clearly, we have $(x, q', o) \models \phi_2 \wedge \rho_{\theta_{23}}$ by construction of q'. We also have $(x, o) \models \phi_1 \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}} \wedge C$ since no free variables are in Q are and $(x, q, o) \models A$. Thus by D, we have $(x, q', o) \models \mathsf{in}(\phi_3)$. Thus we have $(x, o) \models E$. End of proof of $\phi \rightarrow E$.

Proof of $\phi \rightarrow F$: Suppose we are given an assignment $(x, q, o) \models \phi$ where x is over $X_1 \cup X_2 \cup X_3$, q is over Q, and o is over $O \setminus Q$. We want to show that $(x, q, o) \models F$ (i.e. $x \models F$).

Let (q', o') be an arbitrary assignment over O such that $(x, q', o') \models \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{23}}$. We want to now show that $(x, q', o') \models \mathsf{in}(\phi_{23})$. To do so, we first expand $\mathsf{in}(\phi_{23})$:

$$\mathsf{in}(\phi_{23}) \equiv \exists Q(\phi_2 \wedge \phi_3 \wedge \rho_{\theta_{23}}) \wedge \forall Q(\phi_2 \wedge \rho_{\theta_{23}} \rightarrow \mathsf{in}(\phi_3))$$

Thus we can reduce the proof to two parts:

(a) $(x, o') \models \exists Q(\phi_2 \wedge \phi_3 \wedge \rho_{\theta_{23}})$, and

(b) $(x, o') \models \forall Q(\phi_2 \wedge \rho_{\theta_{23}} \rightarrow \mathsf{in}(\phi_3))$

For part (a), we want to show that for any assignment q_a over Q: $(x, q_a, o') \models \phi_2 \wedge \rho_{\theta_{23}} \Rightarrow (x, q_a, o') \models \mathsf{in}(\phi_3)$. We start with such an assignment q_a. Combining this with the fact that $(x, o') \models \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{23}}$, we get $(x, q_a, o') \models \phi_1 \wedge \phi_2 \wedge \rho_\theta$. Combined with the fact that $x \models C$, we get $(x, q_a, o') \models \phi_1 \wedge \phi_2 \wedge \rho_\theta \wedge C$. This is exactly the premise of D. Since $x \models D$, this gives us $(x, q_a, o') \models \mathsf{in}(\phi_3)$, which is exactly what we wanted to prove.

For part (b), we want to show that there exists an assignment over Q that models $\phi_2 \wedge \phi_3 \wedge \rho_{\theta_{23}}$. For our purposes, we will divide this assignment into q_{Y2} over $Y_2 \cup Y_{23}$, q_{X3} over X_{23}, and q_{Y3} over Y_3. First, since $x \models C$ and $(x, o') \models \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{23}}$ we have that $(x, o') \models \mathsf{in}(\phi_2)$. Expanding the definition of in, this means that $\exists Y_2 \phi_2$. Using this as our assignment of q_{Y2}, we have that $(x, q_{Y2}, o') \models \phi_2$. We can set the values of X_{23} to those of Y_{23} in order to get an assignment of q_{X3} that satisfies $\rho_{\theta_{23}}$. Combining the definition of o' with the assignments to q_{Y2}, q_{X3} with the fact that $x \models C$, gives us:

$$(x, q_{Y2}, q_{X3}, o') \models (\phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{23}}) \wedge (\phi_2 \wedge \rho_{\theta_{23}}) \wedge C$$

Since this is exactly the premise of D, we get $(x, q_{Y2}, q_{X3}, o') \models \mathsf{in}(\phi_3)$. But this means that $\exists Y_3 \phi_3$. Using this as our assignment to q_{Y3}, we get $(x, q_{Y2}, q_{X3}, q_{Y3}, o') \models \phi_3$. Combining the terms that we have satisfied over the course of our assignment, we get $(x, q_{Y2}, q_{X3}, q_{Y3}, o') \models \phi_2 \wedge \phi_3 \wedge \rho_{\theta_{23}}$, which is what we wanted to prove.

Combining our results from part (a) and part (b) we get $(x, o') \models \mathsf{in}(\phi_{23})$. Thus $(x, q, o) \models F$. End of proof of $\phi \rightarrow F$.

Proof of $\phi' \rightarrow C$: Suppose $(x, p, o) \models B$ where $x \in X_1 \cup X_2 \cup X_3$, $p \in P$, and $o \in O \setminus P$. We want to show that $(x, p, o) \models C$ (i.e. $(x, o) \models C$).

Let p' be an assignment over P such that $(x, p', o) \models \phi_1 \wedge \rho_{\theta_{12}}$. Now take o' over $O \setminus P$ such that $(x, p', o') \models \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}}$. This can be done by setting the variables of Y_{13} to those of X_{13}. By F, we have that $(x, p', o') \models \mathsf{in}(\phi_{23})$, so in particular, $(x, p', o') \models \mathsf{in}(\phi_2)$. Since $\mathsf{in}(\phi_2)$ does not contain free variables in $O \setminus P$, this means $(x, p', o) \models \mathsf{in}(\phi_2)$. Thus we have $(x, o) \models C$. End of proof of $\phi' \rightarrow C$.

Proof of $\phi' \rightarrow D$: Suppose $(x, o) \models \phi'$, where x is over $X_1 \cup X_2 \cup X_3$, and o is over O.

Let o' be an arbitrary assignment over O with $(x, o') \models \phi_{12} \wedge \rho_{\theta_{13}} \wedge \rho_{\theta_{23}}$. Clearly $(x, o') \models \phi_1 \wedge \rho_{\theta_{12}} \wedge \rho_{\theta_{13}}$. By F, we have $(x, o') \models \mathsf{in}(\phi_{23})$. But this also means that $(x, o') \models \mathsf{in}(\phi_3)$ Thus we have $(x, o) \models D$. End of proof of $\phi' \rightarrow D$.

■

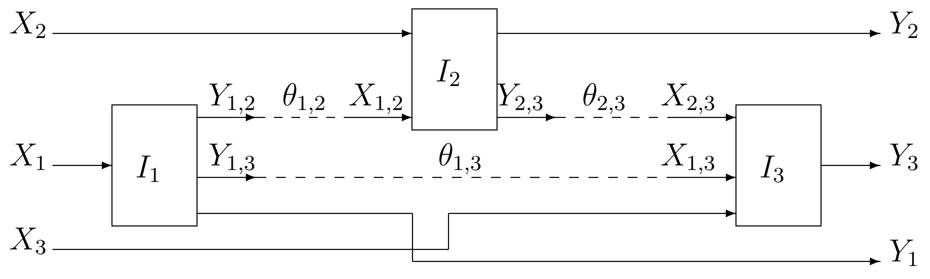

Figure 1: Setting used in the proof of associativity.

Example 4 *Consider the diagram of stateless interfaces shown in Figure 2, where:*

$$\begin{aligned} I_{id} &:= (\{x_1\}, \{y_1\}, y_1 = x_1) \\ I_{+1} &:= (\{x_2\}, \{y_2\}, y_2 = x_2 + 1) \\ I_{eq} &:= (\{z_1, z_2\}, \{\}, z_1 = z_2) \end{aligned}$$

This diagram can be viewed as the equivalent compositions

$$\theta_2\big(I_{+1}, \theta_1(I_{id}, I_{eq})\big) \equiv (\theta_1 \cup \theta_2)\big((I_{id} \| I_{+1}), I_{eq}\big)$$

where $\theta_1 := \{(y_1, z_1)\}$ and $\theta_2 := \{(y_2, z_2)\}$. We proceed to compute the contract of the interface defined by the diagram. It is easier to consider the composition $(\theta_1 \cup \theta_2)((I_{id} \| I_{+1}), I_{eq})$. Define $\theta_3 := \theta_1 \cup \theta_2$. From Lemma 4 we get:

$$I_{id} \| I_{+1} = (\{x_1, x_2\}, \{y_1, y_2\}, y_1 = x_1 \wedge y_2 = x_2 + 1)$$

Then, for $\theta_3((I_{id} \| I_{+1}), I_{eq})$, Formula (4) gives:

$$\Phi := (y_1 = x_1 \wedge y_2 = x_2 + 1 \wedge y_1 = z_1 \wedge y_2 = z_2) \rightarrow z_1 = z_2$$

By quantifier elimination, we get

$$\forall y_1, y_2, z_1, z_2 : \Phi \quad \equiv \quad x_1 = x_2 + 1$$

therefore

$$\begin{aligned} \theta_3((I_{id} \| I_{+1}), I_{eq}) \quad = \quad & (\{x_1, x_2\}, \{y_1, y_2, z_1, z_2\}, \\ & y_1 = x_1 \wedge y_2 = x_2 + 1 \wedge z_1 = z_2 \wedge y_1 = z_1 \wedge y_2 = z_2 \wedge z_1 = z_2 \wedge x_1 = x_2 + 1) \end{aligned}$$

Notice that $\mathsf{in}(\theta_3((I_{id} \| I_{+1}), I_{eq})) \equiv x_1 = x_2 + 1$. That is, because of the connection θ, new assumptions have been generated for the external inputs x_1, x_2.

A composite interface is not guaranteed to be well-formed, even if its components are well-formed, as shown by Example 5 that follows. This is because we do not impose a *compatibility* condition on connections, contrary to [5]. We could easily add well-formedness as a compatibility condition. But we prefer not to do so, because this allows us to state more general results. In particular, Theorem 12 holds independently of whether the connection yields a well-formed interface or not. And together with Theorems 9 and 10, it guarantees that if the refined composite interface is well-formed/formable, then so is the refining one.

Example 5 *Consider the composition $\theta_3((I_{id} \| I_{+1}), I_{eq})$ introduced in Example 4, and let*

$$I_y \quad := \quad (\{\}, \{y\}, \mathsf{true})$$

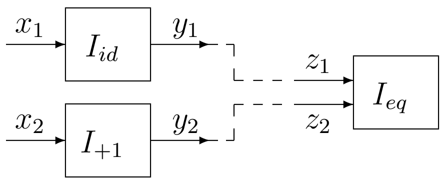

Figure 2: The interface diagram of Example 4.

