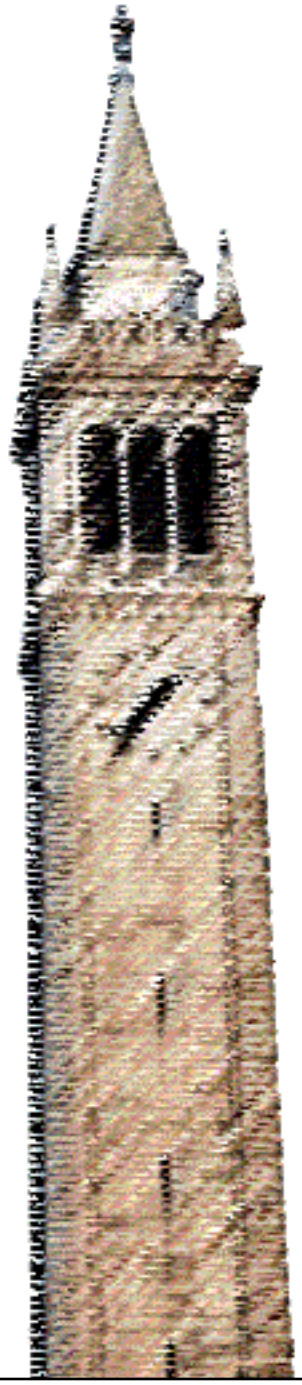
Hypercontracts



Inigo Incer
Albert Benveniste
Alberto L. Sangiovanni-Vincentelli
Sanjit A. Seshia

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-158
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-158.html

May 31, 2021

Hypercontracts

INIGO INCER, University of California, Berkeley, USA
ALBERT BENVENISTE, INRIA/IRISA, Rennes, France
ALBERTO L. SANGIOVANNI-VINCENTELLI, University of California, Berkeley, USA
SANJIT A. SESHIA, University of California, Berkeley, USA

Contracts (or interface) theories have been proposed to formally support distributed and decentralized system design while ensuring safe system integration. Over the last decades, a number of formalisms were proposed, sometimes very different in their form and algebra. This motivated the quest for a unification by some authors, e.g., specifications through contracts by Bauer et al. and the contract metatheory by Benveniste et al. to cite a few. These generic models establish precise links between the different contract frameworks. In this paper we propose *hypercontracts*, a generic model with a richer structure for its underlying model of components, subsuming simulation preorders. While this new model remains generic, it provides a much more elegant and richer algebra for its key notions of refinement, parallel composition, and quotient, and it allows considering new operations. On top of these foundations, we propose *conic hypercontracts*, which are still generic but come with a finite description. We show how to specialize conic hypercontracts to Assume-Guarantee contracts as well as to Interface Automata, two known contract frameworks very different in style. We illustrate conic hypercontracts on specifications involving security and the robustness of machine-learning components.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; **Embedded and cyber-physical systems**; • **Theory of computation → Formalisms**; **Formalisms**; • **Computing methodologies → Modeling methodologies**.

Additional Key Words and Phrases: contracts, hypercontracts, contract-based design, component-based design, assume-guarantee, hyperproperties, downward-closed

1 INTRODUCTION

The need for compositional algebraic frameworks to design and analyze cyber-physical systems is widely recognized. The aim is to support distributed and decentralized system design based on a proper definition of *interfaces* supporting the specification of subsystems having a partially specified context of operation, and subsequently guaranteeing safe system integration. Over the last few decades, we have seen the introduction of several formalisms to do this: interface automata [6, 9–11, 20], process spaces [23], modal interfaces [3, 17–19, 26], assume-guarantee contracts [4], rely-guarantee reasoning [8, 13, 15, 16], and variants of these. The system interfaces state (i) what the component guarantees and (ii) what it assumes from its environment in order for those guarantees to hold. That is, all these frameworks implement a form of assume-guarantee reasoning.

These algebraic frameworks have a notion of a component, of an environment, and of a specification, also called contract to stress the give-and-take dynamics between the component and its environment. They all have notions of satisfaction of a specification by a component, and of contract composition. The fundamental object they manipulate is the specification. Since many algebraic frameworks have been proposed, there have been efforts to systematize this knowledge by building a high-level theory of which these algebras are instantiations. Thus, Bauer et al. [2] describe how to build a contract theory if one has a specification theory available. Benveniste et al. [5] provide a meta-theory that builds contracts starting from an algebra of components. They provide several operations on contracts and show how this meta-theory can describe, e.g., interface automata, assume-guarantee contracts, modal interfaces, and rely-guarantee reasoning. This meta-theory is, however, low-level, specifying contracts as unstructured sets of environments and

Authors' addresses: Inigo Incer, University of California, Berkeley, USA, inigo@eecs.berkeley.edu; Albert Benveniste, albert.benveniste@inria.fr, INRIA/IRISA, Rennes, France; Alberto L. Sangiovanni-Vincentelli, University of California, Berkeley, USA, alberto@eecs.berkeley.edu; Sanjit A. Seshia, University of California, Berkeley, USA, sseshia@eecs.berkeley.edu.

implementations. As a consequence, important concepts such as parallel composition and quotient of contracts are expressed in too abstract terms, and parallel composition was not associative (only "sub-associative") unless more assumptions were formulated—see [5], chapter 4.

In this work, we provide an improved theory of contracts, called *hypercontracts*, that addresses the above deficiencies, by giving more structure while defining environments and implementations. This theory is built in three stages. We begin with a theory of components. Then we state what are the sets of components that our theory can express; we call such objects compsets, which are equivalent to hyperproperties in behavioral formalisms [22]. From these compsets, we build hypercontracts. We provide closed-form expressions for hypercontract manipulations. Then we show how the hypercontract theory applies to two specific cases: downward-closed hypercontracts and interface hypercontracts (equivalent to interface automata).

Our contributions are the following: (i) a new model of *hypercontracts* possessing a richer algebra than the metatheory of [5], (ii) a calculus of *conic hypercontracts* offering finite representations of downward-closed hypercontracts, (iii) a reformulation of de Alfaro-Henzinger interface automata [9] by specializing the components of hypercontracts to be receptive languages, and (iv) illustrations of hypercontracts addressing secure information flow and the robustness of data-driven components used in safety-critical applications.

2 PRELIMINARIES

Preorders. Many concepts in this paper will be inherited from preorders. We recall that a preorder $(P, \leq)$ consists of a set P and a relation $\leq$ which is transitive (i.e., $a \leq b$ and $b \leq c$ implies that $a \leq c$ for all $a, b, c \in P$) and reflexive ($a \leq a$ for all $a \in P$). A partial order is a preorder whose relation is also antisymmetric (i.e., from $a \leq b$ and $b \leq a$ we conclude that $a = b$).

Our preorders will come equipped with a partial binary operation called composition, usually denoted $\times$. Composition is often understood as a means of connecting elements together and is assumed to be monotonic in the preorder, i.e., we assume composing with bigger elements yields bigger results: $\forall a, b, c \in P.\ a \leq b \Rightarrow a \times c \leq b \times c$. We will also be interested in taking elements apart. For a notion of composition, we can always ask the question, for $a, b \in P$, what is the largest element $b \in P$ such that $a \times b \leq c$? Such an element is called *quotient* or *residual*, usually denoted c/a. Formally, the definition of the quotient c/a is

$$\forall b \in P.\ a \times b \leq c \text{ if and only if } b \leq c/a, \tag{1}$$

which means that the quotient is the right adjoint of composition (in the sense of category theory). A synonym of this notion is to say that composing by a fixed element a (i.e., $b \mapsto a \times b$) and taking quotient by the same element (i.e., $c \mapsto c/a$) form a Galois connection. A description of the use of the quotient in many fields of engineering and computer science is given in [14]. From this abstract definition, we can obtain an important property of the quotient:

Proposition 2.1. *The quotient is monotonic in the first argument and antitone (i.e., order-reversing) in the second.*

A partial order for which every two elements have a well-defined LUB (aka join), denoted $\vee$, and GLB (aka meet), denoted $\wedge$, is a lattice. A lattice in which the meet has a right adjoint is called Heyting algebra. This right adjoint usually goes by the name exponential, denoted $\rightarrow$. In other words, the exponential is the notion of quotient if we take composition to be given by the meet, that is, for a Heyting algebra H with elements a, c, the exponential is defined as

$$\forall b \in H.\ a \wedge b \leq c \text{ if and only if } b \leq a \rightarrow c, \tag{2}$$

which is the familiar notion of implication in Boolean algebras.

Hyperproperties. The traditional definition of a property in the formal methods community is "a set of traces." This notion is based on the behavioral approach to system modelling: we assume we start with a set of behaviors $\mathcal{B}$, and properties are defined as subsets of $\mathcal{B}$. In this approach, design elements or components are also defined as subsets of $\mathcal{B}$. The difference between components and properties is semantics: a component collects the behaviors that can be observed from that component, while a property collects the behaviors meeting some criterion of interest. We say a component M satisfies a property P, written $M \models P$, when $M \subseteq P$, that is, when the behaviors of M meet the criterion that determines P.

Properties of this sort are also called *trace properties*. Many design qualities are of this type, such as safety. But there are many system attributes that can only be determined by analyzing multiple traces: mean response times, security attributes, reliability, etc. This suggests the need for a richer formalism for expressing design attributes.

Hyperproperties are instead subsets of $2^{\mathcal{B}}$. A component M satisfies a hyperproperty H if $M \in H$. Since hyperproperties allow us to define exactly what components satisfy them, we can define them using any number of behaviors of a component (as opposed to trace properties which can only predicate about single traces). For example, one can specify secure-information-flow attributes as follows: through a non-interference hyperproperty, we can state that a system user having unprivileged credentials should see the same output to her execution anytime she runs the system with the same inputs; in other words, the data of other users should not leak to what she sees. This statement cannot be expressed as a trace property [7].

3 THE THEORY OF HYPERCONTRACTS

Our objective is to develop a theory of assume-guarantee reasoning for any kind of attribute of cyber-physical systems. We do this in three steps:

(1) we consider components coming with notions of preorder (e.g., simulation) and parallel composition;
(2) we discuss the notion of a compset and give it some structure—unlike the unstructured sets of components considered in the metatheory of [5];
(3) we build hypercontracts as pairs of compsets with additional structure—capturing environments and implementations.

In this section we describe how this construction is performed, and in the next we show how some existing and new assume-guarantee theories are instances of our considerations of this section. We will use the theory of assume-guarantee contracts as a running example for how these notions map to existing frameworks.

3.1 Components

In the theory of hypercontracts, the most primitive concept is the component. Let $(\mathbb{M}, \leq)$ be a preorder. The elements $M \in \mathbb{M}$ are called *components*. We say that M is a subcomponent of M' when $M \leq M'$. If we represented components as automata, the statement "is a subcomponent of" is equivalent to "is simulated by."

There exists a partial binary operation, $\times : \mathbb{M}, \mathbb{M} \to \mathbb{M}$, monotonic in both arguments, called *composition*. If $M \times M'$ is not defined, we say that M and M' are not composable (and composable otherwise). A component E is an environment for component M if E and M are composable. We assume that composition is associative and commutative.

Similarly, we assume the existence of a second, partial binary operation which is the right adjoint of composition. This is the quotient (1) for the component theory. Given two components M and

M', the quotient, denoted M/M', is the largest component M'' satisfying $M' \times M'' \leq M$. In other words, it gives us the largest component whose composition with M' is a subcomponent of M.

Running example. In the behavioral approach to system modeling, we start with a set $\mathcal{B}$ whose elements we call behaviors. Components are defined as subsets of $\mathcal{B}$. They contain the behaviors they can display. A component M is a subcomponent of M' if M' contains all the behaviors of M, i.e., if $M \subseteq M'$. Component composition is given by set intersection: $M \times M' \stackrel{\text{def}}{=} M \cap M'$. If we represent the components as $M = \{b \in \mathcal{B} \mid \phi(b)\}$ and $M' = \{b \in \mathcal{B} \mid \phi'(b)\}$ for some constraints ϕ and ϕ', then composition is $M \times M' = \{b \in \mathcal{B} \mid \phi(b) \wedge \phi'(b)\}$, i.e., the behaviors that simultaneously meet the constraints of M and M'. This notion of composition is independent of the connection topology: the topology is inferred from the behaviors of the components. The quotient is given by implication: $M/M' = M' \rightarrow M$.

3.2 Compsets

CmpSet is a lattice whose objects are sets of components, called *compsets*. In general, not every set of components is necessarily an object of **CmpSet**.

CmpSet comes with a notion of satisfaction. Suppose $M \in \mathbb{M}$ and H is a compset. We say that M *satisfies* H or conforms to H, written $M \models H$, when $M \in H$. For compsets H, H', we say that H *refines* H', written $H \leq H'$, when $M \models H \Rightarrow M \models H'$, i.e., when $H \subseteq H'$.

