

Compositional Contract Abstraction for System Design

Albert Benveniste, Dejan Nickovic, Tom Henzinger

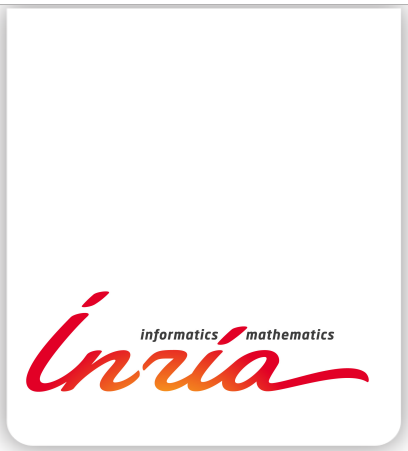
RESEARCH REPORT

N° 8460

January 2014

Project-Teams Hycomes

ISSN 0249-6399 ISRN INRIA/RR--8460--FR+ENG



Compositional Contract Abstraction for System Design

Albert Benveniste*, Dejan Nickovic†, Tom Henzinger‡

Project-Teams Hycomes

Research Report n° 8460 — January 2014 — 13 pages

Abstract: *Contract-based design* has been recently proposed as a framework for concurrent system design in the context of complex supplier chains, where sub-system design can be sub-contracted to suppliers while guaranteeing correct system integration. A unifying *meta-theory of contracts* was proposed in [6], which subsumes known frameworks such as interface theories, modal interfaces, and Assume/Guarantee contracts. This report proposes, for this meta-theory of contracts, a generic abstraction technique allowing to prove contract properties based on their abstractions. More precisely, we show how to lift abstractions, from components to contracts, in a systematic way. In doing so, fundamental relations such as being a correct implementation or a valid environment, refining, can be checked on abstractions. Our abstraction technique is fully compositional with respect to contract conjunction. Compositionality of abstraction with respect to contract composition is only partially achieved. We believe that the results we obtain are the best achievable ones and we explain the obstructions we see against improving them. Our abstraction technique complements observers, proposed as a testing technique adapted to contracts in [6]. The latter allow *disproving* properties, whereas abstraction allows *proving* them.

Key-words: system design, component based design, contract, interface, abstraction, abstract interpretation.

This work was funded in part by the European STREP-COMBEST project number 215543, the European projects CESAR of the ARTEMIS Joint Undertaking.

* INRIA, Rennes, France. corresp. author: Albert.Benveniste@inria.fr
† Austrian Institute of Technology (AIT)
‡ IST Austria, Klosterneuburg

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Abstraction compositionnelle des contrats pour la conception de systèmes

Résumé : La *conception par contrats* a été proposée récemment comme une approche formelle pour la conception de systèmes permettant le développement parallèle de sysèmes dans un contexte de chaine complexe de sous-traitants. Les théories d'interfaces, les interfaces modales et les contrats hypothèse/garantie, sont autant de formalismes en ce sens. L'article collectif [6] a proposé une "méta-théorie" des contrats, unifiant les formalismes précédents. Le présent rapport développe, pour cette méta-théorie des contrats, une technique systématique d'*abstraction*. Les propriétés fondamentales des contrats (relation d'implémentation, d'environnement, de raffinement) peuvent être prouvées sur les abstractions. L'abstraction proposée offre de bonnes propriétés de compositionnalité, même si toutes les propriétés souhaitables ne sont pas valides. Cette technique d'abstraction complète celle des observateurs, qui permettent d'invalider des propriétés de contrats par une approche de type test.

Mots-clés : conception des systèmes, composant, contrat, interface, abstraction, interprétation abstraite.