Let $\theta_4 := \{(y, x_1), (y, x_2)\}$. That is, the output y of I_y is connected to both external inputs x_1 and x_2 of $\theta_3((I_{id}\|I_{+1}), I_{eq})$. The composite interface $I_4 := \theta_4(I_y, \theta_3((I_{id}\|I_{+1}), I_{eq})$ is not well-formed, even though both I_y and $\theta_3((I_{id}\|I_{+1}), I_{eq})$ are well-formed. This is because, for I_4, Formula (4) gives

$$\Phi \quad := \quad (\mathsf{true} \wedge y = x_1 \wedge y = x_2) \rightarrow x_1 = x_2 + 1$$

therefore,

$$\forall x_1, x_2, y_1, y_2, z_1, z_2 : \Phi \quad \equiv \quad y = y + 1$$

Since the above formula is unsatisfiable, I_4 is not well-formed.

We also define a *feedback composition* where an output variable of an interface I is connected to one of its input variables x. For feedback, I is required to be *Moore with respect to x*. The term "Moore interfaces" has been introduced in [2]. Our definition is similar in spirit, but less restrictive than the one in [2]. Both definitions are inspired by Moore machines, where the outputs are determined by the current state alone and do not depend directly on the input. In our version, an interface is Moore with respect to a given input variable x, meaning that the contract may depend on the current state as well as on input variables other than x. This allows to connect an output to x to form a feedback loop without creating causality cycles.

Definition 11 (Moore interfaces) *An interface $I = (X, Y, \xi)$ is* Moore with respect to $x \in X$ *iff for all $s \in \mathcal{R}(I)$, $\xi(s)$ is a property over $(X \cup Y) \setminus \{x\}$. I is* Moore *when it is Moore with respect to every $x \in X$.*

Example 6 *A* unit delay *is a basic building block in many modeling languages (including Simulink and SCADE). Its specification is roughly: "output at time k the value of the input at time $k-1$; at time $k = 0$ (initial time), output some initial value v_0". We can capture this specification as a Moore interface (with respect to its unique input variable) $I_{ud} := (\{x\}, \{y\}, \xi_{ud})$, where ξ_{ud} is defined as follows:*

$$\begin{aligned} \xi_{ud}(\varepsilon) &:= (y = v_0) \\ \xi_{ud}(s \cdot a) &:= (y = a(x)) \end{aligned}$$

That is, initially the contract guarantees $y = v_0$. Then, when the state is some sequence $s \cdot a$, the contract guarantees $y = a(x)$, where $a(x)$ is the last value assigned to input x.

Definition 12 (Composition by feedback) *Let $I = (X, Y, \xi)$ be Moore with respect to some $x \in X$. A* feedback connection *κ* on *I is a pair (y, x) such that $y \in Y$. Define $\rho_\kappa := (x = y)$. The feedback connection κ defines the interface:*

$$\kappa(I) \quad := \quad (X \setminus \{x\}, Y \cup \{x\}, \xi_\kappa) \tag{5}$$

$$\xi_\kappa(s) \quad := \quad \xi(s) \wedge \rho_\kappa, \qquad \text{for all } s \in \mathcal{A}(X \cup Y)^* \tag{6}$$

Theorem 6 *Let $I = (X, Y, \xi)$ be Moore with respect to both $x_1, x_2 \in X$, where $x_1 \neq x_2$. Let $\kappa_1 = (y_1, x_1)$ and $\kappa_2 = (y_2, x_2)$ be feedback connections. Then $\kappa_1(\kappa_2(I)) \equiv \kappa_2(\kappa_1(I))$.*

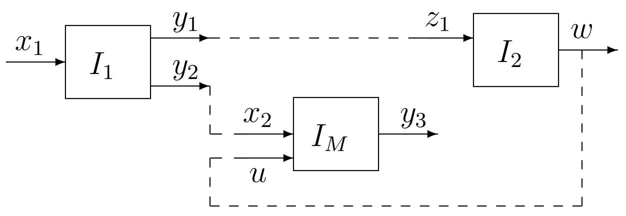

Figure 3: An interface diagram with feedback.

Proof: Following Definition 12, we derive

$$\begin{aligned}\kappa_1(\kappa_2(I)) &= (X \setminus \{x_1, x_2\}, Y \cup \{x_1, x_2\}, \xi_1)\\ \kappa_2(\kappa_1(I)) &= (X \setminus \{x_1, x_2\}, Y \cup \{x_1, x_2\}, \xi_2)\end{aligned}$$

where for all $s \in \mathcal{A}(X \cup Y)^*$

$$\xi_1(s) \equiv \xi(s) \wedge y_1 = x_1 \wedge y_2 = x_2 \equiv \xi_2(s)$$

■

Example 7 *Consider the diagram of interfaces shown in Figure 3. Suppose that I_M is a Moore interface with respect to u. This diagram can be expressed as the composition*

$$\kappa\Big(\theta\big(I_1, (I_2 \| I_M)\big)\Big)$$

where $\theta := \{(y_1, z_1), (y_2, x_2)\}$ and $\kappa := (w, u)$.

Lemma 5 *Let I be a Moore interface with respect to some of its input variables, and let κ be a feedback connection on I. Then $\mathcal{R}(\kappa(I)) \subseteq \mathcal{R}(I)$.*

Proof: Let $I = (X, Y, \xi)$ be Moore w.r.t. $x \in X$. Let $\kappa = (y, x)$. We prove the result by induction on the length of states. The result holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose the result holds for a given state s. We prove it for $s \cdot a$. Let $s \cdot a \in \mathcal{R}(\kappa(I))$. This means that the assignment a satisfies the contract of $\kappa(I)$ at state s, that is, $a \models \xi(s) \wedge x = y$, which implies $a \models \xi(s)$. $s \cdot a \in \mathcal{R}(\kappa(I))$ implies $s \in \mathcal{R}(\kappa(I))$. By the induction hypothesis, $s \in \mathcal{R}(I)$. This and $a \models \xi(s)$ imply $s \cdot a \in \mathcal{R}(I)$. ■

Lemma 6 *Let $I = (X, Y, \xi)$ be a Moore interface with respect to $x \in X$, and let $\kappa = (y, x)$ be a feedback connection on I. Let $\kappa(I) = (X \setminus \{x\}, Y \cup \{y\}, \xi_\kappa)$. Then for any $s \in \mathcal{R}(\kappa(I))$, the formula $\mathsf{in}(\xi_\kappa(s)) \equiv \mathsf{in}(\xi(s))$ is valid.*

Proof: Let $s \in \mathcal{R}(\kappa(I))$. Notice that $\mathsf{in}(\xi_\kappa(s)) \equiv \mathsf{in}(\xi(s))$ is a formula over X. Indeed, $\mathsf{in}(\xi_\kappa(s))$ is a formula over $X \setminus \{x\}$, but $\mathsf{in}(\xi(s))$ is a formula over X.

To show that $\mathsf{in}(\xi_\kappa(s)) \rightarrow \mathsf{in}(\xi(s))$ is valid, we need to show that every assignment over X that satisfies $\mathsf{in}(\xi_\kappa(s))$ also satisfies $\mathsf{in}(\xi(s))$. Consider such an assignment (a, p), where a is an assignment over $X \setminus \{x\}$ and p is an assignment over $\{x\}$. $(a, p) \models \mathsf{in}(\xi_\kappa(s))$ means $(a, p) \models \exists Y \cup \{x\} : \xi(s) \wedge x = y$. Therefore, there exists assignment b over $Y \cup \{x\}$ such that $(a, b) \models \xi(s) \wedge x = y$. Let b' be the restriction of b to Y. We claim that $(a, p, b') \models \xi(s)$. Indeed, since I is Moore w.r.t. x, $\xi(s)$ does not depend on x, therefore, we can assign any value to x, in particular, the value assigned by p. $(a, p, b') \models \xi(s)$ implies $(a, p) \models \exists Y : \xi(s) \equiv \mathsf{in}(\xi(s))$.

To show that $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi_\kappa(s))$ is valid, we need to show that every assignment over X that satisfies $\mathsf{in}(\xi(s))$ also satisfies $\mathsf{in}(\xi_\kappa(s))$. Consider such an assignment (a,p), where a is an assignment over $X \setminus \{x\}$ and p is an assignment over $\{x\}$. $(a,p) \models \mathsf{in}(\xi(s))$ means $(a,p) \models \exists Y : \xi(s)$. Therefore, there exists assignment b over Y such that $(a,p,b) \models \xi(s)$. Let p' be the assignment over $\{x\}$ such that $p'(x) := b(y)$. Since I is Moore w.r.t. x, $\xi(s)$ does not depend on x, therefore, $(a,p',b) \models \xi(s)$. Moreover, $(a,p',b) \models x = y$, therefore $(a,p',b) \models \xi(s) \wedge x = y \equiv \xi_\kappa(s)$. This implies $a \models \exists X \setminus \{x\} : \xi_\kappa(s) \equiv \mathsf{in}(\xi_\kappa(s))$. Therefore $(a,p) \models \mathsf{in}(\xi_\kappa(s))$. ■

Theorem 7 (Feedback preserves well-formedness) *Let I be a Moore interface with respect to some of its input variables, and let κ be a feedback connection on I. If I is well-formed then $\kappa(I)$ is well-formed.*

Proof: Let $I = (X, Y, \xi)$ and $\kappa = (y, x)$. Let $s \in \mathcal{R}(\kappa(I))$. We must show that $\xi(s) \wedge x = y$ is satisfiable. By Lemma 5, $s \in \mathcal{R}(I)$. Since I is well-formed, $\xi(s)$ is satisfiable. Let a be an assignment such that $a \models \xi(s)$. Consider the assignment a' which is identical to a, except that $a'(x) := a(y)$. Since I is Moore w.r.t. x, the satisfaction of $\xi(s)$ does not depend on the value x. Therefore, $a' \models \xi(s)$. Moreover, by definition, $a' \models x = y$, and the proof is complete. ■

6 Hiding

As can been seen in Example 4, composition often creates redundant output variables, in the sense that some of these variables are equal to each other. This happens because input variables that get connected become output variables.

To remove redundant (or other) output variables, we propose a *hiding* operator. For a stateless interface $I = (X, Y, \phi)$, the (stateless) interface resulting from hiding a subset of output variables $Y' \subseteq Y$ can simply be defined as:

$$\mathsf{hide}(Y', I) := (X, Y \setminus Y', \exists Y' : \phi)$$

This definition does not directly extend to the general case of stateful interfaces, however. The reason is that the contract of a stateful interface I may depend on the history of an output y. Then, hiding y is problematic because it is unclear how the contracts of different histories should be combined. To avoid this problem, we allow hiding only those outputs which do not influence the evolution of the contract.