Since we assume **CmpSet** is a lattice, the greatest lower bounds and least upper bounds of finite sets are defined. Observe, however, that although the partial order of **CmpSet** is given by subsetting, the meet and join of **CmpSet** are not necessarily intersection and union, respectively, as the union or intersection of any two elements are not necessarily elements of **CmpSet**.

3.2.1 Composition and quotient. We extend the operation of composition to **CmpSet**:

$$H \times H' = \{M \times M' \mid M \models H, M' \models H', \text{ and } M \text{ and } M' \text{ are composable}\}. \tag{3}$$

Composition is total and monotonic, i.e., if $H' \leq H''$, then $H \times H' \leq H \times H''$. It is also commutative and associative, by the commutativity and associativity, respectively, of component composition.

We assume the existence of a second (but partial) binary operation on the objects of **CmpSet**. This operation is the right adjoint of composition: for compsets H and H', the residual H/H' (also called *quotient*), is defined by the universal property (1). From the definition of composition, we must have

$$H/H' = \{M \in \mathbb{M} \mid \{M\} \times H' \subseteq H\}. \tag{4}$$

Running example. Assume $\mathbb{M}$ contains behavioral components, as in the previous example. We can instantiate trace properties as a lattice **CmpSet**. Each compset is of the form $H = 2^M$ for some component $M \subseteq \mathcal{B}$. Observe that the satisfaction of a compset by a component $M' \in 2^M$ happens if and only if $M' \leq M$. The meet of two compsets $2^M \wedge 2^{M'}$ is $2^{M \cap M'} = 2^M \cap 2^{M'}$, but the join of two elements $2^M \vee 2^{M'}$ is $2^{M \cup M'} \neq 2^M \cup 2^{M'}$. The composition of two compsets is given by $2^M \times 2^{M'} = 2^{M \cap M'}$, and the quotient is $2^M / 2^{M'} = 2^{M/M'}$, where the quotient M/M' was defined in the introduction of the running example in page 4.

3.2.2 Convexity, co-convexity, and flatness. A compset H is *convex* if $M, M' \models H \Rightarrow M \times M' \models H$. In other words, H is convex if $H \times H \leq H$. A compset H is *co-convex* if $H \leq H \times H$.

We can say that convex compsets are those such that the composition of two components that satisfy them also satisfies the compset. Conversely, co-convex compsets are those whose components can be expressed as the product of two components that satisfy them. If a compset is both convex and co-convex, it is called *flat*. Flat compsets H are precisely those that satisfy $H = H \times H$. If all compsets are flat, composition in **CmpSet** is idempotent.

Proposition 3.1. *Convexity, co-convexity, and flatness are preserved under composition.*

Proposition 3.2. *If component composition is idempotent, all elements of* **CmpSet** *are co-convex.*

3.2.3 Downward-closed compsets. The set of components was introduced with a partial order. We say that a compset H is *downward-closed* when $M' \leq M$ and $M \models H$ imply $M' \models H$, i.e., if a component satisfies a downward-closed compset, so does its subcomponent. Section 4.2 treats downward-closed compsets in detail.

3.3 Hypercontracts

Hypercontracts as pairs (environments, closed-system specification). A hypercontract is a specification for a design element that tells what is required from the design element when it operates in an environment that meets the expectations of the hypercontract. A hypercontract is thus a pair of compsets:

$$C = (\mathcal{E}, \mathcal{S}) = \text{(environments, closed-system specification)}.$$

$\mathcal{E}$ states the environments in which the object being specified must adhere to the specification. $\mathcal{S}$ states the requirements that the design element must fulfill when operating in an environment which meets the expectations of the hypercontract. We say that a component E *is an environment of hypercontract* C, written $E \models^E C$, if $E \models \mathcal{E}$. We say that a component M *is an implementation of* C, written $M \models^I C$, when $M \times E \models \mathcal{S}$ for all $E \models \mathcal{E}$. We thus define the set of implementations $\mathcal{I}$ of C as the compset containing all implementations, i.e., as the quotient:

$$\text{implementations} = \mathcal{I} = \mathcal{S}/\mathcal{E}.$$

A hypercontract with a nonempty set of environments is called *compatible*; if it has a nonempty set of implementations, it is called *consistent*. For $\mathcal{S}$ and $\mathcal{I}$ as above, the compset $\mathcal{E}'$ defined as $\mathcal{E}' = \mathcal{S}/\mathcal{I}$ contains all environments in which the implementations of C satisfy the specifications of the hypercontract. Thus, we say that a hypercontract is saturated if its environments compset is as large as possible in the sense that adding more environments to the hypercontract would reduce its implementations. This means that C satisfies the following fixpoint equation:

$$\mathcal{E} = \mathcal{S}/\mathcal{I} = \mathcal{S}/(\mathcal{S}/\mathcal{E}).$$

Hypercontracts as pairs (environments, implementations). Another way to interpret a hypercontract is by telling explicitly which environments and implementations it supports. Thus, we would write the hypercontract as $C = (\mathcal{E}, \mathcal{I})$. We will see that assume-guarantee theories can differ as to what is the most convenient representation for their hypercontracts.

The lattice **Contr** *of hypercontracts.* Just as with **CmpSet**, we define **Contr** as a lattice formed by putting together two compsets in one of the above two ways. Not every pair of compsets is necessarily a valid hypercontract. We will define soon the operations that give rise to this lattice.

3.3.1 Preorder. We define a preorder on hypercontracts as follows: we say that C *refines* C', written $C \leq C'$, when every environment of C' is an environment of C, and every implementation of C is an implementation of C', i.e., $E \models^E C' \Rightarrow E \models^E C$ and $M \models^I C \Rightarrow M \models^I C'$. We can express this as

$$\mathcal{E}' \leq \mathcal{E} \text{ and } \mathcal{S}/\mathcal{E} = \mathcal{I} \leq \mathcal{I}' = \mathcal{S}'/\mathcal{E}'.$$

Any two C, C' with $C \leq C'$ and $C' \leq C$ are said to be *equivalent* since they have the same environments and the same implementations. We now obtain some operations using preorders which are defined as the LUB or GLB of **Contr**. We point out that the expressions we obtain are unique up to the preorder.

Running example. Assume-guarantee contracts are often given as a pair of trace-properties (A, G), where A states the assumptions made on the environment, and G states what the component

in question should guarantee when operating in a valid environment (i.e., one that meets the assumptions). We observe that any closed system obtained using environments that meet the assumptions is restricted to $G \cap A$; thus, we set the closed-system spec to $\mathcal{S} = 2^{A \cap G}$. Define the hypercontract $C = (2^A, 2^{A \cap G})$. The environments are $\mathcal{E} = 2^A$, namely, all $E \subseteq A$, and the implementations are $\mathcal{I} = 2^{(A \cap G)/A} = 2^{G/A}$, that is, all $M \subseteq G/A$. Observe that $\mathcal{S}/\mathcal{I} = \mathcal{E}$, so C is saturated. Now suppose we have another hypercontract $C' = (2^{A'}, 2^{A' \cap G'})$ with environments $\mathcal{E}'$ and implementations $\mathcal{I}'$. We observe that $\mathcal{E} \leq \mathcal{E}'$ if and only if $A \subseteq A'$; moreover, $\mathcal{I}' \leq \mathcal{I}$ if and only if $G'/A' \leq G/A$. This means that $C' \leq C$ if and only if the assume-guarantee contracts (A, G) and (A', G') satisfy $(A', G') \leq (A, G)$.

3.3.2 GLB and LUB. From the preorder just defined, the GLB of C and C' satisfies: $M \models^I C \wedge C'$ if and only if $M \models^I C$ and $M \models^I C'$; and $E \models^E C \wedge C'$ if and only if $E \models^E C$ or $E \models^E C'$. If we write $C = (\mathcal{E}, \mathcal{I})$, $C' = (\mathcal{E}', \mathcal{I}')$, this means that

$$C \wedge C' = \bigvee \{C'' = (\mathcal{E}'', \mathcal{I}'') \in \mathbf{Contr} \mid \mathcal{I}'' \leq \mathcal{I} \wedge \mathcal{I}' \text{ and } \mathcal{E} \vee \mathcal{E}' \leq \mathcal{E}''\}. \tag{5}$$

Conversely, the LUB satisfies $M \models^I C \vee C'$ if and only if $M \models^I C$ or $M \models^I C'$, and $E \models^E C \vee C'$ if and only if $E \models^E C$ and $E \models^E C'$, which means that we can write

$$C \vee C' = \bigwedge \{C'' = (\mathcal{E}'', \mathcal{I}'') \in \mathbf{Contr} \mid \mathcal{I} \vee \mathcal{I}' \leq \mathcal{I}'' \text{ and } \mathcal{E}'' \leq \mathcal{E} \wedge \mathcal{E}'\}. \tag{6}$$

The lattice **Contr** has hypercontracts for objects, and meet and join as just described.

3.3.3 Parallel composition. The composition of hypercontracts $C = (\mathcal{E}, \mathcal{I})$ and $C' = (\mathcal{E}', \mathcal{I}')$, denoted $C \parallel C'$, is the smallest hypercontract $C'' = (\mathcal{E}'', \mathcal{I}'')$ (up to equivalence) meeting the following requirements:

- any composition of two implementations of C and C' is an implementation of C''; and
- any composition of an environment of C'' with an implementation of C yields an environment for C' and vice-versa.

These requirements were stated for the first time by Abadi and Lamport [1]. We can write

$$\begin{aligned} C \parallel C' &= \bigwedge \left\{ C'' = (\mathcal{E}'', \mathcal{I}'') \;\middle|\; \left[\begin{array}{l} \mathcal{I} \times \mathcal{I}' \leq \mathcal{I}'', \\ \mathcal{E}'' \times \mathcal{I} \leq \mathcal{E}', \text{ and} \\ \mathcal{E}'' \times \mathcal{I}' \leq \mathcal{E} \end{array} \right] \right\} \\ &= \bigwedge \left\{ C'' = (\mathcal{E}'', \mathcal{I}'') \;\middle|\; \left[\begin{array}{l} \mathcal{I} \times \mathcal{I}' \leq \mathcal{I}'', \text{ and} \\ \mathcal{E}'' \leq \frac{\mathcal{E}'}{\mathcal{I}} \wedge \frac{\mathcal{E}}{\mathcal{I}'} \end{array} \right] \right\}. \end{aligned} \tag{7}$$

Running example. In our previous example, we saw that an assume-guarantee contract (A, G) can be mapped to an environment/implementation hypercontract $C = (2^A, 2^{G/A})$. Suppose we have a second hypercontract $C' = (2^{A'}, 2^{G'/A'})$. Applying (7), we obtain a composition formula for these hypercontracts: $C \parallel C' = \left(2^{A'/(G/A)} \wedge 2^{A/(G'/A')}, 2^{G/A} \wedge 2^{G'/A'}\right)$, whose environments and implementations are exactly those obtained from the composition of the assume-guarantee contracts (A, G) and (A', G') [4].

3.3.4 Mirror or reciprocal. We assume we have an additional operation on hypercontracts, called both mirror and reciprocal, which flips the environments and implementations of a hypercontract: $C^{-1} = (\mathcal{E}, \mathcal{I})^{-1} = (\mathcal{I}, \mathcal{E})$ and $C^{-1} = (\mathcal{E}, \mathcal{S})^{-1} = (\mathcal{S}/\mathcal{E}, \mathcal{S})$. This notion gives us, so to say, the hypercontract obeyed by the environment. The introduction of this operation assumes that for every hypercontract C, its reciprocal is also an element of **Contr**. Moreover, we assume that, when the infimum of a collection of hypercontracts exists, the following identity holds:

$$\left(\bigwedge_i C_i\right)^{-1} = \bigvee_i {C_i}^{-1}. \tag{8}$$

3.3.5 Hypercontract quotient. The *quotient* or residual for hypercontracts $C = (\mathcal{E}, \mathcal{I})$ and $C'' = (\mathcal{E}'', \mathcal{I}'')$, written C''/C, has the universal property (1), namely $\forall C'.\ C \parallel C' \leq C''$ if and only if $C' \leq C''/C$. We can obtain a closed-form expression using the reciprocal:

Proposition 3.3. *The hypercontract quotient obeys the relation* $C''/C = \left((C'')^{-1} \parallel C\right)^{-1}$.