CONTENTS

I Introduction 3

II Background on Contract Meta-Theory 4
- II-A Components and their composition . . . 4
- II-B Contracts . . . 4
- II-C Refinement and conjunction . . . 4
- II-D Contract composition . . . 5
- II-E Issues of effectiveness . . . 6

III Abstractions for contracts 6
- III-A Background on Galois connections . . . 7
- III-B Lifting abstractions, from components to contracts . . . 7
- III-C Using abstractions for proofs . . . 7
- III-D Compositionality of abstraction . . . 8
- III-E Concluding discussion . . . 9

IV The case of A/G-contracts 9
- IV-A Components specified as sets of behaviors 9
- IV-B Components specified as transition systems . . . 10
- IV-C Using predicate abstraction . . . 11

V Obstructions to getting stronger results 11
- V-A Why not Galois connections for contracts? 11
- V-B Can we expect better results for the special case of A/G-contracts? . . . 12

VI Conclusion 12

References 13

*

I. INTRODUCTION

Handling complexity in the design of critical systems has been an important challenge and numerous techniques have been developed in order to address this problem. While testing remains the most prominent method in checking a system's correctness, it can find bugs but not guarantee their absence. Formal verification provides such guarantees, but is subject to a number of limitations related to the combinatorial state explosion in the analysis of large systems as well as undecidability results for many systems with infinite state spaces.

Contract-based (or interface-based) design [2], [6] is a rigorous framework which addresses the complexity of system development by the exploitation of the component-based structure of systems and the separation of orthogonal viewpoints during the design process. A partial system representation, which may describe either a *component* which is to be composed (disjuncted) with other components, or a *viewpoint* (or *aspect*) which is to be superposed (conjuncted) with other viewpoints, is specified by a *contract* (or *interface*), which consists of two dual parts: an *assumption* constraining the environment of the specified subsystem, and a *guarantee* constraining the subsystem (component or viewpoint) itself, provided the assumption is met. Contract $\mathcal{C}'$ *refines* contract $\mathcal{C}$ if every implementation of $\mathcal{C}'$ can operate in every legal environment for $\mathcal{C}$ and then implements $\mathcal{C}$. Providing a calculus of open systems, contract theories support the incremental component-based and aspect-oriented development of systems. When a contract for the overall system is the composition of subcontracts for different subsystems, then each subcontract can be delegated to a different supplier for independent implementation, because all information about the possible uses of the components is provided by the subcontracts. The conjunction of contracts allows the separate handling of a system's orthogonal but cross-cutting concerns, and the proper fusion of these viewpoints at a later stage of the design process. Contract-based theories were developed for various classes of systems [17], [22], [25], including asynchronous and synchronous systems with and without shared actions or variables [16], [5], [7], [8], [19], real-time and probabilistic systems [15], [18].

Abstract interpretation (AI) is a formal framework proposed by P. Cousot and R. Cousot [10], [11], [12] which addresses the above problems by allowing the systematic simplification of certain classes of systems, making them amenable to formal verification: from undecidable to decidable, or from high to low complexity. At its core, AI offers formal means to travel back and forth between *concrete* (detailed and realistic) and *abstract* (simpler but approximate) representations of a system. Central to this is the Galois connection (α, γ) which relates concrete and abstract domains, such that the *concretization* function γ is the best possible approximation of an inverse for the *abstraction* function α. Based on this concept, AI theory offers powerful techniques to abstract classes of systems defined through fixpoint equations by "widening." A correctness proof of an abstract system representation obtained by applying the AI framework carries over to the concrete system. On the other hand, finding a bug in an abstraction may be a false alarm resulting from oversimplification. In this way AI can be seen as a dual activity to testing. The tool support for AI is well-developed and mature [13], [14], [27] and has been successfully applied in various areas, in particular the static analysis of programs.

To summarize, both AI and contract-based design aim to handle complexity in system development and analysis. While AI maps complex systems to simpler ones that preserve essential properties, contract-based theories facilitate rigorous top-down and bottom-up design by exploiting the compositional structure of systems and enabling the separation of orthogonal viewpoints.