Given $s, s' \in \mathcal{A}(X \cup Y)^*$ such that $|s| = |s'|$ (i.e., s, s' have same length), and given $Z \subseteq X \cup Y$, we say that s and s' *agree on* Z, denoted $s =_Z s'$, when for all $i \in \{1, ..., |s|\}$, and all $z \in Z$, $s_i(z) = s'_i(z)$. Given interface $I = (X, Y, \xi)$, we say that ξ *is independent from* Z if for every $s, s' \in \mathcal{A}(X \cup Y)^*$, $s =_{(X \cup Y) \setminus Z} s'$ implies $\xi(s) = \xi(s')$. That is, the evolution of the variables in Z does not affect the evolution of the contract of I.

Notice that ξ being independent from Z does *not* imply that the contracts of I cannot refer to variables in Z. Indeed, all stateless interfaces trivially satisfy the independence condition: their contracts are invariant in time, i.e., they do not depend on the evolution of variables. Clearly, the contract of a stateless interface can refer to any of its variables.

When ξ is independent from Z, variables in Z can be hidden. In particular, ξ can be viewed as a function from $\mathcal{A}((X \cup Y) \setminus Z)^*$ to $\mathcal{F}(X \cup Y)$ instead of a function from $\mathcal{A}(X \cup Y)^*$ to $\mathcal{F}(X \cup Y)$. We use this when we write $\xi(s)$ for $s \in \mathcal{A}((X \cup Y) \setminus Z)^*$ in the definition the follows:

Definition 13 (Hiding) *Let $I = (X, Y, \xi)$ be an interface and let $Y' \subseteq Y$, such that ξ is independent from Y'. Then,* $\mathsf{hide}(Y', I)$ *is defined to be the interface*

$$\mathsf{hide}(Y', I) := (X, Y \setminus Y', \xi') \qquad (7)$$

such that for any $s \in \mathcal{A}(X \cup Y \setminus Y')^*$, $\xi'(s) := \exists Y' : \xi(s)$.

7 Refinement

Definition 14 (Refinement) *Consider interfaces $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$. We say that I'* refines *I, written $I' \sqsubseteq I$, iff $X = X'$, $Y = Y'$, and for any $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, the following formulae are valid:*

$$\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi'(s)) \tag{8}$$

$$\big(\mathsf{in}(\xi(s)) \wedge \xi'(s)\big) \rightarrow \xi(s) \tag{9}$$

This definition is similar in spirit to other input-contravariant refinement relations, such as *alternating refinement* [1] or refinement of A/G interfaces [4, 5], which, roughly speaking, state that I' refines I iff I' accepts more inputs and produces less outputs than I. In the case of A/G interfaces, where input assumptions are separated from output guarantees, this can be simply stated as $\mathsf{in} \rightarrow \mathsf{in}'$ and $\mathsf{out}' \rightarrow \mathsf{out}$. Our refinement is not strictly output-covariant, however: it requires $\xi'(s) \rightarrow \xi(s)$ only for those inputs that are legal in I.

The reader may wonder why Condition (9) could not be replaced with a simpler condition, namely:

$$\xi'(s) \rightarrow \xi(s) \tag{10}$$

Indeed, for input-complete interfaces, Conditions (8) and (9) are equivalent to Condition (10), (see Theorem 21). In general, however, the two definitions are different in a profound way. Our definition characterizes pluggability in the sense of Theorem 11: I' refines I iff I' can replace I in any context. If we used Condition (10) instead of (9), then this characterization would not hold. We demonstrate this by an example.

Example 8 *Consider interface I_1 from Example 1 and interface $I_{id} := (\{x\}, \{y\}, x = y)$. It can be checked that $I_{id} \sqsubseteq I_1$. If we used Condition (10) instead of Condition (9), however, then I_{id} would not refine I_1: this is because $x = y \not\rightarrow x > 0$. Yet there is no environment E such that $I_1 \models E$ but $I_{id} \not\models E$: this follows from Theorem 11.*

Perhaps surprisingly, among all interfaces with same sets of input and output variables, the interface with contract false is the "top" element with respect to the $\sqsubseteq$ order, that is, it is refined by every other interface. This is in accordance with Theorem 11. The false interface is pluggable only in the trivial environment that stops immediately. Clearly, any other interface can be plugged into this environment as well.

We proceed to state our main results about refinement.

Lemma 7 *Let I, I', I'' be interfaces and suppose $I'' \sqsubseteq I'$ and $I' \sqsubseteq I$. Then $\mathcal{R}(I) \cap \mathcal{R}(I'') \subseteq \mathcal{R}(I')$.*

Proof: Let $I = (X, Y, \xi)$, $I' = (X, Y, \xi')$, $I'' = (X, Y, \xi'')$. By induction on the length of states. Basis: the result holds for the state of length zero, that is, the empty state ε, because ε is reachable in any interface. Induction step: suppose $s \cdot a \in \mathcal{R}(I) \cap \mathcal{R}(I'')$. Then $s \in \mathcal{R}(I) \cap \mathcal{R}(I'')$. From the induction hypothesis, $s \in \mathcal{R}(I')$. $s \cdot a \in \mathcal{R}(I) \cap \mathcal{R}(I'')$ implies $a \models \xi(s) \wedge \xi''(s)$. $a \models \xi(s)$ implies $a \models \mathsf{in}(\xi(s))$. The latter and $I' \sqsubseteq I$ imply $a \models \mathsf{in}(\xi'(s))$. The latter, together with $I'' \sqsubseteq I'$ and $a \models \xi''(s)$, imply $a \models \xi'(s)$. This and $s \in \mathcal{R}(I')$ imply $s \cdot a \in \mathcal{R}(I')$. ∎

Theorem 8 *$\sqsubseteq$ is a reflexive and transitive relation on interfaces.*

Proof: $\sqsubseteq$ is reflexive because $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi(s))$ and $\mathsf{in}(\xi(s)) \wedge \xi(s) \rightarrow \xi(s)$, for any $s \in \mathcal{R}(I)$. To show that $\sqsubseteq$ is transitive, let $I = (X, Y, \xi)$, $I' = (X, Y, \xi')$, $I'' = (X, Y, \xi'')$, and suppose $I'' \sqsubseteq I'$ and $I' \sqsubseteq I$. We must prove $I'' \sqsubseteq I$. Suppose $s \in \mathcal{R}(I) \cap \mathcal{R}(I'')$. By Lemma 7, $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$ and $s \in \mathcal{R}(I') \cap \mathcal{R}(I'')$. These facts together with $I'' \sqsubseteq I'$ and $I' \sqsubseteq I$ imply $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi'(s))$, $\mathsf{in}(\xi(s)) \wedge \xi'(s) \rightarrow \xi(s)$, $\mathsf{in}(\xi'(s)) \rightarrow \mathsf{in}(\xi''(s))$, and $\mathsf{in}(\xi'(s)) \wedge \xi''(s) \rightarrow \xi'(s)$. These imply $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi''(s))$ and $\mathsf{in}(\xi(s)) \wedge \xi''(s) \rightarrow \xi(s)$. ∎

Theorem 9 (Refinement preserves well-formedness for stateless interfaces) *Let I, I' be stateless interfaces such that $I' \sqsubseteq I$. If I is well-formed then I' is well-formed.*

Proof: Let $I' = (X, Y, \phi')$ and $I = (X, Y, \phi)$. I is well-formed, thus ϕ is satisfiable. Let a be an assignment satisfying ϕ and let a_X and a_Y be the restrictions of a to X and Y, respectively. By definition of $\mathsf{in}(\phi)$, $a_X \models \mathsf{in}(\phi)$. By Condition (8), $a_X \models \mathsf{in}(\phi') \equiv \exists Y : \phi'$. Therefore, there exists a'_Y such that $(a_X, a'_Y) \models \phi'$. Thus, ϕ' is satisfiable. Thus, I' is well-formed. ■

Theorem 9 does not generally hold for stateful interfaces: the reason is that, because I' may accept more inputs than I, there may be states that are reachable in I' but not in I, and the contract of I' in these states may be unsatisfiable. When this situation does not occur, refinement preserves well-formedness also in the stateful case. Moreover, refinement always preserves well-formability:

Theorem 10 (Refinement preserves well-formability) *Let I, I' be interfaces such that $I' \sqsubseteq I$. If I is well-formed and $\mathcal{R}(I') \subseteq \mathcal{R}(I)$ then I' is well-formed. Moreover, if I is well-formable then I' is well-formable.*

Proof: Let $I = (X, Y, \xi)$. Since $I' \sqsubseteq I$, $I' = (X, Y, \xi')$, for some ξ'.

Suppose I is well-formed and $\mathcal{R}(I') \subseteq \mathcal{R}(I)$. We need to show that for any $s \in \mathcal{R}(I')$, $\xi'(s)$ is satisfiable. By hypothesis, $s \in \mathcal{R}(I)$ and I is well-formed, therefore, $\xi(s)$ is satisfiable. Reasoning as in the proof of Theorem 9, we can show that $\xi'(s)$ is also satisfiable.

Suppose I is well-formable. Then there exists $I_1 = (X, Y, \xi_1)$ such that I_1 is well-formed, and for all $s \in \mathcal{R}(I_1)$, $\xi_1(s) \equiv \xi(s) \wedge \phi_s$, for some property ϕ_s over X. Since ξ_1 strengthens ξ, $\mathcal{R}(I_1) \subseteq \mathcal{R}(I)$. Since $\xi(s) \wedge \phi_s \equiv \xi(s) \wedge \mathsf{in}(\xi(s)) \wedge \phi_s$, we can assume without loss of generality that $\phi_s \to \mathsf{in}(\xi(s))$. We define $I_2 := (X, Y, \xi_2)$ such that $\xi_2(s) := \xi'(s) \wedge \phi_s$, if $s \in \mathcal{R}(I_1)$, and $\xi_2(s) := \xi'(s)$, if $s \notin \mathcal{R}(I_1)$.

Claim 1: $\mathcal{R}(I_2) \subseteq \mathcal{R}(I_1)$. By induction on the length of a state s. The result holds for $s = \varepsilon$. Suppose $s \cdot a \in \mathcal{R}(I_2)$. Then $s \in \mathcal{R}(I_2)$ and from the induction hypothesis, $s \in \mathcal{R}(I_1)$. Also, $a \models \xi_2(s) \equiv \xi'(s) \wedge \phi_s$ (because $s \in \mathcal{R}(I_1)$). Since $\phi_s \to \mathsf{in}(\xi(s))$, $a \models \mathsf{in}(\xi(s)) \wedge \xi'(s)$. This and $I' \sqsubseteq I$ imply $a \models \xi(s)$, thus, $a \models \xi(s) \wedge \phi_s \equiv \xi_1(s)$. Thus, $s \cdot a \in \mathcal{R}(I_1)$.