3.3.6 Merging. The composition of two hypercontracts yields the specification of a system comprised of two design objects, each adhering to one of the hypercontracts being composed. Another important operation on hypercontracts is viewpoint merging, or *merging* for short. It can be the case that the same design element is assigned multiple specifications corresponding to multiple viewpoints, or design concerns [4, 24] (e.g., functionality and a performance criterion). Suppose $C_1 = (\mathcal{E}_1, \mathcal{S}_1)$ and $C_2 = (\mathcal{E}_2, \mathcal{S}_2)$ are the hypercontracts we wish to merge. Two slightly different operations can be considered as candidates for formalizing viewpoint merging:

- A *weak merge* which is the GLB (5); and
- A *strong merge* which states that environments of the merger should be environments of both C_1 and C_2 and that the closed systems of the merger are closed systems of both C_1 and C_2. If we let $C_1 \bullet C_2 = (\mathcal{E}, \mathcal{I})$, we have

$$\mathcal{E} = \vee \{\mathcal{E}' \in \mathbf{CmpSet} \mid \mathcal{E}' \leq \mathcal{E}_1 \wedge \mathcal{E}_2 \text{ and } \exists C'' = (\mathcal{E}'', \mathcal{I}'') \in \mathbf{Contr}.\ \mathcal{E}' = \mathcal{E}''\} \text{ and}$$
$$\mathcal{I} = \vee \{\mathcal{I}' \in \mathbf{CmpSet} \mid \mathcal{I}' \leq (\mathcal{S}_1 \wedge \mathcal{S}_2)/\mathcal{E} \text{ and } (\mathcal{E}, \mathcal{I}) \in \mathbf{Contr}\}.$$

The difference is that, whereas the commitment to satisfy $\mathcal{S}_2$ survives when under the weak merge when the environment fails to satisfy $\mathcal{E}_1$, no obligation survives under the strong merge. This distinction was proposed in [27] under the name of weak/strong assumptions.

Running example. Given two assume-guarantee contracts (A_i, G_i) for $i = 1, 2$, we consider the merging of their hypercontracts. We have $(2^{A_1}, 2^{A_1 \cap G_1}) \bullet (2^{A_2}, 2^{A_2 \cap G_2}) = (2^{A_1 \cap A_2}, 2^{A_1 \cap G_1 \cap A_2 \cap G_2})$. Observe that this last hypercontract has environments $2^{A_1 \cap A_2}$ and implementations $2^{(G_1 \cap G_2)/(A_1 \cap A_2)}$. This is the definition of merging for assume-guarantee contracts [24].

We presented the theory of hypercontracts. Now we will consider specializations of this theory to behavioral and interface formalisms.

4 BEHAVIORAL MODELING

Under behavioral modeling, design components are represented by the behaviors they can display. Fix once and for all a set $\mathcal{B}$ whose elements we call *behaviors*. We set $\mathbb{M} = 2^{\mathcal{B}}$. In this modeling philosophy, to be a subcomponent of a component is equivalent to being its subset. The composition of M and M' yields the component supporting the behaviors that both M and M' support. This means that $M \times M' = M \cap M'$.

We will consider two ways of building the **CmpSet** lattice: first we will allow it to contain any set of components, and then we will only allow it to contain downward-closed compsets.

4.1 General hypercontracts

The most expressive behavioral theory of hypercontracts is obtained when we place no restrictions on the structure of compsets and hypercontracts. In this case, the elements of **CmpSet** are all objects $H \in 2^{2^{\mathcal{B}}}$, i.e., all hyperproperties. The meet and join of compsets are set intersection and union, respectively, and their composition and quotient are given by (3) and (4), respectively. Hypercontracts are of the form $C = (\mathcal{E}, \mathcal{I})$ with all extrema achieved in the binary operations, i.e., for a second hypercontract $C' = (\mathcal{E}', \mathcal{I}')$, (5), (6), and (7) are, respectively,

$$C \wedge C' = (\mathcal{E} \cup \mathcal{E}', \mathcal{I} \cap \mathcal{I}'), C \vee C' = (\mathcal{E} \cap \mathcal{E}', \mathcal{I} \cup \mathcal{I}'), \text{ and } C \parallel C' = \left(\tfrac{\mathcal{E}'}{\mathcal{I}} \cap \tfrac{\mathcal{E}}{\mathcal{I}'}, \mathcal{I} \times \mathcal{I}'\right).$$

From these follow the operations of quotient, and merging.

4.2 Conic (or downward-closed) hypercontracts

We assume that **CmpSet** contains exclusively downward-closed hyperproperties. Let $H \in \mathbf{CmpSet}$. We say that $M \models H$ is a maximal component of H when H contains no set bigger than M, i.e., if

$$\forall M' \models H.\ M \leq M' \Rightarrow M' = M.$$

We let $\overline{H}$ be the set of maximal components of H:

$$\overline{H} = \{M \models H \mid \forall M' \models H.\ M \leq M' \Rightarrow M' = M\}.$$

Due to the fact H is downward-closed, the set of maximal components is a unique representation of H. We can express H as

$$H = \bigcup_{M \in \overline{H}} 2^M.$$

We say that H is *k-conic* if the cardinality of $\overline{H}$ is finite and equal to k, and we write this

$$H = \langle M_1, \ldots, M_k \rangle, \text{ where } \overline{H} = \{M_1, \ldots, M_k\}.$$

4.2.1 Order. The notion of order on **CmpSet** can be expressed using this notation as follows: suppose $H' = \langle M' \rangle_{M' \in \overline{H}'}$. Then

$$H' \leq H \text{ if and only if } \forall M' \in \overline{H}'\ \exists M \in \overline{H}.\ M' \leq M.$$

4.2.2 Composition. Composition in **CmpSet** becomes

$$H \times H' = \bigcup_{\substack{M \in \overline{H} \\ M' \in \overline{H}'}} 2^{M \cap M'} = \langle M \cap M' \rangle_{\substack{M \in \overline{H} \\ M' \in \overline{H}'}}. \tag{9}$$

Therefore, if H and H' are, respectively, k- and k'-conic, $H \times H'$ is at most kk'-conic.

4.2.3 Quotient. Suppose H_q satisfies

$$H' \times H_q \leq H.$$

Let $M_q \in \overline{H}_q$. We must have

$$M_q \times M' \models H \text{ for every } M' \in \overline{H}',$$

which means that for each $M' \in \overline{H}'$ there must exist an $M \in \overline{H}$ such that $M_q \times M' \leq M$; let us denote by $M(M')$ a choice $M' \mapsto M$ satisfying this condition. Therefore, we have

$$M_q \leq \bigwedge_{M' \in \overline{H}'} \frac{M(M')}{M'}, \tag{10}$$

Clearly, the largest such M_q is obtained by making (10) an equality. Thus, the cardinality of the quotient is bounded from above by $k^{k'}$ since we have

$$H_q = \left\langle \bigwedge_{M' \in \overline{H}'} \frac{M(M')}{M'} \right\rangle_{\substack{M(M') \in \overline{H} \\ \forall M' \in \overline{H}'}}. \tag{11}$$

4.2.4 Contracts. Now we assume that the objects of **CmpSet** are pairs of *downward-closed compsets*. If we have two hypercontracts $C = (\mathcal{E}, \mathcal{I})$ and $C' = (\mathcal{E}', \mathcal{I}')$, their composition is

$$C \parallel C' = \left(\frac{\mathcal{E}}{\mathcal{I}'} \wedge \frac{\mathcal{E}'}{\mathcal{I}}, \mathcal{I} \times \mathcal{I}' \right). \tag{12}$$

We can also write an expression for the quotient of two hypercontracts:

$$C/C' = \left(\mathcal{E} \times \mathcal{I}', \frac{\mathcal{I}}{\mathcal{I}'} \wedge \frac{\mathcal{E}'}{\mathcal{E}} \right). \tag{13}$$

Now that we have covered the theory of hypercontracts and its specialization to behavioral theories, we will consider two applications.

4.3 Application: Secure information flow

Secure information flow is a prototypical example of a design quality which trace properties are unable to capture. It can be expressed with hyperproperties, and is in fact one reason behind their introduction. A common information-flow attribute is *non-interference*, introduced by Goguen and Meseguer [12]. It states that privileged data does not leak to an unprivileged path. Suppose σ is one of the behaviors that our system can display, understood as the state of memory locations through time. Some of those memory locations we call *privileged*, some *unprivileged*. Let $L_0(\sigma)$ and $L_f(\sigma)$ be the projections of the behavior σ to the unprivileged memory locations of the system, at time zero, and at the final time (when execution is done). We say that a component M meets the non-interference hyperproperty when

$$\forall \sigma, \sigma' \in M.\ L_0(\sigma) = L_0(\sigma') \Rightarrow L_f(\sigma) = L_f(\sigma'),$$

i.e., if two traces begin with the unprivileged locations in the same state, the final state of the unprivileged locations matches.

Non-interference is a downward-closed hyperproperty [22, 25], and a 2-safety hyperproperty—hyperproperties called *k-safety* are those for the refutation of which one must provide at least k traces. In our example, to refute the hyperproperty, it suffices to show two traces that share the same unprivileged initial state, but which differ in the unprivileged final state.

As an example, consider the digital system shown in Figure 1; this system is similar to those presented in [22, 25] to illustrate non-interference. Here we have a secret data input S and an n-bit public input P. The system has an output O. There is also an input H which is equal to zero when the system is being accessed by a user with low-privileges, i.e., a user who cannot use the secret data, and equal to one otherwise. The high level hypercontract states that for all environments with $H = 0$, the implementations can only make the output O depend on P, the public data. We thus write:

$$C \;=\; \big(\langle H{=}0\rangle\,,\ \exists f{\in}(2^n \to 2).\, O{=}f(P)\big) \;=\; \big(\langle \neg H\rangle\,,\ \langle O{=}f(P)\rangle_{f\in 2^n\to 2}\big).$$

First we are saying that O is a function of P for some function $2^n \to 2$. In the second equality, we use the notation for k-conic compsets, saying that the implementations can evaluate any of the possible 2^{2^n} functions from n bits to 1 bit. This means that the hyperproperty corresponding to the implementations is 2^{2^n}-conic.

Now suppose we have specifications for components that implement a function $f^* : 2^n \to 2$. One implements it when $S = 1$, and the other when $S = 0$. We will let s be the proposition $S = 1$. Letting complementation have the highest precedence, followed by the meet, and followed by the join, we have

$$C_1 = (\langle \mathcal{B} \rangle\,,\ \langle s \wedge (O_1{=}f^*(P)) \vee \neg s \rangle) \text{ and } C_2 = (\langle \mathcal{B} \rangle\,,\ \langle \neg s \wedge (O_2{=}f^*(P)) \vee s \rangle)\,.$$

Fig. 1. A digital system with a secret input S and a public input P. The overall system must meet the requirement that the secure input does not affect the value of the output O when the signal H is deasserted (this signal is asserted when a privileged user uses the system). Two components with hypercontracts C_1 and C_2 are are available and possesses information-flow properties of their own. Their composite is C_c. Through the quotient C_q, we discover the functionality that needs to be added in order for the design to meet the top-level information-flow spec C_q.

The environments and implementations are both 1-conic. We evaluate $C_c = C_1 \parallel C_2$:

$$C_c = (\langle \mathcal{B} \rangle\,,\ \langle s \wedge (O_1{=}f^*(P)) \vee \neg s \wedge (O_2{=}f^*(P)) \rangle) = (\mathcal{E}_c\,,\ \mathcal{I}_c).$$

Now we compute the quotient C/C_c:

$$C/C_c = \left(\langle H = 0 \rangle \wedge \mathcal{I}_c\,,\ \left\langle \frac{O{=}f(P)}{s \wedge (O_1{=}f^*(P)) \vee \neg s \wedge (O_2{=}f^*(P))} \right\rangle_{f \in 2^n \to 2} \right).$$

We can refine the quotient by allowing the environments to be everything, and picking from the implementations the term with $f = f^*$. Thus we obtain the hypercontract

$$\left(\langle \mathcal{B} \rangle\,,\ \left\langle \frac{O{=}f^*(P)}{s \wedge (O_1{=}f^*(P)) \vee \neg s \wedge (O_2{=}f^*(P))} \right\rangle \right).$$

A further refinement of this is the 1-conic hypercontract

$$C_r = (\langle \mathcal{B} \rangle\,,\ \langle s \wedge (O{=}O_1) \vee \neg s \wedge (O{=}O_2) \rangle)\ .$$

By the properties of the quotient, composing this hypercontract, which knows nothing about f^*, with C_c will yield a hypercontract which meets the non-interference hypercontract C. Note that this hypercontract is consistent, i.e., it has implementations (in general, refining may lead to inconsistency).

4.4 Application: Robustness of Machine Learning systems

In recent years, systems developed for an increasing number of verticals comprise Machine Learning (ML) components. This has spurred an interest in proving properties about these components, particularly since some of these systems are used in safety-critical applications. Yet, even stating what to prove comes with challenges, as ML methodologies are employed when we lack in advance a known mathematical relation between an input and an output space, whereas the statement of a formal property usually requires us to state the behavior that we wish the system to have.