The original agenda of this paper was to lift abstract interpretation from components to contracts, by constructing a canonical way of deriving a Galois connection on contracts from a Galois connection on components. So far we failed achieving this and we discuss in this report the obstructions we found. Still, we were able to derive a very useful canonical lifting $(\alpha, \gamma) \mapsto \overline{\alpha}$, mapping a Galois connection on components to an abstraction on contracts, although with no associated concretization making it a Galois connection. For $\mathcal{C}$ a contract,

proving that a component M is a correct implementation of $\mathcal{C}$ can be achieved by confronting abstraction $\alpha(M)$ to $\overline{\alpha}(\mathcal{C})$, and the same holds regarding correctness of environments. A set of contracts is called consistent (respectively, compatible), if a shared implementation can be found for all of them (respectively, if an environment can be found for the parallel composition of these contracts). Compositionality of our lifting with respect to the conjunction of contracts holds: $\overline{\alpha}(\mathcal{C}_1 \wedge \mathcal{C}_2) = \overline{\alpha}(\mathcal{C}_1) \wedge \overline{\alpha}(\mathcal{C}_2)$. Regarding the parallel composition of contracts, under easily achievable conditions for the Galois connection on components, we have $\overline{\alpha}(\mathcal{C}_1 \otimes \mathcal{C}_2) \preceq \overline{\alpha}(\mathcal{C}_1) \otimes \overline{\alpha}(\mathcal{C}_2)$, which supports compositional abstraction-based proofs of consistency of the parallel composition of contracts, but not the compatibility.

There is little literature on using abstract interpretation in the context of contracts, interfaces, or specifications. Some AI frameworks make abstraction compositional, with extensive studies on shared-variable concurrency [24]. Bauer, Hennicker, and Wirsing [3] develop a (partial) theory of interfaces with AI for dealing with data. Their framework consists of Modal Interfaces [25] in which *may* and *must* transitions are equipped with pre- and post-conditions in the form of predicates involving data, which can be abstracted. Conjunction is handled in later work [4]. None of these papers propose a systematic and generic lifting of AI from systems or components to contracts or interfaces.

The paper is organized as follows. Section II recalls the background on the generic theory of contracts developed in [6], referred to as the *meta-theory of contracts*. The lifting $(\alpha, \gamma) \mapsto \overline{\alpha}$ is presented in Section III, for the meta-theory. Properties of this lifting are presented. We detail in Section IV the instantiation of this lifting for the case of Assume/Guarantee contracts. Finally, we develop in Section V the obstructions we see in improving the results we have.

II. Background on Contract Meta-Theory

We briefly recall here the *meta-theory of contracts* proposed in [6].[1] The reader is referred to [6] for motivations, context, and bibliographical considerations. This meta-theory is summarized in Table I. It comes as a few primitive concepts, on top of which derived concepts can be built. A number of key properties can be proved about the resulting framework. These properties demonstrate that contracts are a convenient paradigm to support incremental development and independent implementability in system design. This meta-theory subsumes known concrete contract theories [6].

A. Components and their composition

We start from a universe $\mathcal{M}$ of possible *components,* each denoted by the symbol M or E, and a universe of their specifications, or *contracts,* each denoted by the symbol $\mathcal{C}$. Our meta-theory does not presume any particular modeling style, neither for components nor for contracts. More generally, some frameworks may represent components and contracts with sets of discrete time or even continuous time traces, other theories use logics, or state-based models of various kinds, and so on.

We assume a *composition* $M_1 \times M_2$ acting on pairs of components. Component composition $\times$ is partially, not totally, defined. Two components such that $M_1 \times M_2$ is well defined are called *composable*. Composability of components is meant to be a typing property. In order to guarantee that different composable components may be assembled together in any order, it is required that component composition $\times$ is associative and commutative. An *environment* for a component M is another component E composable with M.