Claim 2: $\mathcal{R}(I_2) \subseteq \mathcal{R}(I')$. Trivial because ξ_2 is a strengthening of ξ'.

Claim 3: $I_2 \sqsubseteq I_1$. Suppose $s \in \mathcal{R}(I_1) \cap \mathcal{R}(I_2)$. By Claim 2 and the fact $\mathcal{R}(I_1) \subseteq \mathcal{R}(I)$, we have $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. Then: $\mathsf{in}(\xi_1(s)) \equiv \mathsf{in}(\xi(s)) \wedge \phi_s$. Since $I' \sqsubseteq I$ and $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi'(s))$. Therefore $\mathsf{in}(\xi(s)) \wedge \phi_s \to \mathsf{in}(\xi'(s)) \wedge \phi_s$. The latter formula is equivalent to $\mathsf{in}(\xi_2(s))$ because $s \in \mathcal{R}(I_1)$. Also, $\mathsf{in}(\xi_1(s)) \wedge \xi_2(s) \equiv \mathsf{in}(\xi(s)) \wedge \xi'(s) \wedge \phi_s \to \xi(s) \wedge \phi_s \equiv \xi_1(s)$. This completes Claim 3.

Claim 4: for all $s \in \mathcal{R}(I_2)$, $\xi_2(s) \equiv \xi'(s) \wedge \phi_s$. Follows by definition of ξ_2 and Claim 3.

Claim 1 and Claim 3, together with the fact that I_1 is well-formed, and by the first part of this theorem, imply that I_2 is well-formed. This, together with Claims 2 and 4 imply that I_2 is a well-formed witness for I', thus, I' is well-formable. ■

Lemma 8 *Consider properties ϕ, ϕ' over $X \cup Y$ such that*

$$\Big(\mathsf{in}(\phi) \to \mathsf{in}(\phi')\Big) \wedge \Big(\big(\mathsf{in}(\phi) \wedge \phi'\big) \to \phi\Big)$$

is valid. Then for any property ψ over Y, the following formula is also valid:

$$\big(\mathsf{in}(\phi) \wedge (\phi \to \psi)\big) \to \big(\mathsf{in}(\phi') \wedge (\phi' \to \psi)\big)$$

Proof: We already have $\mathsf{in}(\phi) \to \mathsf{in}(\phi')$, so it remains to prove $\big(\mathsf{in}(\phi) \wedge (\phi \to \psi)\big) \to (\phi' \to \psi)$. Consider an assignment (a, b) on $X \cup Y$ such that $(a, b) \models \mathsf{in}(\phi) \wedge (\phi \to \psi)$. This means: (1) $a \models \mathsf{in}(\phi)$; and (2) if $(a, b) \models \phi$ then $b \models \psi$. We must show that if $(a, b) \models \phi'$ then $b \models \psi$. Suppose $(a, b) \models \phi'$. This, together with (1) and our hypothesis, gives $(a, b) \models \phi$, which together with (2) gives $b \models \psi$. ■

Lemma 9 *Let I, I' be interfaces and E be a environment. If I is pluggable to E and $I' \sqsubseteq I$ then $\mathcal{R}(I'_E) \subseteq \mathcal{R}(I_E)$.*

Proof: By induction on the length of states. It holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose the result holds for a given state s. We prove it for $s \cdot a$. Let $s \cdot a \in \mathcal{R}(I'_E)$. This means that the assignment a satisfies the contract of I'_E at state s, that is, $a \models \xi'(s) \wedge h_X(s)$. Also, $s \in \mathcal{R}(I'_E)$, therefore, by the induction hypothesis, $s \in \mathcal{R}(I_E)$. This and the hypothesis that I is pluggable to E imply that $h_X(s) \rightarrow \mathsf{in}(\xi(s))$ is valid. Therefore, $a \models \xi'(s) \wedge \mathsf{in}(\xi(s))$. Now, by Lemma 2 and the facts $s \in \mathcal{R}(I'_E)$ and $s \in \mathcal{R}(I'_E)$, we derive $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. This and the hypothesis $I' \sqsubseteq I$ imply that $(\mathsf{in}(\xi(s)) \wedge \xi'(s)) \rightarrow \xi(s)$ is valid, therefore, $a \models \xi(s) \wedge \mathsf{in}(\xi(s))$. Thus, $s \cdot a \in \mathcal{R}(I_E)$, and the proof of Lemma 9 is complete. ■

Theorem 11 (Refinement characterizes pluggability) *$I \sqsubseteq I'$ iff for all environments E, $I \models E$ implies $I' \models E$.*

Proof: Suppose $I \sqsubseteq I'$ and let E be an environment such that $I \models E$. We prove that $I' \models E$. Let $I = (X, Y, \xi)$. Since $I' \sqsubseteq I$, $I' = (X, Y, \xi')$, for some ξ'. Since I is pluggable to E, $E = (X, Y, h_X, h_Y)$, for some h_X, h_Y. Let I_E and I'_E be the interfaces defined by (2) and (3) for I and I', respectively. Consider some $s \in \mathcal{R}(I'_E)$. To show $I' \models E$, we need to show that the following formula is valid:

$$h_X(s) \rightarrow \big(\mathsf{in}(\xi'(s)) \wedge (\xi'(s) \rightarrow h_Y(s))\big)$$

By Lemma 9, $s \in \mathcal{R}(I_E)$. This and the hypothesis $I \models E$ imply that the following formula is valid:

$$h_X(s) \rightarrow \big(\mathsf{in}(\xi(s)) \wedge (\xi(s) \rightarrow h_Y(s))\big)$$

Lemma 2 and $s \in \mathcal{R}(I_E) \cap \mathcal{R}(I'_E)$ imply $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. This and $I' \sqsubseteq I$ imply that Formulae (8) and (9) are valid. This and Lemma 8, imply that the following formula is valid:

$$\big(\mathsf{in}(\xi(s)) \wedge (\xi(s) \rightarrow h_Y(s))\big) \rightarrow \big(\mathsf{in}(\xi'(s)) \wedge (\xi'(s) \rightarrow h_Y(s))\big)$$

This proves the first part of the theorem.

For the converse, suppose $I' \not\sqsubseteq I$. We build an environment E such that $I \models E$ but $I' \not\models E$. The construction is trivial when I and I' have different sets of input or output variables, so we will assume that $I = (X, Y, \xi)$ and $I' = (X, Y, \xi')$. Since $I' \not\sqsubseteq I$, there exists $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$ such that: either $\mathsf{in}(\xi(s)) \wedge \neg\mathsf{in}(\xi'(s))$ is satisfiable (Case 1); or $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi'(s))$ is valid, but $\mathsf{in}(\xi(s)) \wedge \xi'(s) \wedge \neg\xi(s)$ is satisfiable (Case 2). Suppose $s = (a_1, b_1)(a_2, b_2) \cdots (a_n, b_n)$, for some integer $n \geq 0$, where a_i is an assignment over X and b_i is an assignment on Y, for $i = 1, ..., n$.

Case 1: Let a_{n+1} be an assignment over X such that $a_{n+1} \models \mathsf{in}(\xi(s))$ but $a_{n+1} \not\models \mathsf{in}(\xi'(s))$. Similarly to the proof of Theorem 3, we define formula ϕ_X^i on X such that the only assignment satisfying ϕ_X^i is a_i, for $i = 1, ..., n, n+1$. We then define environment $E := (X, Y, h_X, h_Y)$ such that $h_Y(r) := \mathsf{true}$ for all states r, and h_X is defined as follows:

$$h_X(s') := \begin{cases} \phi_X^{i+1}, & \text{if } s' = (a_1, b_1) \cdots (a_i, b_i), 0 \leq i \leq n \\ \mathsf{false}, & \text{otherwise} \end{cases}$$

In other words, E issues inputs $a_1, a_2, ..., a_{n+1}$ in the first $n+1$ steps, provided the outputs match those of state s. As soon as an output does not match or step $n+1$ is reached, E stops (i.e., issues false).

Claim: $I \models E$ but $I' \not\models E$. By construction, $s \in \mathcal{R}(I_E)$ and $s \in \mathcal{R}(I'_E)$. Thus, from the definition of pluggability, we know that I and I' are pluggable if the following two formulae are valid, respectively:

$$\phi_X^{n+1} \rightarrow \big(\mathsf{in}(\xi(s)) \wedge (\xi(s) \rightarrow \mathsf{true})\big)$$
$$\phi_X^{n+1} \rightarrow \big(\mathsf{in}(\xi'(s)) \wedge (\xi'(s) \rightarrow \mathsf{true})\big)$$

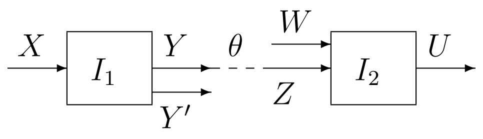

Figure 4: Composing two interfaces.

Since $a_{n+1} \models \mathsf{in}(\xi(s))$ but $a_{n+1} \not\models \mathsf{in}(\xi'(s))$, this implies that $I \models E$ but $I' \not\models E$. This completes Case 1.

Case 2: Note that $\mathsf{in}(\xi(s)) \wedge (\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi'(s)))$ is equivalent to $\mathsf{in}(\xi(s)) \wedge \mathsf{in}(\xi'(s))$. Let (a_{n+1}, b_{n+1}) be an assignment over $X \cup Y$, with a_{n+1} an assignment on X and b_{n+1} an assignment on Y, such that $a_{n+1} \models \mathsf{in}(\xi(s)) \wedge \mathsf{in}(\xi'(s))$ and $(a_{n+1}, b_{n+1}) \models \xi'(s)$, but $(a_{n+1}, b_{n+1}) \not\models \xi(s)$. We define E almost identically to Case 1, except that $h_Y(s) := \mathsf{out}(\xi(s) \wedge \phi_X^{n+1})$. Note that, from $(a_{n+1}, b_{n+1}) \not\models \xi(s)$ $a_{n+1} \models \phi_X^{n+1}$, we get $b_{n+1} \not\models h_Y(s)$.