It has been shown that existing ML components can be brittle, in the sense that small changes to inputs can produce large changes in their outputs [31]. Thus, we are often interested in showing that ML systems are robust: if the inputs to the ML component are sufficiently close, the outputs should be sufficiently close [30]. Suppose $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces where the ML component operates. In a supervised setting, we start with a finite set of samples $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ for all i. Using this set, a learning algorithm generates a function $f : \mathcal{X} \to \mathcal{Y}$. If $d_\mathcal{X}$ and $d_\mathcal{Y}$ are distance metrics on the input and output spaces, for a given $x \in \mathcal{X}$, we can ask that the function be robust against small variations around x through the hyperproperty $\forall x' \in \mathcal{X}.\ d_\mathcal{X}(x, x') < \delta \implies d_\mathcal{Y}(f(x), f(x')) < \varepsilon$ for some nonzero δ and ϵ. In the case of a

classification task, where $\mathcal{Y}$ is finite, we can state that inputs which are close in some metric to the samples corresponding to a known class are mapped to that class.

Consider a vehicle approaching an intersection with a stop sign. We wish the vehicle to respect the hypercontract C = (all driving conditions, if there is a stop sign then stop vehicle within a distance D from the stop sign). The implementations say that the vehicle must do something when an object is present in the physical world. An ideal perception hypercontract would be something like C_p = (all driving conditions, say there is a stop sign if and only if there is a stop sign in the real world). Now we can let $\mathcal{X}$ be the space of all possible images one can encounter while driving. Then we can say that $\mathcal{S} \subseteq \mathcal{X}$ is the subset of images containing a stop sign. Currently we have no means for characterizing $\mathcal{S}$, but we construct, for example, an approximation $\hat{\mathcal{S}}$ containing all images which are close in some metric to images known to have stop signs. Suppose we partition the input space in $n+1$ regions $\hat{S}, \mathcal{X}_1, \ldots, \mathcal{X}_n$, where $\mathcal{X}_i$ is smaller than a $d_{\mathcal{X}}$-ball of radius δ. If we train a classifier f to recognize K different categories, we require this classifier to map the values belonging to a set in the partition to one specific class. Since the values of $\hat{\mathcal{S}}$ must map to the stop sign, the only choice is for the rest of the elements of the partition. This means that there are K^n possible classifiers. Call $\mathcal{F}$ the set of all K^n possible classifiers. If we call x the image input provided by the environment and y its assigned class, the perception hypercontract can become

$$C_p = (\langle \text{all possible images} \rangle, \langle y = f(x) \rangle_{f \in \mathcal{F}}).$$

This hypercontract has a 1-conic environment set and a K^n-conic implementation set. However, the perception hypercontract, likely implemented by a neural network, is still too removed from the description of the high-level hypercontract C. We can bridge the gap by writing the high-level hypercontract as

$$C = (\langle x \in \mathcal{X} \rangle, x \in \hat{\mathcal{S}} \Rightarrow \text{stop vehicle within a distance } D \text{ from the stop sign}).$$

Now both the system spec and the perception spec are given at the same level of abstraction. Thus, using the quotient we can focus on what the perception specification does not capture, namely, the control aspects of the design.

5 RECEPTIVE LANGUAGES AND INTERFACE HYPERCONTRACTS

In this section we connect the notion of a hypercontract with specifications expressed as interface automata [9]. With interface theories, we bring in the notion of input-output profiles as an extra typing for components—so far, this was not considered in our development. This effectively partitions $\mathbb{M}$ into sets containing components sharing the same profile.

Our theory of components is constructed from a new notion called *receptive languages*. These objects can be understood as the trace denotations of receptive I/O automata [21]. We will consider downward-closed, 1-conic compsets, see Section 4.2. And interface hypercontracts will be pairs of these with a very specific structure. At the end of the section we show how the denotation of interface automata is captured by interface hypercontracts. One novelty of our approach is that the computation of the composition of hypercontracts, which matches that of interface automata (as we will see), is inherited from our general theory by specializing the component and compset operations.

5.1 The components are receptive languages

Fix once and for all an alphabet Σ. When we operate on words of Σ^*, we will use $\circ$ for word concatenation, and we'll let $\mathrm{Pre}(w)$ be the set of prefixes of a word w. These operations are extended to languages: $L \circ L' = \{w \circ w' \mid w \in L \text{ and } w' \in L'\}$, and $\mathrm{Pre}(L) = \bigcup_{w \in L} \mathrm{Pre}(w)$. An

input-output signature of Σ (or simply an io signature when the alphabet is understood), denoted (I, O), is a partition of Σ in sets I and O, i.e., I and O are disjoint sets whose union is Σ.

Definition 5.1. Let (I, O) be an io signature. A language L of Σ is an I-receptive language if

- L is prefix-closed; and
- if $w \in L$ and $w' \in I^*$ then $w \circ w' \in L$.

The set of all I-receptive languages is denoted $\mathcal{L}_I$.

Proposition 5.2. *Let (I, O) be an io signature. Then $\mathcal{L}_I$ is closed under intersection and union.*

Under the subset order, $\mathcal{L}_I$ is a lattice with intersection as the meet and union as the join. Further, the smallest and largest elements of $\mathcal{L}_I$ are, respectively, $0 = I^*$ and $1 = \Sigma^*$. It so happens that $\mathcal{L}_I$ is a Heyting algebra. To prove this, it remains to be shown that it has exponentiation (i.e., that the meet has a right adjoint).

Proposition 5.3. *Let $L, L' \in \mathcal{L}_I$. The object $L' \to L = \{w \in \Sigma^* \mid Pre(w) \cap L' \subseteq L\}$ is an element of $\mathcal{L}_I$ and satisfies the property (2) of the exponential.*

We further explore the structure of the exponential. To do this, it will be useful to define the following set: for languages L, L' and a set $\Gamma \subseteq \Sigma$, we define the set of *missing Γ-extensions of L' with respect to L* as

$$\mathrm{MissExt}(L, L', \Gamma) = (((L \cap L') \circ \Gamma) \setminus L') \circ \Sigma^* .$$

The elements of this set are all words of the form $w \circ \sigma \circ w'$, where $w \in L \cap L'$, $\sigma \in \Gamma$, and $w' \in \Sigma^*$. These words satisfy the condition $w \circ \sigma \notin L'$. In other words, we find the words of $L \cap L'$ which, when extended by a symbol of Γ, leave the language L', and extend these words by the symbols that make them leave L' and then by every possible word of Σ^*.

Proposition 5.4. *Let $L, L' \in \mathcal{L}_I$. The exponential is given by $L' \to L = L \cup MissExt(L, L', O)$.*

At this point, it has been established that each $\mathcal{L}_I$ is a Heyting algebra. Now we move to composition and quotient, which involve languages of different io signatures.

5.2 Composition and quotient of receptive languages

To every $I \subseteq \Sigma$, we have associated the set of languages $\mathcal{L}_I$. Suppose $I' \subseteq I$. Then $L \in \mathcal{L}_I$ if it is prefix-closed, and the extension of any word of L by any word of I^* remains in L. But since $I' \subseteq I$, this means that the extension of any word of L by any word of $(I')^*$ remains in L, so $L \in \mathcal{L}_{I'}$. We have shown that $I \subseteq I' \Rightarrow \mathcal{L}_{I'} \leq \mathcal{L}_I$. Thus, the map $I \mapsto \mathcal{L}_I$ is a contravariant functor $2^\Sigma \to 2^{2^{\Sigma^*}}$.

Since $I' \subseteq I$ implies that $\mathcal{L}_I \leq \mathcal{L}_{I'}$, we define the embedding $\iota : \mathcal{L}_I \to \mathcal{L}_{I'}$ which maps a language of $\mathcal{L}_I$ to the same language, but interpreted as an element of $\mathcal{L}_{I'}$,

Let (I, O) and (I', O') be io signatures of Σ, $L \in \mathcal{L}_I$, and $L' \in \mathcal{L}_{I'}$. The composition of structures with labeled inputs and outputs traditionally requires that objects to be composed can't share outputs. We say that io signatures (I, O) and (I', O') are *compatible* when $O \cap O' = \emptyset$. This is equivalent to requiring that $I \cup I' = \Sigma$. Moreover, the object generated by the composition should have as outputs the union of the outputs of the objects being composed. This reasoning leads us to the definition of composition:

Definition 5.5 (composition). Let (I, O) and (I', O') be compatible io signatures of Σ. Let $L \in \mathcal{L}_I$ and $L' \in \mathcal{L}_{I'}$. The operation of language *composition*, $\times : \mathcal{L}_I, \mathcal{L}_{I'} \to \mathcal{L}_{I \cap I'}$, is given by

$$L \times L' = \iota L \wedge \iota' L',$$

for the embeddings $\iota : \mathcal{L}_I \to \mathcal{L}_{I \cap I'}$ and $\iota' : \mathcal{L}_{I'} \to \mathcal{L}_{I \cap I'}$.

The adjoint of this operation is the quotient. We will investigate when the quotient is defined. Let $I, I' \subseteq \Sigma$ with $I \subseteq I'$, $L \in \mathcal{L}_I$, and $L' \in \mathcal{L}_{I'}$. Suppose there is $I_r \subseteq \Sigma$ such that the composition rule $\times : \mathcal{L}_{I'}, \mathcal{L}_{I_r} \to \mathcal{L}_I$ is defined. This means that $I' \cup I_r = \Sigma$ and $I' \cap I_r = I$. Solving yields $I_r = I \cup \neg I' = I \cup O'$.

Observe that the smallest element of $\mathcal{L}_{I_r}$ is I_r^*. Thus, the existence of a language $L'' \in \mathcal{L}_{I_r}$ such that $L'' \times L' \leq L$ requires that $L' \cap I_r^* \subseteq L$. Clearly, not every pair L, L' satisfies this property since we can take, for example, $L = I^*$ and $L' = \Sigma^*$ to obtain $L' \cap I_r^* = (I \cup O')^* \not\subseteq I^*$, provided $I' \neq \Sigma$.

We proceed to obtain a closed-form expression for the quotient, but first we define a new operator. For languages L, L' and sets $\Gamma, \Delta \subseteq \Sigma$, the following set of (L', Γ, Δ)-*uncontrollable extensions of* $L \cap L'$:

$$\mathrm{Unc}(L, L', \Gamma, \Delta) = \left\{ w \in L \cap L' \;\middle|\; \begin{array}{l} \exists w' \in (\Gamma \cup \Delta)^* \;\wedge\; \sigma \in \Gamma. \\ w \circ w' \in L \cap L' \;\wedge\; w \circ w' \circ \sigma \in L' \setminus L \end{array} \right\} \circ \Sigma^*. \tag{14}$$

contains: (i) all words of $L \cap L'$ which can be uncontrollably extended to a word of $L' \setminus L$ by appending a word of $(\Gamma \cup \Delta)^*$ and a symbol of Γ, and (ii) all suffixes of such words. Equivalently, $\mathrm{Unc}(L, L', \Gamma, \Delta)$ contains all extensions of the words $w \in L \cap L'$ such that there are extensions of w by words $w' \in (\Gamma \cup \Delta)^*$ that land in L' but not in L after appending to the extensions $w \circ w'$ a symbol of Γ.

Proposition 5.6. *Let (I, O) and (I', O') be io signatures of Σ such that $I \subseteq I'$. Let $L \in \mathcal{L}_I$ and $L' \in \mathcal{L}_{I'}$. Let $I_r = I \cup O'$, and assume that $L' \cap I_r^* \subseteq L$. Then the largest $L'' \in \mathcal{L}_{I_r}$ such that $L'' \times L' \leq L$ is denoted L/L' and is given by*

$$L/L' = (L \cap L' \cup \mathit{MissExt}(L, L', O')) \setminus \mathit{Unc}(L, L', O', I).$$

We have defined receptive languages together with a preorder and a composition operation with its adjoint. These objects will constitute our theory of components, i.e., $\mathbb{M} = \oplus_{I \in 2^\Sigma} \mathcal{L}_I$.

5.3 Compsets and interface hypercontracts

Using the set of components just defined, we proceed to build compsets and hypercontracts. The compsets contain components adhering to the same io signature. Thus, again the notion of an io signature will partition the set of compsets (and the same will happen with hypercontracts). This means that for every compset H, there will always be an $I \subseteq \Sigma$ such that $H \subseteq \mathcal{L}_I$.