B. Contracts

Definition 1: We consider a class $\mathbb{C}$ *of* contracts $\mathcal{C}$ *whose semantics is a pair* $[\![\mathcal{C}]\!] = (\mathcal{C}^{\mathsf{env}}, \mathcal{C}^{\mathsf{imp}}) \in 2^{\mathcal{M}} \times 2^{\mathcal{M}}$*, where:*

- $\mathcal{C}^{\mathsf{imp}} \subseteq \mathcal{M}$ *is the set of* implementations *of* $\mathcal{C}$*, and*
- $\mathcal{C}^{\mathsf{env}} \subseteq \mathcal{M}$ *is the set of* environments *of* $\mathcal{C}$*.*
- *For any pair* $(E, M) \in \mathcal{C}^{\mathsf{env}} \times \mathcal{C}^{\mathsf{imp}}$*,* E *is an environment for* M*.*

A contract possessing no implementation is called inconsistent*. A contract possessing no environment is called* incompatible*. Write*

$$M \models^{\mathsf{imp}} \mathcal{C} \quad \textit{and} \quad E \models^{\mathsf{env}} \mathcal{C}$$

to express that $M \in \mathcal{C}^{\mathsf{imp}}$ *and* $E \in \mathcal{C}^{\mathsf{env}}$*, respectively.*

In the meta-theory the class $\mathbb{C}$ is abstract. Each particular contract framework comes with a concrete definition of $\mathbb{C}$ and specifies all the concepts listed in the last column of Table I.

C. Refinement and conjunction

To support independent implementability, the concept of contract refinement must ensure the following: if contract $\mathcal{C}'$ refines contract $\mathcal{C}$, then any implementation of $\mathcal{C}'$ should implement $\mathcal{C}$ and be able to operate in any environment for $\mathcal{C}$. Hence the following definition for refinement preorder $\preceq$ between contracts: $\mathcal{C}'$ refines $\mathcal{C}$, written $\mathcal{C}' \preceq \mathcal{C}$, if and only if $\mathcal{C}'^{\mathsf{imp}} \subseteq \mathcal{C}^{\mathsf{imp}}$ and $\mathcal{C}'^{\mathsf{env}} \supseteq \mathcal{C}^{\mathsf{env}}$. As a direct consequence, the following property holds, which justifies the use of the term "refinement" for this relation:

Property 1 (refinement):

1) *Any implementation of* $\mathcal{C}'$ *is an implementation of* $\mathcal{C}$*:* $M \models^{\mathsf{imp}} \mathcal{C}' \Rightarrow M \models^{\mathsf{imp}} \mathcal{C}$*, and*
2) *Any environment of* $\mathcal{C}$ *is an environment of* $\mathcal{C}'$*:* $E \models^{\mathsf{env}} \mathcal{C} \Rightarrow E \models^{\mathsf{env}} \mathcal{C}'$*.*

At this point we need the following assumption:

Assumption 1: For $\mathbb{C}' \subseteq \mathbb{C}$ *any subset of expressible contracts, Greatest Lower Bound (GLB)* $\bigwedge \mathbb{C}'$ *and Least Upper Bound (LUB)* $\bigvee \mathbb{C}'$ *both exist in* $\mathbb{C}$*, where GLB and LUB refer to refinement order.*

This allows us to define the *conjunction* of contracts $\mathcal{C}_1$ and $\mathcal{C}_2$ as being $\mathcal{C}_1 \wedge \mathcal{C}_2$, the GLB of these two contracts. The intent is to define this conjunction as the intersection of sets of implementations and the union of sets of environments. However,

[1] In software engineering, meta-models are "models of models", i.e., formal ways of specifying a certain family of models. Similarly, we call here *meta-theory* a way to specify a particular family of theories.