Claim: $I \models E$ but $I' \not\models E$. By construction, $s \in \mathcal{R}(I_E)$ and $s \in \mathcal{R}(I'_E)$. Thus, from the definition of pluggability, we know that I and I' are pluggable if the following two formulae are valid, respectively:

$$\phi_X^{n+1} \to \big(\mathsf{in}(\xi(s)) \wedge (\xi(s) \to h_Y(s))\big)$$
$$\phi_X^{n+1} \to \big(\mathsf{in}(\xi'(s)) \wedge (\xi'(s) \to h_Y(s))\big)$$

The crux of this case hinges on whether $\xi(s) \to h_Y(s)$ and $\xi'(s) \to h_Y(s)$ are satisfied under (a_{n+1}, b_{n+1}). Since $(a_{n+1}, b_{n+1}) \models \xi'(s)$ but $(a_{n+1}, b_{n+1}) \not\models \xi(s)$ the first formula is satisfied, but the second is not. Thus $I \models E$ but $I' \not\models E$. This completes Case 2 and the proof of the theorem. ■

Lemma 10 *Consider two disjoint interfaces I_1 and I_2, and a connection θ between I_1, I_2. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the projections of $\mathcal{R}(\theta(I_1, I_2))$ to states over the variables of I_1 and I_2, respectively. Then $\mathcal{R}_1 \subseteq \mathcal{R}(I_1)$ and $\mathcal{R}_2 \subseteq \mathcal{R}(I_2)$.*

Proof: Let $I_1 = (X_1, Y_1, \xi_1)$ and $I_2 = (X_2, Y_2, \xi_2)$. Let ξ be the contract function of $\theta(I_1, I_2)$. We prove the result by induction on the length of states. It clearly holds for the empty state ε.

Let $s_1 \cdot a_1 \in \mathcal{R}_1$. This means that there exists state $s \cdot a \in \mathcal{R}(\theta(I_1, I_2))$ such that $s_1 \cdot a_1$ is the projection of $s \cdot a$ to the variables of I_1. From $s \cdot a \in \mathcal{R}(\theta(I_1, I_2))$, we get

$$a \models \xi(s) \text{ i.e. } a \models \xi_1(s_1) \wedge \xi_2(s_2) \wedge \cdots$$

Therefore, $a \models \xi_1(s_1)$, which means $a_1 \models \xi_1(s_1)$. By the induction hypothesis, $s_1 \in \mathcal{R}(I_1)$. These two facts imply $s_1 \cdot a \in \mathcal{R}(I_1)$. This proves $\mathcal{R}_1 \subseteq \mathcal{R}(I_1)$. The proof of $\mathcal{R}_2 \subseteq \mathcal{R}(I_2)$ is similar. ■

Theorem 12 (Connection preserves refinement) *Consider two disjoint interfaces I_1 and I_2, and a connection θ between I_1, I_2. Let I'_1, I'_2 be interfaces such that $I'_1 \sqsubseteq I_1$ and $I'_2 \sqsubseteq I_2$. Then, $\theta(I'_1, I'_2) \sqsubseteq \theta(I_1, I_2)$.*

Proof: Let $I_1 = (X, Y \cup Y', \xi_1)$ and $I_2 = (Z \cup W, U, \xi_2)$. Let θ be a connection between I_1, I_2, such that $Y = \{y \mid (y, z) \in \theta \text{ for some } z\}$ and $Z = \{z \mid (y, z) \in \theta \text{ for some } y\}$. In other words, Y represents the set of output variables of I_1 that are connected to input variables of I_2. Y' is the set of the rest of the output variables of I_1. Z represents those input variables of I_2 that are connected (to outputs of I_1) and W those that are not connected. Any of the sets X, Y, Y', Z, W, U may be empty. Since $I'_i \sqsubseteq I_i$, for $i = 1, 2$, we have $I'_1 = (X, Y \cup Y', \xi'_1)$ and $I'_2 = (Z \cup W, U, \xi'_2)$. The composition setting is illustrated in Figure 4.

Given the above, and Definition 10, we have, for $s \in \mathcal{A}(X \cup W \cup Y \cup Y' \cup Z \cup U)^*$, s_1 the projection of s to $X \cup Y \cup Y'$, and s_2 the projection of s to $W \cup Z \cup U$:

$$\begin{aligned}
\theta(I_1, I_2) &:= (X \cup W, Y \cup Y' \cup Z \cup U, \xi) && (11)\\
\xi(s) &:= \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta \wedge \Psi && (12)\\
\Psi &:= \forall Y \cup Y' \cup Z \cup U : (\xi_1(s_1) \wedge \rho_\theta) \to \mathsf{in}(\xi_2(s_2)) && (13)\\
\theta(I'_1, I'_2) &:= (X \cup W, Y \cup Y' \cup Z \cup U, \xi') && (14)\\
\xi'(s) &:= \xi'_1(s_1) \wedge \xi'_2(s_2) \wedge \rho_\theta \wedge \Psi' && (15)\\
\Psi' &:= \forall Y \cup Y' \cup Z \cup U : (\xi'_1(s_1) \wedge \rho_\theta) \to \mathsf{in}(\xi'_2(s_2)) && (16)
\end{aligned}$$

Let $s \in \mathcal{R}(\theta(I_1, I_2)) \cap \mathcal{R}(\theta(I'_1, I'_2))$. To prove $\theta(I'_1, I'_2) \sqsubseteq \theta(I_1, I_2)$ we need to prove that: (A) $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi'(s))$ is valid; and (B) $(\mathsf{in}(\xi(s)) \wedge \xi'(s)) \to \xi(s)$ is valid. Note that, by Lemma 10, $s_1 \in \mathcal{R}(I_1) \cap \mathcal{R}(I'_1)$ and $s_2 \in \mathcal{R}(I_2) \cap \mathcal{R}(I'_2)$. We use these two facts without mention in the rest of the proof. We proceed in proving claims (A) and (B).

(A): $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi'(s))$ is valid: Suppose the result does not hold. This means that $\mathsf{in}(\xi(s)) \wedge \neg\mathsf{in}(\xi'(s))$ is satisfiable, i.e.,

$$\psi_1 := (\exists Y \cup Y' \cup Z \cup U : \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta \wedge \Psi) \wedge (\forall Y \cup Y' \cup Z \cup U : \neg\xi'_1(s_1) \vee \neg\xi'_2(s_2) \vee \neg\rho_\theta \vee \neg\Psi')$$

is satisfiable. Note that ψ_1, Ψ and Ψ' are all formulae over $X \cup W$, therefore, ψ_1 is equivalent to:

$$\psi_2 := \Psi \wedge (\exists Y \cup Y' \cup Z \cup U : \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta) \wedge \Big(\neg\Psi' \vee (\forall Y \cup Y' \cup Z \cup U : \neg\xi'_1(s_1) \vee \neg\xi'_2(s_2) \vee \neg\rho_\theta)\Big)$$

Let a be an assignment over $X \cup W$ satisfying ψ_2. We claim that $a \models \neg\Psi'$. Suppose not, i.e., $a \models \Psi'$. Then, from $a \models \psi_2$, we derive $a \models \forall Y \cup Y' \cup Z \cup U : \neg\xi'_1(s_1) \vee \neg\xi'_2(s_2) \vee \neg\rho_\theta$. Also, $a \models \mathsf{in}(\xi_1(s_1))$. Since $I'_1 \sqsubseteq I_1$, $a \models \mathsf{in}(\xi'_1(s_1))$. This means that there exists an assignment c over $Y \cup Y'$ such that $(a, c) \models \xi'_1(s_1)$. Let d be an assignment over Z such that $(c, d) \models \rho_\theta$: that is, we set an input variable z of I_2 to the value $c(y)$ of the output variable y of I_1 that z is connected to. Combining, we have $(a, c, d) \models \xi'_1(s_1) \wedge \rho_\theta$. This and $a \models \Psi'$ imply that $(a, c, d) \models \mathsf{in}(\xi'_2(s_2))$. Therefore, there exists an assignment e over U such that $(a, c, d, e) \models \xi'_2(s_2)$. Combining, we have $(a, c, d, e) \models \xi'_1(s_1) \wedge \xi'_2(s_2) \wedge \rho_\theta$, which contradicts $a \models \forall Y \cup Y' \cup Z \cup U : \neg\xi'_1(s_1) \vee \neg\xi'_2(s_2) \vee \neg\rho_\theta$. Thus, the claim $a \models \neg\Psi'$ is proven and we have that a satisfies:

$$\psi_3 := \Psi \wedge \neg\Psi' \wedge (\exists Y \cup Y' \cup Z \cup U : \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta)$$

Since a does not satisfy Ψ', there exists an assignment b over $Y \cup Y' \cup Z \cup U$, such that $(a, b) \models \xi'_1(s_1) \wedge \rho_\theta \wedge \neg\mathsf{in}(\xi'_2(s_2))$. Since $I'_2 \sqsubseteq I_2$, $\mathsf{in}(\xi_2(s_2)) \to \mathsf{in}(\xi'_2(s_2))$, or $\neg\mathsf{in}(\xi'_2(s_2)) \to \neg\mathsf{in}(\xi_2(s_2))$. Therefore, $(a, b) \models \neg\mathsf{in}(\xi_2(s_2))$. Now, from $a \models \psi_3$, we get $(a, b) \models \mathsf{in}(\xi_1(s_1))$. From $I'_1 \sqsubseteq I_1$ we have $\mathsf{in}(\xi_1(s_1)) \wedge \xi'_1(s_1) \to \xi_1(s_1)$. Therefore, $(a, b) \models \xi_1(s_1)$. This, together with $a \models \Psi$ and $(a, b) \models \rho_\theta$, imply $(a, b) \models \mathsf{in}(\xi_2(s_2))$. Contradiction. This completes the proof of Part (A).