For $I \subseteq \Sigma$, and $L \in \mathcal{L}_I$, we will consider compsets of the form

$$\left\{ M \in 2^L \;\middle|\; I^* \subseteq M \right\}, \text{ denoted by } [0, L], \text{ where } 0 \text{ is } I^*, \text{ the smallest element of } \mathcal{L}_I,$$

i.e., compsets are all I-receptive languages smaller than L. We will focus on hypercontracts whose implementations have signature (I, O) and whose environments have (O, I). Thus, hypercontracts will consist of pairs $\mathcal{C} = (\mathcal{E}, \mathcal{S})$ of O- and $\emptyset$-receptive compsets, respectively. We will let

$$\mathcal{S} = [0, S] = \{ M \in \mathcal{L}_\emptyset \mid M \subseteq S \}$$

for some $S \in \mathcal{L}_\emptyset$. We will restrict the environments $E \in \mathcal{E}$ to those that never extend a word of S by an input symbol that S does not accept. The largest such environment is given by

$$E_S = S \cup \mathrm{MissExt}(S, S, O). \tag{15}$$

Since S is prefix-closed, so is E_S. Moreover, observe that E_S adds to S all those strings that are obtained by continuations of words of S by an output symbol that S does not produce. This makes E_S O-receptive. The set of environments is thus $\mathcal{E} = [O^*, E_S]$.

Having obtained the largest environment, we can find the implementations. These are given by $\mathcal{I} = [I^*, M_S]$ for $M_S = S/E_S$. Plugging the definition, we have

$$M_S = (S \cap E_S \cup \text{MissExt}(S, E_S, I)) \setminus \text{Unc}(S, E_S, I, \emptyset).$$

There is no word of I^* which can extend a word of S into $E_S \setminus S$. Thus,

$$M_S = S \cup \text{MissExt}(S, S, I).$$

Observe that S and $\text{MissExt}(S, S, I)$ are disjoint (same for $\text{MissExt}(S, S, O)$). Thus, $E_S \times M_S = S$. In summary, we observe that our hypercontracts are highly structured. They are in 1-1 correspondence with a language $S \in \mathcal{L}_\emptyset$ and an input alphabet $I \subseteq \Sigma$, i.e., there is a set isomorphism

$$\mathcal{L}_\emptyset, 2^\Sigma \xrightarrow{\sim} \mathbf{Contr}\,. \tag{16}$$

Indeed, given S and I, we build E_S by extending S by $\Sigma \setminus I = O$, and M_S by extending S by I. After this, the hypercontract has environments, closed systems, and implementations $[O^*, E_S]$, $[\emptyset, S]$, and $[I^*, M_S]$, respectively.

5.4 Hypercontract composition

Let $S, S' \in \mathcal{L}_\emptyset$. We consider the composition of the interface hypercontracts

$$C_R = C_S \parallel C_{S'}, \text{ where } C_S = ([0, E_S], [0, S])\,,\ C_{S'} = ([0, E_{S'}], [0, S'])\,,$$

and E_S and $E_{S'}$ have signatures (O, I) and (O', I'), respectively. From the structure of interface hypercontracts, we have the relations

$$E_S = S \cup \text{MissExt}(S, S, O) \text{ and } E_{S'} = S' \cup \text{MissExt}(S', S', O').$$

Moreover, the implementations of C, C' are, respectively, $\mathcal{I} = [I^*, M_S]$ and $\mathcal{I}' = [I'^*, M_{S'}]$, where

$$M_S = S \cup \text{MissExt}(S, S, I) \text{ and } M_{S'} = S' \cup \text{MissExt}(S', S', I').$$

The composition of these hypercontracts is defined if (I, O) and (I', O') have compatible signatures. Suppose $C_R = C_S \parallel C_{S'} = ([0, E_R], [0, R])$ for some $R \in \mathcal{L}_\emptyset$. Then the environments must have signature $(O \cup O', I \cap I')$, and the implementations $(I \cap I', O \cup O')$.

Finally, as usual, $E_R = R \cup \text{MissExt}(R, R, O \cup O')$ and $M_R = R \cup \text{MissExt}(R, R, I \cap I')$ are the maximal environment and implementation. R is determined as follows:

Proposition 5.7. *Let C_S and $C_{S'}$ be interface hypercontracts and let $C_R \overset{def}{=} C_S \parallel C_{S'}$. Then R is given by the expression $R = (S \cap S') \setminus [Unc(S', S, O, O') \cup Unc(S, S', O', O)]$.*

The quotient for interface hypercontracts follows from Proposition 3.3.

5.5 Connection with interface automata

Now we explore the relation of interface hypercontracts with interface automata. Let (I, O) be an io signature. An I-interface automaton [9] is a tuple $A = (Q, q_0, \rightarrow)$, where Q is a finite set whose elements we call states, $q_0 \in Q$ is the initial state, and $\rightarrow \subseteq Q \times \Sigma \times Q$ is a deterministic transition relation (there is at most one next state for every symbol of Σ). We let $\mathcal{A}_I$ be the class of I-interface automata, and $\mathcal{A} = \oplus_{I \in 2^\Sigma} \mathcal{A}_I$. In the language of interface automata, input and output symbols are referred to as *actions*.

Given two interface automata (IA) $A_i = (Q_i, q_{i,0}, \rightarrow_i) \in \mathcal{A}_I$ for $i \in \{1, 2\}$, we say that the state $q_1 \in Q_1$ refines $q_2 \in Q_2$, written $q_1 \preceq q_2$, if

- $\forall \sigma \in O, q_1' \in Q_1.\ q_1 \xrightarrow{\sigma}_1 q_1' \Rightarrow \exists q_2' \in Q_2.\ q_2 \xrightarrow{\sigma}_2 q_2'$ and $q_1' \preceq q_2'$ and
- $\forall \sigma \in I, q_2' \in Q_2.\ q_2 \xrightarrow{\sigma}_2 q_2' \Rightarrow \exists q_1' \in Q_1.\ q_1 \xrightarrow{\sigma}_1 q_1'$ and $q_1' \preceq q_2'$.

We say that A_1 refines A_2, written $A_1 \preceq A_2$, if $q_{1,0} \preceq q_{2,0}$. This defines a preorder in $\mathcal{A}_I$.

5.5.1 Mapping to interface hypercontracts. Suppose $A = (Q, q_0, \to) \in \mathcal{A}_I$. We define the language of A, denoted $\ell(A)$, as the set of words obtained by "playing out" the transition relation, i.e., $\ell(A) = \left\{\sigma_0\sigma_1 \dots \sigma_n \;\middle|\; \exists q_1, \dots, q_{n-1}.\ q_i \xrightarrow{\sigma_i} q_{i+1} \text{ for } 0 \le i < n\right\}$. Since $\ell(A)$ is prefix-closed, it is an element of $\mathcal{L}_\emptyset$.

From Section 5.3, we know that interface hypercontracts are isomorphic to a language S of $\mathcal{L}_\emptyset$ and an io signature I. The operation $A \mapsto \ell(A)$ maps an I-receptive interface automaton A to a language of $\mathcal{L}_\emptyset$. Composing this map with the map (16) discussed in Section 5.3, we have maps $\mathcal{A} \to \mathcal{L}_\emptyset, 2^\Sigma \xrightarrow{\sim} \mathbf{Contr}$.

Thus, the interface hypercontract associated to $A \in \mathcal{A}_I$ is $C_A = \left([0, E_{\ell(A)}], [0, \ell(A)]\right)$, where $E_{\ell(A)} \in \mathcal{L}_O$ is given by (15). The following result tells us that refinement of interface automata is equivalent to refinement of their associated hypercontracts.

Proposition 5.8. *Let $A_1, A_2 \in \mathcal{A}_I$. Then $A_1 \preceq A_2$ if and only if $C_{A_1} \preceq C_{A_2}$.*

5.5.2 Composition. Let $A_1 = (Q_1, q_{1,0}, \to_1) \in \mathcal{A}_{I_1}$ and $A_2 = (Q_2, q_{2,0}, \to_2) \in \mathcal{A}_{I_2}$. The composition of the two IA is defined if $I_1 \cup I_2 = \Sigma$. In that case, the resulting IA, $A_1 \parallel A_2$, has io signature $(I_1 \cap I_2, O_1 \cup O_2)$. The elements of the composite IA are $(Q, (q_{1,0}, q_{2,0}), \to_c)$, where the set of states and the transition relation are obtained through the following algorithm:

- Initialize $Q \coloneqq Q_1 \times Q_2$. For every $\sigma \in \Sigma$, $(q_1, q_2) \xrightarrow{\sigma}_c (q_1', q_2')$ if $q_1 \xrightarrow{\sigma}_1 q_1'$ and $q_2 \xrightarrow{\sigma}_2 q_2'$.
- Initialize the set of invalid states to those states where one interface automaton can generate an output action which the other interface automaton does not accept:

$$N \coloneqq \left\{ (q_1, q_2) \in Q_1 \times Q_2 \;\middle|\; \begin{array}{l} \exists\, q_2' \in Q_2, \sigma \in O_2\ \forall q_1' \in Q_1.\ q_2 \xrightarrow{\sigma}_2 q_2' \wedge \neg\left(q_1 \xrightarrow{\sigma}_1 q_1'\right) \text{ or} \\ \exists\, q_1' \in Q_1, \sigma \in O_1\ \forall q_2' \in Q_2.\ q_1 \xrightarrow{\sigma}_1 q_1' \wedge \neg\left(q_2 \xrightarrow{\sigma}_2 q_2'\right) \end{array} \right\}.$$

- Also deem invalid a state such that an output action of one of the interface automata makes a transition to an invalid state, i.e., iterate the following rule until convergence:

$$N \coloneqq N \cup \left\{ (q_1, q_2) \in Q_1 \times Q_2 \;\middle|\; \exists\, (q_1', q_2') \in N, \sigma \in O_1 \cup O_2.\ (q_1, q_2) \xrightarrow{\sigma}_c (q_1', q_2') \right\}.$$

- Now remove the invalid states from the IA:

$$Q \coloneqq Q \setminus N \text{ and } \to_c \coloneqq \to_c \setminus \{(q, \sigma, q') \in \to_c \mid q \in N \text{ or } q' \in N\}.$$

It turns out that composing IA is equivalent to composing their associated hypercontracts:

Proposition 5.9. *Let $A_1, A_2 \in \mathcal{A}_I$. Then $C_{A_1 \| A_2} = C_{A_1} \parallel C_{A_2}$.*

Propositions 5.8 and 5.9 express that our model of interface hypercontracts is equivalent to Interface Automata. We observe that the definition for the parallel composition of interface hypercontracts is straightforward, unlike for the Interface Automata (the latter involves the iterative pruning of invalid states). In fact, in our case this pruning is hidden behind the formula (14) defining the set Unc().

6 CONCLUSIONS

We proposed hypercontracts, a generic model of contracts providing a richer algebra than the metatheory of [5]. We started from a generic model of components equipped with a simulation preorder and a parallel composition. On top of them we considered compsets (or hyperproperties), which are lattices of sets of components equipped with parallel composition and quotient; compsets

are our generic model formalizing "properties." Hypercontracts are then defined as pairs of compsets specifying the allowed environments and either the obligations of the closed system or the set of allowed implementations—both forms are useful.

We specialized hypercontracts by restricting them to pairs of downward closed compsets (where downward closed refers to the component preorder), and then to conic hypercontracts, whose environments and closed systems are described by a finite number of components. Conic hypercontracts possess Assume/Guarantee contracts as a specialization. Specializing them in a different direction provided us with a compact and elegant model of interface hypercontracts, which are conic hypercontracts built on top of input/output components specified as receptive languages. We showed that interface hypercontracts coincide with interface automata; however, our formulas for the parallel composition are direct and do not need the iterative procedure of state pruning, needed in interface automata. We illustrated the versatility of our model on the definition of contracts for information flow in security.

The flexibility and power of our model suggests that a number of directions that were opened in [5], but not explored to their end, can now be re-investigated with much better tools: contracts and testing, subcontract synthesis (for requirement engineering), contracts and abstract interpretation, contracts in physical system modeling.[1] Furthermore, contracts were also developed in the neighbor community of control, which motivates us to establish further links. In particular, Saoud et al. [28, 29] proposed a framework of Assume/Guarantee contracts for input/output discrete or continuous time systems. Assumptions vs. Guarantees are properties stated on inputs vs. outputs; with this restriction, reactive contracts are considered and an elegant formula is proposed for the parallel composition of contracts.

REFERENCES

[1] Martín Abadi and Leslie Lamport. 1993. Composing Specifications. *ACM Trans. Program. Lang. Syst.* 15, 1 (Jan. 1993), 73–132. https://doi.org/10.1145/151646.151649

[2] Sebastian S. Bauer, Alexandre David, Rolf Hennicker, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wąsowski. 2012. Moving from Specifications to Contracts in Component-Based Design. In *Fundamental Approaches to Software Engineering*, Juan de Lara and Andrea Zisman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 43–58.