Concept	**Definition and generic properties**	**What depends on the particular contract framework**
Primitive		
Component	Components are denoted by M	How components are specified
Composability of components	A type property on pairs of components (M_1, M_2)	How this type property is defined
Composition of components	$M_1 \times M_2$ is well defined if and only if M_1 and M_2 are composable; It is required that $\times$ is associative and commutative	The definition of the composition
Environment	An *environment* for component M is a component E such that $E \times M$ is well defined	
Derived		
Contract	The semantics of *contract* $\mathcal{C}$ is a pair $(\mathcal{C}^{\mathsf{env}}, \mathcal{C}^{\mathsf{imp}})$, where $\mathcal{C}^{\mathsf{imp}}$ is a subset of components and $\mathcal{C}^{\mathsf{env}}$ a subset of valid environments	The class $\mathbb{C}$ of contracts; unless otherwise specified, quantifying is implicitly over $\mathcal{C} \in \mathbb{C}$
Consistency	$\mathcal{C}$ is *consistent* iff it has at least one component: $\mathcal{C}^{\mathsf{imp}} \neq \emptyset$	How consistency is checked
Compatibility	$\mathcal{C}$ is *compatible* iff it has at least one environment: $\mathcal{C}^{\mathsf{env}} \neq \emptyset$	How compatibility is checked
Implementation	$M \models^{\mathsf{imp}} \mathcal{C}$ if and only if $M \in \mathcal{C}^{\mathsf{imp}}$ $E \models^{\mathsf{env}} \mathcal{C}$ if and only if $E \in \mathcal{C}^{\mathsf{env}}$	How implementation is checked
Refinement	$\mathcal{C}' \preceq \mathcal{C}$ iff $\mathcal{C}'^{\mathsf{env}} \supseteq \mathcal{C}^{\mathsf{env}}$ and $\mathcal{C}'^{\mathsf{imp}} \subseteq \mathcal{C}^{\mathsf{imp}}$; Property 1 holds	How refinement is checked
GLB and LUB of contracts	$\mathcal{C}_1 \wedge \mathcal{C}_2 =$ Greatest Lower Bound (GLB) for $\preceq$; $\mathcal{C}_1 \vee \mathcal{C}_2 =$ Least Upper Bound (LUB) for $\preceq$; Assumption 1 is in force and Property 2 holds Say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *consistent* if so is $\mathcal{C}_1 \wedge \mathcal{C}_2$	How GLB and LUB are expressed and computed
Composition of contracts	$\mathcal{C}_1 \otimes \mathcal{C}_2$ is defined if $\left.\begin{array}{r} M_1 \models^{\mathsf{imp}} \mathcal{C}_1 \\ M_2 \models^{\mathsf{imp}} \mathcal{C}_2 \end{array}\right\} \Rightarrow (M_1, M_2)$ composable $\mathcal{C}_1 \otimes \mathcal{C}_2 = \min \left\{ \mathcal{C} \;\middle|\; \left[\begin{array}{r} M_1 \models^{\mathsf{imp}} \mathcal{C}_1 \\ \text{and } M_2 \models^{\mathsf{imp}} \mathcal{C}_2 \\ \text{and } E \models^{\mathsf{env}} \mathcal{C} \end{array}\right] \Rightarrow \left[\begin{array}{r} M_1 \times M_2 \models^{\mathsf{imp}} \mathcal{C} \\ \text{and } E \times M_2 \models^{\mathsf{env}} \mathcal{C}_1 \\ \text{and } E \times M_1 \models^{\mathsf{env}} \mathcal{C}_2 \end{array}\right] \right\}$ Assumption 2 is in force; Properties 3, 4, and 5 hold Say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *compatible* if so is $\mathcal{C}_1 \otimes \mathcal{C}_2$	How composition is expressed and computed

Table I
Summary of the meta-theory of contracts. We first list **primitive** concepts and then **derived** concepts introduced by the meta-theory.

not every pair of sets of components can be the semantics of a contract belonging to class $\mathbb{C}$. The best approximation consists in taking the greatest lower bound for the refinement relation. The following immediate properties hold:

Property 2 (shared refinement):