(B): $(\mathsf{in}(\xi(s)) \wedge \xi'(s)) \to \xi(s)$ is valid: Suppose the result does not hold. This means that $\mathsf{in}(\xi(s)) \wedge \xi'(s) \wedge \neg\xi(s)$ is satisfiable, i.e.,

$$\psi_4 := (\exists Y \cup Y' \cup Z \cup U : \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta \wedge \Psi) \wedge (\xi'_1(s_1) \wedge \xi'_2(s_2) \wedge \rho_\theta \wedge \Psi') \wedge (\neg\xi_1(s_1) \vee \neg\xi_2(s_2) \vee \neg\rho_\theta \vee \neg\Psi)$$

is satisfiable. Because Ψ and Ψ' are formulae over $X \cup W$, ψ_4 simplifies to:

$$\psi_5 := \Psi \wedge \Psi' \wedge (\exists Y \cup Y' \cup Z \cup U : \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta) \wedge (\xi'_1(s_1) \wedge \xi'_2(s_2) \wedge \rho_\theta) \wedge (\neg\xi_1(s_1) \vee \neg\xi_2(s_2))$$

Let a be an assignment over $X \cup W$ such that $a \models \psi_5$. Then $a \models \mathsf{in}(\xi_1(s_1)) \wedge \mathsf{in}(\xi_2(s_2)) \wedge \xi'_1(s_1) \wedge \xi'_2(s_2)$. From the hypotheses $I'_1 \sqsubseteq I_1$ and $I'_2 \sqsubseteq I_2$, we get $\mathsf{in}(\xi_1(s_1)) \wedge \xi'_1(s_1) \to \xi_1(s_1)$ and $\mathsf{in}(\xi_2(s_1)) \wedge \xi'_2(s_2) \to \xi_2(s_2)$. Therefore $a \models \xi_1(s_1) \wedge \xi_2(s_2)$, which contradicts $a \models \psi_5$. This completes the proof of Part (B) and of the theorem. ■

Theorem 13 (Feedback preserves refinement) *Let I, I' be interfaces such that $I' \sqsubseteq I$. Suppose both I and I' are Moore interfaces with respect to one of their input variables, x. Let $\kappa = (y, x)$ be a feedback connection. Then $\kappa(I') \sqsubseteq \kappa(I)$.*

Proof: Let $I = (X, Y, \xi)$. Because $I' \sqsubseteq I$, $I' = (X, Y, \xi')$. Then: $\kappa(I) = (X \setminus \{x\}, Y \cup \{x\}, \xi_\kappa)$ and $\kappa(I') = (X \setminus \{x\}, Y \cup \{x\}, \xi'_\kappa)$, where $\xi_\kappa(s) := \xi(s) \wedge x = y$ and $\xi'_\kappa(s) := \xi'(s) \wedge x = y$, for all $s \in \mathcal{A}(X \cup Y)^*$. To show that $\kappa(I') \sqsubseteq \kappa(I)$, we need to prove that for any $s \in \mathcal{R}(\kappa(I)) \cap \mathcal{R}(\kappa(I'))$, the following formulae are valid:

$$\mathsf{in}(\xi_\kappa(s)) \to \mathsf{in}(\xi'_\kappa(s))$$
$$\big(\mathsf{in}(\xi_\kappa(s)) \wedge \xi'_\kappa(s)\big) \to \xi_\kappa(s)$$

By Lemma 5, $s \in \mathcal{R}(\kappa(I)) \cap \mathcal{R}(\kappa(I'))$ implies $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$. Then:

By Lemma 6 and the fact $I' \sqsubseteq I$:

$$\mathsf{in}(\xi_\kappa(s)) \equiv \mathsf{in}(\xi(s)) \to \mathsf{in}(\xi'(s)) \equiv \mathsf{in}(\xi'_\kappa(s))$$

By Lemma 6, $\mathsf{in}(\xi_\kappa(s)) \to \mathsf{in}(\xi(s))$. By the fact $I' \sqsubseteq I$, $(\mathsf{in}(\xi(s)) \wedge \xi'(s)) \to \xi(s)$. Therefore,

$$\big(\mathsf{in}(\xi_\kappa(s)) \wedge \xi'_\kappa(s)\big) \to \big(\mathsf{in}(\xi(s)) \wedge \xi'(s) \wedge x = y\big) \to (\xi(s) \wedge x = y) \equiv \xi_\kappa(s)$$

■

Note that the assumption that I' be Moore w.r.t. x in Theorem 13 is essential. Indeed, Mooreness is not generally preserved by refinement, as Example 9 shows.

Example 9 *Consider the stateless interfaces $I_{even} := (\{x\}, \{y\}, y \mod 2 = 0)$, where $\mod$ denotes the modulo operator, and $I_{\times 2} := (\{x\}, \{y\}, y = 2x)$. I_{even} is Moore. $I_{\times 2}$ is not Moore. Yet $I_{\times 2} \sqsubseteq I_{even}$.*

Thanks to Theorems 9 and 10, a corollary of Theorem 12 is that composition by connection preserves well-formability for general interfaces, and well-formedness for stateless interfaces. Similarly, a corollary of Theorem 13 is that feedback composition preserves well-formability for general Moore interfaces, and well-formedness for stateless Moore interfaces.

8 Shared refinement

Shared refinement is introduced in [5] as a mechanism to combine two interfaces I and I' into a single interface $I \sqcap I'$ that refines both I and I': $I \sqcap I'$ is able to accept inputs that are legal in either I or I', and provide outputs that are legal in both I and I'. Because of this, $I \sqcap I'$ can replace both I and I', which, as argued in [5], is important for component reuse.

A shared refinement operator for extended (i.e., relational) interfaces is proposed in the Discussion section of [5], and it is conjectured that this operator represents the greatest lower bound with respect to refinement. We show that this holds only if a *shared refinability* condition is imposed. This condition states that for every inputs that is legal in both I and I', the corresponding sets of outputs of I and I' must have a non-empty intersection. Otherwise, it is impossible to provide an output that is legal in both I and I'.

Definition 15 (Shared refinement) *Two interfaces $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$ are* shared-refinable *if $X = X'$, $Y = Y'$ and the following formula is true for all $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$:*

$$\forall X : \big(\mathsf{in}(\xi(s)) \wedge \mathsf{in}(\xi'(s))\big) \to \exists Y : (\xi(s) \wedge \xi'(s)) \qquad (17)$$

In that case, the shared refinement *of I and I', denoted $I \sqcap I'$, is the interface:*

$$\begin{array}{rcl} I \sqcap I' & := & (X, Y, \xi_\sqcap) \\ \xi_\sqcap(s) & := & \begin{cases} \big(\mathsf{in}(\xi(s)) \vee \mathsf{in}(\xi'(s))\big) \wedge \big(\mathsf{in}(\xi(s)) \to \xi(s)\big) \wedge \big(\mathsf{in}(\xi'(s)) \to \xi'(s)\big), & \textit{if } s \in \mathcal{R}(I) \cap \mathcal{R}(I') \\ \xi(s), & \textit{if } s \in \mathcal{R}(I) \setminus \mathcal{R}(I') \\ \xi'(s), & \textit{if } s \in \mathcal{R}(I') \setminus \mathcal{R}(I) \end{cases} \end{array}$$

Example 10 *Consider interfaces $I_{00} := (\{x\},\{y\}, x=0 \to y=0)$ and $I_{01} := (\{x\},\{y\}, x=0 \to y=1)$. I_{00} and I_{01} are not shared-refinable because there is no way to satisfy $y=0 \land y=1$ when $x=0$.*

Lemma 11 *If I and I' are shared-refinable interfaces then*

$$\mathcal{R}(I) \cap \mathcal{R}(I') \subseteq \mathcal{R}(I \sqcap I') \subseteq \mathcal{R}(I) \cup \mathcal{R}(I')$$

Proof: Let $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$.

$\mathcal{R}(I) \cap \mathcal{R}(I') \subseteq \mathcal{R}(I \sqcap I')$: By induction on the length of states. It holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose $s \cdot a \in \mathcal{R}(I) \cap \mathcal{R}(I')$. Then $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, and from the induction hypothesis, $s \in \mathcal{R}(I \sqcap I')$. Since $s \cdot a \in \mathcal{R}(I)$, $a \models \xi(s)$. Since $s \cdot a \in \mathcal{R}(I')$, $a \models \xi'(s)$. Thus $a \models \xi(s) \land \xi'(s)$. Thus $a \models (\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))) \land (\mathsf{in}(\xi(s)) \to \xi(s)) \land (\mathsf{in}(\xi'(s)) \to \xi'(s)) \equiv \xi_\sqcap(s)$.

$\mathcal{R}(I \sqcap I') \subseteq \mathcal{R}(I) \cup \mathcal{R}(I')$: By induction on the length of states. It holds for the state of length zero, i.e., the empty state ε, because ε is reachable in any interface. Suppose $s \cdot a \in \mathcal{R}(I \sqcap I')$. Then $a \models \xi_\sqcap(s)$. Also, $s \in \mathcal{R}(I \sqcap I')$, and from the induction hypothesis, $s \in \mathcal{R}(I) \cup \mathcal{R}(I')$. Suppose $s \in \mathcal{R}(I)$ (the other case is symmetric). There are two sub-cases:

Case 1: $s \in \mathcal{R}(I')$: Then $\xi_\sqcap(s) \equiv (\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))) \land (\mathsf{in}(\xi(s)) \to \xi(s)) \land (\mathsf{in}(\xi'(s)) \to \xi'(s))$. Since $a \models \xi_\sqcap(s)$, $a \models (\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s)))$. Suppose $a \models \mathsf{in}(\xi(s))$ (the other case is symmetric). Then, since $a \models \mathsf{in}(\xi(s)) \to \xi(s)$, we have $a \models \xi(s)$, thus, $s \cdot a \in \mathcal{R}(I)$.

Case 2: $s \notin \mathcal{R}(I')$: Then $\xi_\sqcap(s) \equiv \xi(s)$, therefore, $a \models \xi(s)$, thus, $s \cdot a \in \mathcal{R}(I)$. ■

Lemma 12 *Let I and I' be shared-refinable interfaces such that $I = (X, Y, \xi)$, $I' = (X, Y, \xi')$ and $I \sqcap I' = (X, Y, \xi_\sqcap)$. Then, for all $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$*

$$\mathsf{in}(\xi_\sqcap(s)) \equiv \mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))$$

Proof: Using the fact that $\mathsf{in}(\xi(s))$ and $\mathsf{in}(\xi'(s))$ are properties over X, and the fact that the existential quantifier distributes over disjunctions, we can show the following equivalences:

$$\begin{aligned}
\mathsf{in}(\xi_\sqcap(s)) \equiv \exists Y : \big(\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))\big) \land \big(\mathsf{in}(\xi(s)) \to \xi(s)\big) \land \big(\mathsf{in}(\xi'(s)) \to \xi'(s)\big) \equiv \\
\big(\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))\big) \land \exists Y : \big(\neg\mathsf{in}(\xi(s)) \lor \xi(s)\big) \land \big(\neg\mathsf{in}(\xi'(s)) \lor \xi'(s)\big) \equiv \\
\big(\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))\big) \land \exists Y : \Big(\neg\mathsf{in}(\xi(s)) \land \neg\mathsf{in}(\xi'(s)) \lor \neg\mathsf{in}(\xi(s)) \land \xi'(s) \lor \xi(s) \land \neg\mathsf{in}(\xi'(s)) \lor \xi(s) \land \xi'(s)\Big) \equiv \\
\big(\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))\big) \land \Big(\neg\mathsf{in}(\xi(s)) \land \mathsf{in}(\xi'(s)) \lor \mathsf{in}(\xi(s)) \land \neg\mathsf{in}(\xi'(s)) \lor \big(\exists Y : \xi(s) \land \xi'(s)\big)\Big) \equiv \\
\neg\mathsf{in}(\xi(s)) \land \mathsf{in}(\xi'(s)) \lor \mathsf{in}(\xi(s)) \land \neg\mathsf{in}(\xi'(s)) \lor \big(\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))\big) \land \big(\exists Y : \xi(s) \land \xi'(s)\big)
\end{aligned}$$