[3] Sebastian S. Bauer, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. 2014. A modal specification theory for components with data. *Sci. Comput. Program.* 83 (2014), 106–128. https://doi.org/10.1016/j.scico.2013.06.003

[4] Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, and Christos Sofronis. 2008. *Multiple Viewpoint Contract-Based Specification and Design.* Springer Berlin Heidelberg, Berlin, Heidelberg, 200–225. https://doi.org/10.1007/978-3-540-92188-2_9

[5] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger, and Kim G. Larsen. 2018. Contracts for System Design. *Foundations and Trends® in Electronic Design Automation* 12, 2-3 (2018), 124–400.

[6] Ferenc Bujtor and Walter Vogler. 2014. Error-Pruning in Interface Automata. In *40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2014).* Nový Smokovec, Slovakia, 162–173. https://doi.org/10.1007/978-3-319-04298-5_15

[7] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210.

[8] Joey W. Coleman and Cliff B. Jones. 2007. A Structural Proof of the Soundness of Rely/guarantee Rules. *J. Log. Comput.* 17, 4 (2007), 807–841. https://doi.org/10.1093/logcom/exm030

[9] Luca de Alfaro and Thomas A. Henzinger. 2001. Interface Automata. In *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Vienna, Austria) *(ESEC/FSE-9).* Association for Computing Machinery, New York, NY, USA, 109–120. https://doi.org/10.1145/503209.503226

[10] Luca de Alfaro and Thomas A. Henzinger. 2001. Interface Theories for Component-Based Design. In *EMSOFT (Lecture Notes in Computer Science, Vol. 2211)*, Thomas A. Henzinger and Christoph M. Kirsch (Eds.). Springer, 148–165.

[1] Simulink and Modelica toolsuites propose requirements toolboxes, in which requirements are physical system properties that can be tested on a given system model, thus providing a limited form of contract. This motivates the development of a richer contract framework helping for requirement engineering in Cyber Physical Systems design.

[11] Laurent Doyen, Thomas A. Henzinger, Barbara Jobstmann, and Tatjana Petrov. 2008. Interface theories with component reuse. In *Proceedings of the 8th ACM & IEEE International conference on Embedded software, EMSOFT'08*. Atlanta, GA, 79–88.

[12] Joseph A. Goguen and José Meseguer. 1982. Security Policies and Security Models. In *1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982*. IEEE Computer Society, Oakland, CA, USA, 11–20. https://doi.org/10.1109/SP.1982.10014

[13] Ian J. Hayes and Cliff B. Jones. 2017. A Guide to Rely/Guarantee Thinking. In *Engineering Trustworthy Software Systems - Third International School, SETSS 2017, Chongqing, China, April 17-22, 2017, Tutorial Lectures (Lecture Notes in Computer Science, Vol. 11174)*, Jonathan P. Bowen, Zhiming Liu, and Zili Zhang (Eds.). Springer, 1–38. https://doi.org/10.1007/978-3-030-02928-9_1

[14] Íñigo X. Íncer Romeo, Leonardo Mangeruca, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. 2020. The Quotient in Preorder Theories. In Proceedings 11th International Symposium on *Games, Automata, Logics, and Formal Verification*, Brussels, Belgium, September 21-22, 2020 *(Electronic Proceedings in Theoretical Computer Science, Vol. 326)*, Jean-Francois Raskin and Davide Bresolin (Eds.). Open Publishing Association, Brussels, Belgium, 216–233. https://doi.org/10.4204/EPTCS.326.14

[15] Cliff B. Jones. 1983. Specification and Design of (Parallel) Programs. In *IFIP Congress*. Paris, France, 321–332.

[16] Cliff B. Jones. 2003. *Wanted: a compositional approach to concurrency*. Springer New York, New York, NY, 5–15. https://doi.org/10.1007/978-0-387-21798-7_1

[17] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. 2006. Interface Input/Output Automata. In *FM*. 82–97.

[18] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. 2007. Modal I/O Automata for Interface and Product Line Theories. In *Programming Languages and Systems, 16th European Symposium on Programming, ESOP'07 (Lecture Notes in Computer Science, Vol. 4421)*. Springer, 64–79.

[19] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. 2007. On Modal Refinement and Consistency. In *Proc. of the 18th International Conference on Concurrency Theory (CONCUR'07)*. Springer, 105–119.

[20] Gerald Lüttgen and Walter Vogler. 2013. Modal Interface Automata. *Logical Methods in Computer Science* 9, 3 (2013). https://doi.org/10.2168/LMCS-9(3:4)2013

[21] Nancy A. Lynch and Mark R. Tuttle. 1989. An introduction to input/output automata. *CWI Quarterly* 2 (1989), 219–246.

[22] Isabella Mastroeni and Michele Pasqua. 2018. Verifying Bounded Subset-Closed Hyperproperties. In *Static Analysis*, Andreas Podelski (Ed.). Springer International Publishing, Cham, 263–283.

[23] Radu Negulescu. 2000. Process Spaces. In *CONCUR 2000 — Concurrency Theory*, Catuscia Palamidessi (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 199–213.

[24] Roberto Passerone, Íñigo X. Íncer Romeo, and Alberto L. Sangiovanni-Vincentelli. 2019. Coherent Extension, Composition, and Merging Operators in Contract Models for System Design. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 86 (Oct. 2019), 23 pages. https://doi.org/10.1145/3358216

[25] Markus N. Rabe. 2016. *A temporal logic approach to information-flow control*. Ph.D. Dissertation. Universität des Saarlandes. https://doi.org/10.22028/D291-26650

[26] Jean-Baptiste Raclet, Eric Badouel, Albert Benveniste, Benoît Caillaud, Axel Legay, and Roberto Passerone. 2009. Modal Interfaces: Unifying Interface Automata and Modal Specifications. In *Proceedings of the Seventh ACM International Conference on Embedded Software* (Grenoble, France) *(EMSOFT '09)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/1629335.1629348

[27] Alberto L. Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. 2012. Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems. *Eur. J. Control* 18, 3 (2012), 217–238. https://doi.org/10.3166/ejc.18.217-238

[28] Adnane Saoud, Antoine Girard, and Laurent Fribourg. 2018. On the Composition of Discrete and Continuous-time Assume-Guarantee Contracts for Invariance. In *16th European Control Conference, ECC, June 12-15, 2018*. IEEE, Limassol, Cyprus, 435–440. https://doi.org/10.23919/ECC.2018.8550622

[29] Adnane Saoud, Antoine Girard, and Laurent Fribourg. 2021. Assume-guarantee contracts for continuous-time systems. (Feb. 2021). https://hal.archives-ouvertes.fr/hal-02196511 working paper or preprint.

[30] Sanjit A. Seshia, Ankush Desai, Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. 2018. Formal Specification for Deep Neural Networks. In *Automated Technology for Verification and Analysis*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer International Publishing, Cham, 20–34.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV]

A PROOFS

A.1 Hypercontract theory

Proof of Proposition 2.1. Let $(P, \leq)$ be a preorder and $a, b, c \in P$. Further, suppose the quotients a/c and b/c are defined, and $a \leq c$. Then $(a/c) \times c \leq a \leq b$. Thus, $a/c \leq b/c$. This shows that the quotient is monotonic in the first argument. Now suppose the quotients c/a and c/b are defined. It follows that, $(c/b) \times a \leq (c/b) \times b \leq c$. This implies that $c/b \leq c/a$. We conclude the quotient is antitone in the second argument. □

Proof of Proposition 3.1. Suppose H and H' are convex compsets. Then $(H \times H') \times (H \times H') = (H \times H) \times (H' \times H') \leq H \times H'$, so their composition is convex. If H and H' are co-convex compsets, $H \times H' \leq (H \times H) \times (H' \times H') = (H \times H') \times (H \times H')$, so their composition is co-convex. The preservation of flatness follows from the preservation of convexity and co-convexity. □

Proof of Proposition 3.2. Suppose H is a compset and $M \models H$. Since component composition is idempotent, $M = M \times M$, so $M \models H \times H$. □

Proof of Proposition 3.3.

$$
\begin{aligned}
C''/C &= \bigvee \{C' \mid C \parallel C' \leq C''\} = \bigvee \left\{ C' = (\mathcal{E}', \mathcal{I}') \;\middle|\; \left[\begin{array}{l} \mathcal{I} \times \mathcal{I}' \leq \mathcal{I}'', \\ \mathcal{E}'' \times \mathcal{I} \leq \mathcal{E}', \text{ and} \\ \mathcal{E}'' \times \mathcal{I}' \leq \mathcal{E} \end{array} \right] \right\} \\
&= \left(\left(\bigvee \left\{ C' = (\mathcal{E}', \mathcal{I}') \;\middle|\; \left[\begin{array}{l} \mathcal{I} \times \mathcal{I}' \leq \mathcal{I}'', \\ \mathcal{E}'' \times \mathcal{I} \leq \mathcal{E}', \text{ and} \\ \mathcal{E}'' \times \mathcal{I}' \leq \mathcal{E} \end{array} \right] \right\} \right)^{-1} \right)^{-1} \\
&\overset{(8)}{=} \left(\bigwedge \left\{ C' = (\mathcal{I}', \mathcal{E}') \;\middle|\; \left[\begin{array}{l} \mathcal{I} \times \mathcal{I}' \leq \mathcal{I}'', \\ \mathcal{E}'' \times \mathcal{I} \leq \mathcal{E}', \text{ and} \\ \mathcal{E}'' \times \mathcal{I}' \leq \mathcal{E} \end{array} \right] \right\} \right)^{-1} \\
&= \left(\bigwedge \left\{ C' = (\mathcal{I}', \mathcal{E}') \;\middle|\; \left[\begin{array}{l} \mathcal{E}'' \times \mathcal{I} \leq \mathcal{E}', \\ \mathcal{I}' \times \mathcal{I} \leq \mathcal{I}'', \text{ and} \\ \mathcal{I}' \times \mathcal{E}'' \leq \mathcal{E} \end{array} \right] \right\} \right)^{-1} = \left((C'')^{-1} \parallel C \right)^{-1}.
\end{aligned}
$$

□

A.2 Receptive languages and hypercontracts

Proof of Proposition 5.2. Suppose $L, L' \in \mathcal{L}_I$. If w is contained in $L \cap L'$, and w_p is a prefix of w, then w is contained in both L and L', and so is w_p, which means intersection is prefix-closed. Moreover, for any $w' \in I^*$, we have $w \circ w' \in L$ and $w \circ w' \in L'$, so $w \circ w' \in L \cap L'$. We conclude that $L \cap L' \in \mathcal{L}_I$.

Similarly, if w is contained in $L \cup L'$, then we may assume that $w \in L$. Any prefix w_p of w is also contained in L, so $w_p \in L \cup L'$, meaning that union is prefix-closed. In addition, for every $w' \in I^*$, we have $w \circ w' \in L$, so $w \circ w' \in L \cup L'$. This means that $L \cup L' \in \mathcal{L}_I$. □

Proof of Proposition 5.3. First we show that $L' \rightarrow L \in \mathcal{L}_I$. Let $w \in L' \rightarrow L$. If w_p is a prefix of w then $\mathrm{Pre}(w_p) \cap L' \subseteq \mathrm{Pre}(w) \cap L' \subseteq L$, so $L' \rightarrow L$ is prefix-closed.

Now suppose $w \in L' \rightarrow L$ and $w \in L$. Then for $w_I \in I^*$, $w \circ w_I \in L$, so $\mathrm{Pre}(w \circ w_I) \subseteq L$. Suppose $w \in L' \rightarrow L$ and $w \notin L$. Let n be the length of w. Since $w \notin L$, $n > 0$ (the empty string is in L). Write $w = \sigma_1 \ldots \sigma_n$ for $\sigma_i \in \Sigma$. Let $k \leq n$ be the largest natural number such that $\sigma_1 \ldots \sigma_k \in L'$ (note that k can be zero). If $k = n$, then $w \in L' \cap \mathrm{Pre}(w) \subseteq L$, which is forbidden by our assumption that $w \notin L$. Thus, $k < n$. Define $w_p = \sigma_1 \ldots \sigma_{k+1}$. Clearly, $w_p \notin L'$. For any $w_\Sigma \in \Sigma^*$, since L' is

prefix-closed, we must have $\mathrm{Pre}(w \circ w_\Sigma) \cap L' = \mathrm{Pre}(w_p) \cap L' = \mathrm{Pre}(w) \cap L' \subseteq L$. We showed that any word of $L' \to L$ extended by a word of I^* remains in $L' \to L$. We conclude that $L' \to L \in \mathcal{L}_I$.