Clearly, $\mathsf{in}(\xi_\sqcap(s))$ implies $\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))$. To show that $\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))$ implies $\mathsf{in}(\xi_\sqcap(s))$, it suffices to show that $\mathsf{in}(\xi(s)) \lor \mathsf{in}(\xi'(s))$ implies the last formula derived above, which is equivalent to showing that $\mathsf{in}(\xi(s)) \land \mathsf{in}(\xi'(s))$ implies $\exists Y : \xi(s) \land \xi'(s)$. This follows from the fact that I and I' are shared-refinable, i.e., from Condition (18). ■

Theorem 14 (Greatest lower bound) *If I and I' are shared-refinable interfaces then $(I \sqcap I') \sqsubseteq I$, $(I \sqcap I') \sqsubseteq I'$, and for any interface I'' such that $I'' \sqsubseteq I$ and $I'' \sqsubseteq I'$, we have $I'' \sqsubseteq (I \sqcap I')$.*

Proof: Since I and I' are shared-refinable, they have the same sets of input and output variables. Let $I = (X, Y, \xi)$ and $I' = (X, Y, \xi')$. Let $I \sqcap I' = (X, Y, \xi_\sqcap)$. We prove $(I \sqcap I') \sqsubseteq I$. We need to show that $\forall s \in \mathcal{R}(I \sqcap I') \cap \mathcal{R}(I)$, the formulae $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi_\sqcap(s))$ and $\mathsf{in}(\xi(s)) \land \xi_\sqcap(s) \to \xi(s)$ are valid. To see why the latter formula is valid, observe that $\mathsf{in}(\xi(s)) \land \xi_\sqcap(s)$ implies $\mathsf{in}(\xi(s)) \land (\mathsf{in}(\xi(s)) \to \xi(s))$, which in turn implies $\xi(s)$. We now prove $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi_\sqcap(s))$. Observe that $s \in \mathcal{R}(I \sqcap I') \cap \mathcal{R}(I)$ implies $s \in \mathcal{R}(I)$. We reason by cases: Case 1: $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$: Then the result follows by Lemma 12. Case 2: $s \in \mathcal{R}(I) \setminus \mathcal{R}(I')$:

Then $\xi_\sqcap(s) \equiv \xi(s)$ and the result follows trivially. This completes the proof for $(I \sqcap I') \sqsubseteq I$. The proof for $(I \sqcap I') \sqsubseteq I'$ is symmetric. Thus, $I \sqcap I'$ is a lower bound of I and I'.

To show that $I \sqcap I'$ is the *greatest* lower bound, consider stateless interface I'' such that $I'' \sqsubseteq I$ and $I'' \sqsubseteq I'$. Let $I'' = (X, Y, \xi'')$. To prove $I'' \sqsubseteq (I \sqcap I')$ we must show that for all $s \in \mathcal{R}(I'') \cap \mathcal{R}(I \sqcap I')$, the formulae $\mathsf{in}(\xi_\sqcap(s)) \to \mathsf{in}(\xi''(s))$ and $\mathsf{in}(\xi_\sqcap(s)) \wedge \xi''(s) \to \xi_\sqcap(s)$ are valid. By Lemma 11, $s \in \mathcal{R}(I \sqcap I')$ implies $s \in \mathcal{R}(I) \cup \mathcal{R}(I')$. We reason by cases:

Case 1: $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$: By Lemma 12, $\mathsf{in}(\xi_\sqcap(s)) \equiv \mathsf{in}(\xi(s)) \vee \mathsf{in}(\xi'(s))$. By $I'' \sqsubseteq I$ and $I'' \sqsubseteq I'$, we have $\mathsf{in}(\xi(s)) \to \mathsf{in}(\xi''(s))$ and $\mathsf{in}(\xi'(s)) \to \mathsf{in}(\xi''(s))$, and $\mathsf{in}(\xi_\sqcap(s)) \to \mathsf{in}(\xi''(s))$ follows. To show $\mathsf{in}(\xi_\sqcap(s)) \wedge \xi''(s) \to \xi_\sqcap(s)$ we derive:

$$\mathsf{in}(\xi_\sqcap(s)) \wedge \xi''(s) \equiv \big(\mathsf{in}(\xi(s)) \vee \mathsf{in}(\xi'(s))\big) \wedge \xi''(s)$$

We reason by sub-cases, using the facts $\mathsf{in}(\xi(s)) \wedge \xi''(s) \to \xi(s)$ and $\mathsf{in}(\xi'(s)) \wedge \xi''(s) \to \xi'(s)$, which come from $I'' \sqsubseteq I$ and $I'' \sqsubseteq I'$.

Case 1.1:

$$\big(\mathsf{in}(\xi(s)) \wedge \mathsf{in}(\xi'(s))\big) \wedge \xi''(s) \to \xi(s) \wedge \xi'(s) \to \xi_\sqcap(s)$$

Case 1.2:

$$\big(\mathsf{in}(\xi(s)) \wedge \neg\mathsf{in}(\xi'(s))\big) \wedge \xi''(s) \to \xi(s) \wedge \neg\mathsf{in}(\xi'(s)) \to \xi_\sqcap(s)$$

Case 1.3:

$$\big(\neg\mathsf{in}(\xi(s)) \wedge \mathsf{in}(\xi'(s))\big) \wedge \xi''(s) \to \neg\mathsf{in}(\xi(s)) \wedge \xi'(s) \to \xi_\sqcap(s)$$

In all three sub-cases we derived $\xi_\sqcap(s)$, therefore Case 1 is proven.

Case 2: $s \in \mathcal{R}(I) \setminus \mathcal{R}(I')$: Then $\xi_\sqcap(s) \equiv \xi(s)$ and the result follows by $I'' \sqsubseteq I$.

Case 3: $s \in \mathcal{R}(I') \setminus \mathcal{R}(I)$: Symmetric to Case 2. ■

Theorem 15 (Shared-refinement preserves well-formedness) *If I and I' are shared-refinable interfaces and both are well-formed, then $I \sqcap I'$ is well-formed.*

9 The input-complete case

Input-complete interfaces do not restrict the set of input values, although they may provide no guarantees when the input values are illegal. Although input-complete interfaces are a special case of general interfaces, it is instructive to study them separately for two reasons: first, input-completeness makes things much simpler, thus easier to understand and implement; second, some interesting results can be derived for input-complete interfaces but not in general.

Definition 16 (Input-complete interface) *An interface $I = (X, Y, \xi)$ is* input-complete *if for all $s \in \mathcal{A}(X \cup Y)^*$, $\mathsf{in}(\xi(s))$ is valid.*

Theorem 16 *Every input-complete interface is well-formed.*

Proof: Let $I = (X, Y, \xi)$ be an input-complete interface. Then $\mathsf{in}(\xi(s))$ is valid for all $s \in \mathcal{A}(X \cup Y)^*$, i.e., $\exists Y : \xi(s) \equiv \mathsf{true}$ for any assignment over X. Let a_X be an assignment over X (note that a_X is defined even when X is empty). Then there exists an assignment a_Y on Y such that the combined assignment (a_X, a_Y) on $X \cup Y$ satisfies $\xi(s)$. Thus, $\xi(s)$ is satisfiable, which means I is well-formed. ■

Definition 17 and Theorem 17 that follow show that every interface I can be turned into an input-complete interface $\mathsf{IC}(I)$ that refines I.

Definition 17 (Input-completion) *Consider an interface $I = (X, Y, \xi)$. The* input-complete version *of I, denoted $\mathsf{IC}(I)$, is the interface $\mathsf{IC}(I) := (X, Y, \xi_{ic})$, where $\xi_{ic}(s) := \xi(s) \vee \neg\mathsf{in}(\xi(s))$, for all $s \in \mathcal{A}(X \cup Y)^*$.*

Theorem 17 *If I is an interface then: (1) $\mathsf{IC}(I)$ is an input-complete interface, and (2) $\mathsf{IC}(I) \sqsubseteq I$.*

Proof: Let $I = (X, Y, \xi)$ and $\mathsf{IC}(I) = (X, Y, \xi_{ic})$. Let $s \in \mathcal{A}(X \cup Y)^*$.

(1) $\mathsf{in}(\xi_{ic}(s)) \equiv \exists Y : (\xi(s) \vee \neg\mathsf{in}(\xi(s))) \equiv (\exists Y : \xi(s)) \vee \neg\mathsf{in}(\xi(s)) \equiv \mathsf{in}(\xi(s)) \vee \neg\mathsf{in}(\xi(s)) \equiv \mathsf{true}$, thus, $\mathsf{IC}(I)$ is input-complete.

(2) Obviously, $\mathsf{in}(\xi(s)) \rightarrow \mathsf{in}(\xi_{ic}(s))$. We need to show that $(\mathsf{in}(\xi(s)) \wedge (\xi(s) \vee \neg\mathsf{in}(\xi(s)))) \rightarrow \xi(s)$. The premise can be rewritten as $(\mathsf{in}(\xi(s)) \wedge \xi(s)) \vee (\mathsf{in}(\xi(s)) \wedge \neg\mathsf{in}(\xi(s))) \equiv \mathsf{in}(\xi(s)) \wedge \xi(s)$, which clearly implies $\xi(s)$. ■

Theorems 17 and 11 imply that for any environment E, if $I \models E$ then $\mathsf{IC}(I) \models E$. The converse does not hold in general (see Examples 1 and 3, and observe that I_2 is the input-complete version of I_1).

Composition by connection reduces to conjunction of contracts for input-complete interfaces, and preserves input-completeness:

Theorem 18 (Connection preserves input-completeness) *Let $I_i = (X_i, Y_i, \xi_i)$, $i = 1, 2$, be disjoint input-complete interfaces, and let θ be a connection between I_1, I_2. Then the contract function ξ of the composite interface $\theta(I_1, I_2)$ is such that for all $s \in \mathcal{A}(X_{\theta(I_1,I_2)} \cup Y_{\theta(I_1,I_2)})^*$*

$$\xi(s) \quad \equiv \quad \xi_1(s) \wedge \xi_2(s) \wedge \rho_\theta$$

Moreover, $\theta(I_1, I_2)$ is input-complete.

Proof: Formula (4) is equivalent to true because $\mathsf{in}(\xi_2(s_2)) \equiv \mathsf{true}$. To see that $\theta(I_1, I_2)$ is input-complete, consider a state $s \in \mathcal{A}(X_{\theta(I_1,I_2)} \cup Y_{\theta(I_1,I_2)})^*$ and let a be an assignment over $X_{\theta(I_1,I_2)}$. Since $\mathsf{in}(\xi_1(s_1)) \equiv \mathsf{true}$, and $X_1 \subseteq X_{\theta(I_1,I_2)}$, there exists an assignment b over Y_1 such that $(a, b) \models \xi_1(s_1)$. Let c be an assignment over X_θ such that $(b, c) \models \rho_\theta$: such an assignment can always be found by setting $c(x)$ to the value that b assigns to y, where $(y, x) \in \theta$. Since $\mathsf{in}(\xi_2(s_2)) \equiv \mathsf{true}$, there exists an assignment d over Y_2 such that $(a, c, d) \models \xi_2(s_2)$. Combining the assignments we get $(a, b, c, d) \models \xi_1(s_1) \wedge \xi_2(s_2) \wedge \rho_\theta \equiv \xi(s)$, therefore, $\theta(I_1, I_2)$ is input-complete. ■

It is important to note that taking $\xi_1(s) \wedge \xi_2(s) \wedge \rho_\theta$ as the contract of a composite interface does not work for general interfaces, even though it works for input-complete interfaces. This is illustrated in the following example.

Example 11 *Let*

$$\begin{aligned} I_{10} \quad &:= \quad \big(\{x\}, \{y\}, x = 0 \wedge (y = 0 \vee y = 1)\big) \\ I_{12} \quad &:= \quad (\{z\}, \{w\}, z = 0 \wedge w = 0) \end{aligned}$$

Let $\theta := \{(y, z)\}$. The conjunction of the contracts of I_{10} and I_{12}, together with the equality $y = z$ imposed by the connection θ, gives the contract $x = 0 \wedge (y = 0 \vee y = 1) \wedge z = 0 \wedge w = 0 \wedge y = z$, which is equivalent to $x = y = z = w = 0$, which is clearly satisfiable. Therefore, we could interpret the composite interface $\theta(I_{10}, I_{12})$ as the interface

$$(\{x\}, \{y, z, w\}, x = y = z = w = 0)$$

Now, consider the interface:

$$I_{11} \quad := \quad (\{x\}, \{y\}, x = 0 \wedge y = 1)$$

It can be checked that $I_{11} \sqsubseteq I_{10}$. But if we connect I_{11} to I_{12}, we find that the conjunction of their contracts (with the connection $y = z$) is unsatisfiable. Therefore, if we used conjunction for composition by connection, then the composite interface $\theta(I_{11}, I_{12})$ would not refine $\theta(I_{10}, I_{12})$, even though I_{11} refines I_{10}, i.e., Theorem 12 would not hold.

Input-complete interfaces alone do not help in avoiding problems with arbitrary feedback compositions: indeed, in the example given in the introduction both interfaces I_{true} and $I_{y \neq x}$ are input-complete.[5] This means that in order to add a feedback connection (y, x) in an input-complete interface, we must still ensure that this interface is Moore w.r.t. input x. In that case, feedback preserves input-completeness.

Theorem 19 (Feedback preserves input-completeness) *Let $I = (X, Y, \xi)$ be an input-complete interface which is also Moore with respect to some $x \in X$. Let $\kappa = (y, x)$ be a feedback connection on I. Then $\kappa(I)$ is input-complete.*

Proof: By definition, $\kappa(I) = (X \setminus \{x\}, Y \cup \{x\}, \xi_\kappa)$, where $\xi_\kappa(s) \equiv \xi(s) \wedge (x = y)$, for all $s \in \mathcal{A}(X \cup Y)^*$. Let $s \in \mathcal{A}(X \cup Y)^*$. We must show that $\mathsf{in}(\xi_\kappa(s)) \equiv \exists Y \cup \{x\} : \xi(s) \wedge (x = y)$ is valid. Because $\xi(s)$ does not refer to x, we have $\exists Y \cup \{x\} : \xi(s) \wedge (x = y) \equiv \exists Y : \exists x : \xi(s) \wedge (x = y) \equiv \exists Y : (\xi(s) \wedge (\exists x : x = y)) \equiv \exists Y : \xi(s) \equiv \mathsf{in}(\xi(s)) \equiv \mathsf{true}$. ■

Theorem 20 (Hiding preserves input-completeness) *Let $I = (X, Y, \xi)$ be an input-complete interface and let $Y' \subseteq Y$, such that ξ is independent from Y'. Then, $\mathsf{hide}(Y', I)$ is input-complete.*

Proof: I is input-complete means $\mathsf{in}(\xi(s))$ is valid for all $s \in \mathcal{A}(X \cup Y)^*$. We must show that $\exists Y \setminus Y' : (\exists Y' : \xi(s))$ is valid: the latter formula is equivalent to $\exists Y : \xi(s)$, i.e., $\mathsf{in}(\xi(s))$. ■

Theorem 21 (Refinement for input-complete interfaces) *Let $I = (X, Y, \xi)$ and $I' = (X', Y', \xi')$ be input-complete interfaces. Then $I' \sqsubseteq I$ iff for all $s \in \mathcal{A}(X \cup Y)^*$, $\xi'(s) \rightarrow \xi(s)$ is valid.*

Proof: Follows directly from Definition 14 and the fact that $\mathsf{in}(\xi(s)) \equiv \mathsf{in}(\xi'(s)) \equiv \mathsf{true}$ for any $s \in \mathcal{A}(X \cup Y)^*$. ■

For input-complete interfaces, the shared-refinability condition, i.e., Condition (17), simplifies to

$$\forall X : \exists Y : \xi(s) \wedge \xi'(s)$$

Clearly, this condition does *not* always hold. Indeed, the interfaces of Example 10 are not shared-refinable, even though they are input-complete. For shared-refinable input-complete interfaces, shared refinement reduces to conjunction of contracts for states that are reachable in both interfaces.

Theorem 22 (Shared refinement for input-complete interfaces) *Let $I = (X, Y, \xi)$ and $I' = (X, Y, \xi')$ be input-complete shared-refinable interfaces. Then $I' \sqcap I = (X, Y, \xi_\sqcap)$, where for all $s \in \mathcal{R}(I) \cap \mathcal{R}(I')$, $\xi_\sqcap(s) \equiv \xi(s) \wedge \xi'(s)$.*

Proof: Follows directly from Definition 15 and the fact that $\mathsf{in}(\xi(s)) \equiv \mathsf{in}(\xi'(s)) \equiv \mathsf{true}$ for any $s \in \mathcal{A}(X \cup Y)^*$. ■

As the above presentation shows, input-complete interfaces are much simpler than general interfaces: refinement is implication of contracts, composition is conjunction, and so on. Then, a legitimate question is, why consider non-input-complete interfaces at all? There are mainly two reasons.

First, non-input-complete interfaces can be used to model situations that cannot be modeled by input-complete interfaces. For example, consider modeling a component implementing some procedure that requires certain conditions on its inputs to be satisfied, otherwise it may not terminate. We can capture the specification of this component as an interface, by imposing these conditions in the contract of the interface. But we cannot capture the same specification as an input-complete interface: for what would the output be

[5] It is not surprising that input-complete interfaces alone cannot solve the problems with arbitrary feedback compositions, since these are general problems of causality, not particular to interfaces.

when the input conditions are violated? We cannot simply add an extra output taking values in $\{T, NT\}$, for "terminates" and "does not terminate", since non-termination is not an observable property.

Second, even in the case where we could use input-complete interfaces to capture a specification, we may decide not to do so, in order to allow for *local compatibility* checks. In particular, when connecting two interfaces I and I', we may want to check that their composition is well-formed *before* proceeding to form an entire interface diagram. Input-complete interfaces are always well-formed and so are their compositions (Theorems 16, 18 and 19), therefore, local compatibility checks provide useful information only in the non-input-complete case.

10 Conclusions

The main message of this paper is that a theory of relational interfaces that has the desired properties *can* be developed, provided feedback compositions are restricted appropriately.

In the future, we plan to further extend this theory. One useful extension would be to refine the definition of Moore interfaces to speak about dependencies between specific pairs of input and output variables. This would allow to express, for example, the fact that in the parallel composition of $(\{x_1\}, \{y_1\}, x_1 = y_1)$ and $(\{x_2\}, \{y_2\}, x_2 = y_2)$, y_1 does not depend on x_2 and y_2 does not depend on x_1 (and therefore one of the feedbacks (y_1, x_2) or (y_2, x_1) can be allowed). Such an extension could be achieved by combining our relational interfaces with the *causality interfaces* of [9], or the coarser *profiles* of [7].

References

[1] R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *CONCUR'98*, volume 1466 of *LNCS*. Springer, 1998.

[2] A. Chakrabarti, L. de Alfaro, T. Henzinger, and F. Mang. Synchronous and bidirectional component interfaces. In *CAV*, LNCS 2404, pages 414–427. Springer, 2002.

[3] L. de Alfaro and T. Henzinger. Interface automata. In *Foundations of Software Engineering (FSE).* ACM Press, 2001.

[4] L. de Alfaro and T. Henzinger. Interface theories for component-based design. In *EMSOFT'01.* Springer, LNCS 2211, 2001.

[5] L. Doyen, T. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *8th ACM & IEEE International conference on Embedded software, EMSOFT*, pages 79–88, 2008.

[6] E. Lee and A. Sangiovanni-Vincentelli. A unified framework for comparing models of computation. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, Dec. 1998.

[7] R. Lublinerman and S. Tripakis. Modularity vs. Reusability: Code Generation from Synchronous Block Diagrams. In *Design, Automation, and Test in Europe (DATE'08).* ACM, Mar. 2008.

[8] G. Tourlakis. *Mathematical Logic.* Wiley, 2008.

[9] Y. Zhou and E. Lee. Causality interfaces for actor networks. *ACM Trans. Embed. Comput. Syst.*, 7(3):1–35, 2008.