Now we show that $L' \to L$ has the properties of the exponential. Suppose $L'' \in \mathcal{L}_I$ is such that $L' \cap L'' \subseteq L$. Let $w \in L''$. Then $\mathrm{Pre}(w) \cap L' \subseteq L$, which means that $L'' \leq L' \to L$. On the other hand,

$$L' \cap (L' \to L) = L' \cap \{w \in \Sigma^* \mid \mathrm{Pre}(w) \cap L' \subseteq L\} \subseteq L.$$

Thus, any $L'' \leq L' \to L$ satisfies $L'' \cap L' \leq L$. This concludes the proof. □

Proof of Proposition 5.4. From Prop. 5.3, it is clear that $L \subseteq L' \to L$. Suppose $w \in L' \cap L$. Since L and L' are I-receptive, $w \circ \sigma \in L \cap L'$ for $\sigma \in I$. Assume $\sigma \in O$. If $w \circ \sigma \notin L'$, then we can extend $w \circ \sigma$ by any word $w' \in \Sigma^*$, and this will satisfy $\mathrm{Pre}(w \circ \sigma \circ w') \cap L' = \mathrm{Pre}(w) \cap L' \subseteq L$ due to the fact L' is prefix-closed. If $w \circ \sigma \in L' \setminus L$, then $w \notin L' \to L$. Thus, we can express the exponential using the closed-form expression of the proposition. □

Proof of Proposition 5.6. Suppose $w \in L/L'$ and $w \in L \cap L'$. We have not lost generality because $\epsilon \in L \cap L'$. We consider extensions of w by a symbol σ:

a. If $\sigma \in I$, σ is an input symbol for both L' and the quotient.
   i. L is receptive to I, so $w \circ \sigma \in L$;
   ii. L' is receptive to $I \subseteq I'$, so $w \circ \sigma \in L'$; and
   iii. L/L' must contain $w \circ \sigma$ because the quotient is I_r-receptive.
b. If $\sigma \in O \cap I'$, then σ is an output of the quotient, and an input of L'.
   i. L' is I'-receptive, so $w \circ \sigma \in L'$;
   ii. σ is an output symbol for both L and L/L', so none of them is required to contain $w \circ \sigma$; and
   iii. if $w \circ \sigma \in L' \setminus L$, the extension $w \circ \sigma$ cannot be in the quotient. Otherwise, it can.
c. If $\sigma \in O'$, σ is an output for L' and an input for the quotient.
   i. Neither L nor L' are O'-receptive;
   ii. L/L' is O'-receptive, so we must have $w \circ \sigma \in L/L'$; and
   iii. if $w \circ \sigma \in L' \setminus L$, we cannot have $w \circ \sigma \in L/L'$.

Starting with a word w in the quotient, statements a and b allow or disallow extensions of that word to be in the quotient. However, statements c.ii and c.iii impose a requirement on the word w itself, i.e., if c.iii is violated, c.ii implies that w is not in the quotient. Statements a.iii and c.ii impose an obligation on the quotient to accept extensions by symbols of I and O'; and those extensions may lead to a violation of c.iii. Thus, we remove from the quotient all words such that extensions of those words by elements of $I \cup O'$ end up in $L' \setminus L$. The expression of the proposition follows from these considerations. □

Proof of Proposition 5.7. From the principle of hypercontract composition, we must have

$$E_R \leq U \stackrel{\text{def}}{=} (E_{S'}/M_S) \wedge (E_S/M_{S'}) \text{ and} \tag{17}$$

$$L \stackrel{\text{def}}{=} M_{S'} \times M_S \leq M_R. \tag{18}$$

Observe that the quotients $E_{S'}/M_S$ and $E_S/M_{S'}$ both have io signature $O \cup O'$, so the conjunction in (17) is well-defined as an operation of the Heyting algebra $\mathcal{L}_{O \cup O'}$. We study the first element:

$$E_{S'}/M_S = (E_{S'} \cap M_S \cup \mathrm{MissExt}(E_{S'}, M_S, O)) \setminus \mathrm{Unc}(E_{S'}, M_S, O, O').$$

We attempt to simplify the terms. Suppose $w \in E_{S'} \cap (M_S \setminus S)$. Then all extensions of w lie in $M_S \setminus S$. This means that $\mathrm{MissExt}(E_{S'}, M_S, O) = \mathrm{MissExt}(E_{S'}, S, O)$. Moreover, if a word is an element of

$E_{S'} \setminus S'$, all its extensions are in this set, as well (i.e., it is impossible to escape this set by extending words). Thus, $\mathrm{Unc}(E_{S'}, M_S, O, O') = \mathrm{Unc}(S', M_S, O, O')$. We have

$$E_{S'}/M_S = (E_{S'} \cap M_S \cup \mathrm{MissExt}(E_{S'}, S, O)) \setminus \mathrm{Unc}(S', M_S, O, O').$$

Now we can write

$$U = [(E_{S'} \cap M_S \cup \mathrm{MissExt}(E_{S'}, S, O)) \cap (E_S \cap M_{S'} \cup \mathrm{MissExt}(E_S, S', O'))] \setminus [\mathrm{Unc}(S', M_S, O, O') \cup \mathrm{Unc}(S, M_{S'}, O', O)] .$$

Observe that

$$\begin{aligned} E_{S'} \cap M_S \cap \mathrm{MissExt}(E_S, S', O') &= (S' \cup \mathrm{MissExt}(S', S', O')) \cap M_S \cap \mathrm{MissExt}(E_S, S', O') \\ &= M_S \cap \mathrm{MissExt}(E_S, S', O') = M_S \cap \mathrm{MissExt}(S, S', O'). \end{aligned}$$

The last equality comes from the following fact: if a word of $\mathrm{MissExt}(E_S, S', O')$ is obtained by extending a word of $(E_S \setminus S) \cap S'$ by O', the resulting word is still an element of E_S, which means it cannot be an element of M_S because M_S and E_S are disjoint outside of S. Therefore,

$$U = \begin{bmatrix} (S \cap S') \cup \\ (M_S \cap \mathrm{MissExt}(S, S', O')) \cup \\ (M_{S'} \cap \mathrm{MissExt}(S', S, O)) \cup \\ (\mathrm{MissExt}(E_{S'}, S, O) \cap \mathrm{MissExt}(E_S, S', O')) \end{bmatrix} \setminus [\mathrm{Unc}(S', M_S, O, O') \cup \mathrm{Unc}(S, M_{S'}, O', O)] . \tag{19}$$

We can write

$$\begin{aligned} \mathrm{MissExt}(E_{S'}, S, O) \cap \mathrm{MissExt}(E_S, S', O') = \\ \mathrm{MissExt}(E_{S'}, S, O) \cap \mathrm{MissExt}(S, S', O') \cup \\ \mathrm{MissExt}(S', S, O) \cap \mathrm{MissExt}(E_S, S', O') \cup \\ \mathrm{MissExt}(\mathrm{MissExt}(S', S', O'), S, O) \cap \mathrm{MissExt}(\mathrm{MissExt}(S, S, O), S', O'). \end{aligned}$$

Note that $\mathrm{MissExt}(E_{S'}, S, O) \cap \mathrm{MissExt}(S, S', O') = \mathrm{MissExt}(S, S, O) \cap \mathrm{MissExt}(S, S', O')$. Hence

$$\begin{aligned} &(M_S \cap \mathrm{MissExt}(S, S', O')) \cup (\mathrm{MissExt}(E_{S'}, S, O) \cap \mathrm{MissExt}(S, S', O')) \\ &= \mathrm{MissExt}(S, S', O') \cap (M_S \cup \mathrm{MissExt}(E_{S'}, S, O)) \\ &= \mathrm{MissExt}(S, S', O') \cap (M_S \cup \mathrm{MissExt}(S, S, O)) = \mathrm{MissExt}(S, S', O'). \end{aligned}$$

Finally, we observe that the set $\mathrm{MissExt}(\mathrm{MissExt}(S', S', O'), S, O) \cap \mathrm{MissExt}(\mathrm{MissExt}(S, S, O), S', O')$ must be empty since the words of the first term have prefixes in $S \setminus S'$, and the second in $S' \setminus S$. These considerations allow us to conclude that

$$U \leq \begin{bmatrix} (S \cap S') \cup \\ \mathrm{MissExt}(S, S', O') \cup \\ \mathrm{MissExt}(S', S, O). \end{bmatrix} \setminus \left[\mathrm{Unc}(S', M_S, O, O') \cup \mathrm{Unc}(S, M_{S'}, O', O)\right].$$

To simplify the expression a step further, suppose $w \in \mathrm{Unc}(S', M_S, O, O')$ and has a prefix in $S' \cap (M_S \setminus S)$. Then $w \notin S \cap S'$. The words of $\mathrm{MissExt}(S, S', O')$ do not have prefixes in $S' \setminus S$, so $w \notin \mathrm{MissExt}(S, S', O')$. The words of $\mathrm{MissExt}(S', S, O)$ belong to E_S, which is disjoint from M_S outside of S. Thus, $w \notin \mathrm{MissExt}(S', S, O)$.

We just learned that the words of $\mathrm{Unc}(S', M_S, O, O')$ having a prefix in $S' \cap (M_S \setminus S)$ are irrelevant for the inequality above. Now consider a word w of $\mathrm{Unc}(S', M_S, O, O')$ with no prefix in $S' \cap (M_S \setminus S)$. Let w_p be the longest prefix of w which is in $S \cap S'$. There is a word $w' \in (O \cup O')^*$ and a symbol $\sigma \in O$ such that $w_p \circ w' \in S' \cap M_S$ and $w_p \circ w' \circ \sigma \in M_S \setminus S'$. Suppose w' is not the empty string.

Then we can let σ' be the first symbol of w'. Then $w_p \circ \sigma' \in M_S \setminus S$, so $\sigma' \in O'$. But this means that $w \in \mathrm{Unc}(S, S', O', O)$. If w' is empty, $w_p \in S \cap S'$ and $w_p \circ \sigma' \in M_S \setminus S'$. Since $\sigma \in O$, $w_p \circ \sigma \in \cap M_S$ if and only if it belongs to S. Thus, $w_p \circ \sigma' \in S \setminus S'$, which means that $w \in \mathrm{Unc}(S', S, O, O')$. We can thus simplify the upper bound on E_R to

$$U = \begin{bmatrix} (S \cap S') \cup \\ \mathrm{MissExt}(S, S', O') \cup \\ \mathrm{MissExt}(S', S, O) \end{bmatrix} \setminus \left[\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)\right]. \tag{20}$$

Define $\hat{R} \stackrel{\text{def}}{=} (S \cap S') \setminus [\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)]$. We want to show that $U = \hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O')$. Note that we only have to prove that

$$\mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O') = \begin{bmatrix} \mathrm{MissExt}(S, S', O') \cup \\ \mathrm{MissExt}(S', S, O) \end{bmatrix} \setminus \left[\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)\right]. \tag{21}$$

Proof of (21). Suppose $w \in \mathrm{MissExt}(S, S', O') \setminus [\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)]$. Write $w = w_p \circ \sigma \circ w'$, where w_p is the longest prefix of w which lies in $S \cap S'$, $\sigma \in O'$, and $w' \in \Sigma^*$. $w_p \notin [\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)]$ because all its extensions would be in this set if w_p were in this set, and we know that w is not in this set. It follows that $w_p \in \hat{R}$ and since $w_p \circ \sigma \notin \hat{R}$, $w_p \circ \sigma$ and all its extensions are in $\hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O')$. Thus, $w \in \hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O')$

The same argument applies when

$$w \in \mathrm{MissExt}(S', S, O) \setminus \left[\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)\right].$$

We conclude that the right hand side of (21) is a subset of the left hand side.

Now suppose that $w \in \mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O')$ and write $w = w_p \circ \sigma \circ w'$, where w_p is the longest prefix of w contained in $\hat{R}$, $\sigma \in O \cup O'$, and $w' \in \Sigma^*$. From the definition of $\hat{R}$, $w_p \in S \cap S'$. Suppose $w_p \circ \sigma \in S \cap S'$. Then

$$w_p \circ \sigma \in \left[\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)\right],$$

which means that w_p also belongs to this set (because $\sigma \in O \cup O'$). This contradicts the fact that $w_p \in \hat{R}$, so our assumption that $w_p \circ \sigma \in S \cap S'$ is wrong. Then w_p is also the longest prefix of w contained in $S \cap S'$.

Without loss of generality, assume $\sigma \in O$. Suppose $w_p \circ \sigma \notin S$. Then $w \in \mathrm{MissExt}(S', S, O)$. Moreover, since $w_p \in \hat{R}$, $w_p \circ \sigma \notin [\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)]$. Since $w_p \circ \sigma \notin S \cap S'$, we have $w \notin [\mathrm{Unc}(S', S, O, O') \cup \mathrm{Unc}(S, S', O', O)]$. Thus, w is in the right hand set of (21).

Now suppose $w_p \circ \sigma \notin S'$. If $w_p \circ \sigma \in S$, then $w_p \in \mathrm{Unc}(S', S, O, O')$, which contradicts the fact that $w_p \in R$. We must have $w_p \circ \sigma \notin S$, which we already showed implies that w is in the right hand set of (21).

An analogous reasoning applies to $\sigma \in O'$. We conclude that the right hand side of (21) is a subset of the left hand side, and this finishes the proof of their equality. ∎

This result and (17) tell us that $E_R \leq \hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, O \cup O')$. Now we study the constraint (18). We want to show that $\hat{R}$ yields the tightest bound $L \leq \hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, I \cap I')$ which also respects the bound (17).

Proof. Observe that $L = (S' \cup \mathrm{MissExt}(S', S', I')) \cap (S \cup \mathrm{MissExt}(S, S, I))$. First we will show that $L \subseteq \hat{R} \cup \mathrm{MissExt}(\hat{R}, \hat{R}, I \cap I')$. Suppose $w \in L$. Then w belongs to at least one of the sets (1) $S \cap S'$, (2) $S \cap \mathrm{MissExt}(S', S', I')$, (3) $S' \cap \mathrm{MissExt}(S, S, I)$, or (4) $\mathrm{MissExt}(S, S, I) \cap \mathrm{MissExt}(S', S', I')$. We analyze each case:

(1) Suppose $w \in S \cap S'$. If $w \in \hat{R}$, then clearly $w \in \hat{R} \cup \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$. Suppose $w \notin \hat{R}$. Then there is word $w' \in (O \cup O')^*$ and either a symbol $\sigma \in O$ such that $w \circ w' \circ \sigma \in S \setminus S'$ or a symbol $\sigma \in O'$ such that $w \circ w' \circ \sigma \in S' \setminus S$. Write $w = w_p \circ w''$ such that w'' is the longest suffix of w which belongs to $O \cup O'$. It follows that the last symbol of w_p is an element of $I \cap I'$. Since $w \notin \hat{R}$, neither does w_p. This shows that for every word $w_r \circ \sigma_r \in S \cap S'$ such that $w_r \in \hat{R}$ but $w_r \circ \sigma_r \notin \hat{R}$, we must have $\sigma_r \in I \cap I'$.
Let w'_p be the longest prefix of w_p which lies in $\hat{R}$. By assumption, $\hat{R}$ is not empty. If we write $w = w'_p \circ w''$, the first symbol of w'' is in $I \cap I'$. Thus, $w \in \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$.

(2) Observe that $\text{MissExt}(S', S', I') = \text{MissExt}(S', S', I' \cap I) \cup \text{MissExt}(S', S', I' \cap O)$. Moreover, $S \cap \text{MissExt}(S', S', I \cap I') \subseteq \text{MissExt}(S \cap S', S \cap S', I \cap I') \subseteq \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$.
Suppose $w \in S \cap \text{MissExt}(S', S', I' \cap O)$. Then $w \in \text{Unc}(S', S, O, O')$, so $w \notin \hat{R}$. Let w_i be the longest prefix of w which lies in $S \cap S'$. Then $w_i \notin \hat{R}$, either. Let w_p be the longest prefix of w_i which is in $\hat{R}$. Then $w_i \in \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$, and therefore, so does w.

(3) If $w \in S' \cap \text{MissExt}(S, S, I)$, an analogous reasoning applies.

(4) Suppose $w \in \text{MissExt}(S, S, I) \cap \text{MissExt}(S', S', I')$. If w has a prefix in $S' \cap \text{MissExt}(S, S, I)$ or $S \cap \text{MissExt}(S', S', I')$, then the reasoning of the last two points applies, and we have $w \in \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$. Suppose w has no such a prefix, and write $w = w_p \circ w'$, where w_p is the longest prefix of w which lies in $S \cap S'$. Let σ be the first symbol of w'. Then $w_p \circ \sigma \in \text{MissExt}(S, S, I) \cap \text{MissExt}(S', S', I')$, which means that $\sigma \in I \cap I'$. Thus, $w \in \text{MissExt}(S \cap S', S \cap S', I \cap I') \subseteq \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$.

We have shown that $L \subseteq \hat{R} \cup \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$. Now suppose $w \in \hat{R} \cup \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$. If $w \in \hat{R}$ then clearly $w \in S \cap S' \subseteq L$. Suppose $w \in \text{MissExt}(\hat{R}, \hat{R}, I \cap I')$ and let w_r be the longest prefix of w contained in $\hat{R}$ and σ_r the symbol that comes immediately after w_r in w. Clearly $\sigma_r \in I \cap I'$.

If $w_r \circ \sigma_r \in S \setminus S'$, then $w_r \circ \sigma_r$ cannot be an element of $\hat{R}$. If it were, we would have $E_{\hat{R}} \times S \not\subseteq E_{S'}$, violating the bound (17). The same applies when $w_r \circ \sigma_r \in S' \setminus S$.

If $w_r \circ \sigma_r \notin S \cup S'$, then $w \circ \sigma_r \in \text{MissExt}(S', S', I') \cap \text{MissExt}(S, S, I) \subseteq L$.

If $w_r \circ \sigma_r \in S \cap S'$, then $w_r \circ \sigma_r \in \text{Unc}(S', S, O, O') \cup \text{Unc}(S, S', O', O)$, which means that $w_r \circ \sigma_r$ is not allowed to be an element of $\hat{R}$; otherwise, there would be a contradiction of (17). ■

We conclude that $R = \hat{R}$. □

Proof of Proposition 5.8. Suppose that $A_1 \preceq A_2$. We want to show that $M_{\ell(A_1)} \leq M_{\ell(A_2)}$ and $E_{\ell(A_2)} \leq E_{\ell(A_1)}$. We proceed by induction in the length n of words, i.e., we will show that this relations hold for words of arbitrary length.

Consider the case $n = 1$. Suppose $\sigma \in M_{\ell(A_1)} \cap \Sigma$. If $\sigma \in I$, then $\sigma \in M_{\ell(A_2)}$ because of I-receptivity. If $\sigma \in O$, then $\sigma \in \ell(A)$, so there exists $q_1 \in Q_1$ such that $q_{1,0} \xrightarrow{\sigma}_1 q_1$, which means that there exists $q_2 \in Q_2$ such that $q_{2,0} \xrightarrow{\sigma}_2 q_2$. Thus, $\sigma \in \ell(A_2) \subseteq M_{\ell(A_2)}$. We have shown that $M_{\ell(A_1)} \subseteq M_{\ell(A_2)}$ for $n = 1$. An analogous reasoning shows that $E_{\ell(A_2)} \subseteq E_{\ell(A_1)}$

Suppose the statement is true for words of length n. Let $w \circ \sigma \in M_{\ell(A_1)}$, where $w \in \Sigma^*$ is a word of length n, and $\sigma \in \Sigma$. By the inductive assumption, $w \in M_{\ell(A_2)}$.

- If $\sigma \in I$, then $w \circ \sigma \in M_{\ell(A_2)}$ due to I-receptiveness.
- Let $\sigma \in O$ and $w \notin \ell(A_1)$. Then we can write $w = w_p \circ w'$, where w_p is the longest prefix of w which lies in $\ell(A_1)$ (suppose it has length l). Let σ' be the first symbol of w'; clearly $\sigma' \in I$. Since $w \in \ell(A_2)$ and this set is prefix-closed, $w_p \in \ell(A_2)$. Since $w_p \in \ell(A_1) \cap \ell(A_2)$, there exist $\{q_{j,i} \in Q_j\}_{i=1}^k$ (for $j \in \{1, 2\}$) such that $q_{j,i-1} \xrightarrow{w_i}_j q_{j,i}$ for $0 < i \leq k$, where w_i is the i-th symbol of w_p. Since the IA are deterministic, we must have $q_{1,i} \leq q_{2,i}$. Suppose

there were a $q_2 \in Q_2$ such that $q_{2,k} \xrightarrow{\sigma'}_2 q_2$; since $q_{1,k} \leq q_{2,k}$ and $\sigma' \in I$, this would mean that there exists $q_1 \in Q_1$ such that $q_{1,k} \xrightarrow{\sigma'}_1 q_1$, which would mean that $w_p \circ \sigma' \in \ell(A_1)$, a contradiction. We conclude that such q_2 does not exist, which means that $w_p \circ \sigma' \notin \ell(A_2)$, which means that $w \circ \sigma \in M_{\ell(A_2)}$ because of I-receptiveness.

- Finally, if $\sigma \in O$ and $w \in \ell(A_1)$, then there exist $\{q_{1,i} \in Q_1\}_{i=1}^n$ such that $q_{1,i-1} \xrightarrow{w_i}_1 q_{1,i}$ for $0 < i \leq n$, where w_i is the i-th symbol of w. Since $w \circ \sigma \in M_{\ell(A_1)}$, $w \in \ell(A_1)$, and $\sigma \in O$, we must have $w \circ \sigma \in \ell(A_1)$. This means that there must exist $q_{1,n+1} \in Q_1$ such that $q_{1,n} \xrightarrow{\sigma}_1 q_{1,n+1}$. We know that $w \in M_{\ell(A_2)}$ by the induction assumption. If $w \notin \ell(A_2)$, then clearly $w \circ \sigma \in M_{\ell(A_2)}$. If $w \in \ell(A_2)$, there are states $\{q_{2,i} \in Q_2\}_{i=1}^n$ such that $q_{2,i-1} \xrightarrow{w_i}_2 q_{2,i}$ for $0 < i \leq n$. Moreover, there exists $q_{n+1} \in Q_1$ such that $q_n \xrightarrow{\sigma}_1 q_{n+1}$ and $q_{1,n} \leq q_{2,n}$, there must be a $q_{2,n+1} \in Q_2$ such that $q_{2,n} \xrightarrow{\sigma}_2 q_{2,n+1}$, which means that $w \circ \sigma \in M_{\ell(A_2)}$.

We have shown that $M_{\ell(A_1)} \subseteq M_{\ell(A_2)}$. An analogous argument proves that $E_{\ell(A_2)} \subseteq E_{\ell(A_1)}$.

Now suppose that $M_{\ell(A_1)} \subseteq M_{\ell(A_2)}$ and $E_{\ell(A_2)} \subseteq E_{\ell(A_1)}$. We want to show that $q_{1,0} \leq q_{2,0}$. We proceed by coinduction.

Let n be a natural number. Suppose there exist sets $\{q_{j,i} \in Q_j\}_{i=1}^n$ with $j \in \{1,2\}$ such that $q_{1,i} \leq q_{2,i}$ for all i and a word w of length n such that $q_{j,i-1} \xrightarrow{w_i}_j q_{j,i}$ for $0 < i \leq n$. Suppose there exists $q_{1,n+1} \in Q_1$ and $\sigma \in O$ such that $q_{1,n} \xrightarrow{\sigma}_1 q_{1,n+1}$. Then $w \circ \sigma \in M_{\ell(A_1)} \subseteq M_{\ell(A_2)}$. Observe that $w \in \ell(A_2)$, so we must have $w \circ \sigma \in \ell(A_2)$. This means there must be a $q_{2,n+1} \in Q_2$ such that $q_{2,n} \xrightarrow{\sigma}_2 q_{2,n+1}$. We assume that $q_{1,n+1} \leq q_{2,n+1}$. Similarly, suppose there exists $q'_{2,n+1} \in Q_2$ and $\sigma \in I$ such that $q_{2,n} \xrightarrow{\sigma}_2 q'_{2,n+1}$. Then $w \circ \sigma \in E_{\ell(A_2)} \subseteq E_{\ell(A_1)}$. Since $w \in \ell(A_1)$, we must have $w \circ \sigma \in \ell(A_1)$. Thus, there must exist $q'_{1,n+1} \in Q_1$ such that $q_{1,n} \xrightarrow{\sigma}_1 q'_{1,n+1}$. We assume that $q_{1,n+1} \leq q_{2,n+1}$. This finished the coinductive proof. □

Proof of Proposition 5.9. Let A_i have io signatures (I_i, O_i) for $i \in \{1,2\}$. For composition to be defined, we need $I_1 \cup I_2 = \Sigma$. Let C_{A_i} be the interface contract associated with A_i. From Proposition 5.7 and Section 5.3, the composition $C_{A_1} \parallel C_{A_2}$ is isomorphic to $I_1 \cap I_2$ and the $\mathcal{L}_\emptyset$ language $R = (\ell(A_1) \cap \ell(A_2)) \setminus [\mathrm{Unc}(\ell(A_1), \ell(A_2), O_2, O_1) \cup \mathrm{Unc}(\ell(A_2), \ell(A_1), O_1, O_2)]$. From Section 5.5.2, we deduce that $\ell(A_1 \parallel A_2) = R$. The proposition follows. □