1) *Any contract that refines $\mathcal{C}_1 \wedge \mathcal{C}_2$ also refines $\mathcal{C}_1$ and $\mathcal{C}_2$. Any implementation of $\mathcal{C}_1 \wedge \mathcal{C}_2$ is a shared implementation of $\mathcal{C}_1$ and $\mathcal{C}_2$. Any environment of $\mathcal{C}_1$ or $\mathcal{C}_2$ is an environment of $\mathcal{C}_1 \wedge \mathcal{C}_2$.*
2) *For $\mathbf{C} \subseteq \mathbb{C}$ a subset of contracts, $\bigwedge \mathbf{C}$ is compatible if and only if there exists a compatible $\mathcal{C} \in \mathbf{C}$.*

The conjunction operation formalizes the intuitive notion of a "set of contracts" or a "set of requirements".

D. Contract composition

On top of component composition, we define a *contract composition* $\mathcal{C}_1 \otimes \mathcal{C}_2$, whose intuition is as follows: composing two implementations of $\mathcal{C}_1$ and $\mathcal{C}_2$ should yield an implementation of $\mathcal{C}_1 \otimes \mathcal{C}_2$ and any environment for $\mathcal{C}_1 \otimes \mathcal{C}_2$, when composed with an implementation for $\mathcal{C}_1$, should yield a valid environment for $\mathcal{C}_2$ and vice-versa. Observe that $E \models^{\mathsf{env}} \mathcal{C}$ implies that E is composable with any implementation of $\mathcal{C}$, and thus $E \times M_i$ are well defined. Formally, $\mathcal{C}_1 \otimes \mathcal{C}_2$ is defined by the formula given in Table I, where "min" refers to refinement order. For this to make sense, we assume the following:

Assumption 2: We assume that the min *in the formula defining $\mathcal{C}_1 \otimes \mathcal{C}_2$ in Table I exists and is unique.*

Equivalently, $\bigwedge \mathbf{C}_{\mathcal{C}_1,\mathcal{C}_2}$ belongs to $\mathbf{C}_{\mathcal{C}_1,\mathcal{C}_2}$, where $\mathbf{C}_{\mathcal{C}_1,\mathcal{C}_2}$ denotes the set of contracts defined by the brackets in this formula. The following lemma will be instrumental:

Lemma 1: Let four contracts be such that $\mathcal{C}'_1 \preceq \mathcal{C}_1$, $\mathcal{C}'_2 \preceq \mathcal{C}_2$, and $\mathcal{C}_1 \otimes \mathcal{C}_2$ is well defined. Then, so is $\mathcal{C}'_1 \otimes \mathcal{C}'_2$ and $\mathbf{C}_{\mathcal{C}'_1,\mathcal{C}'_2} \supseteq \mathbf{C}_{\mathcal{C}_1,\mathcal{C}_2}$.

Proof: Since $\mathcal{C}_1 \otimes \mathcal{C}_2$ is well defined, it follows that every pair (M_1, M_2) of respective implementations of these contracts is a composable pair of components. Hence, $\mathcal{C}'_1 \otimes \mathcal{C}'_2$ is well defined according to the formula of Table I. Next, since $\mathcal{C}'_1 \preceq \mathcal{C}_1$ and $\mathcal{C}'_2 \preceq \mathcal{C}_2$ and using Assumption 2, $M_1 \models^{\mathsf{imp}} \mathcal{C}'_1$ and $M_2 \models^{\mathsf{imp}} \mathcal{C}'_2$ implies $M_1 \models^{\mathsf{imp}} \mathcal{C}_1$ and $M_2 \models^{\mathsf{imp}} \mathcal{C}_2$; similarly $E \times M_2 \models^{\mathsf{env}} \mathcal{C}_1$ and $E \times M_1 \models^{\mathsf{env}} \mathcal{C}_2$ implies $E \times M_2 \models^{\mathsf{env}} \mathcal{C}'_1$ and $E \times M_1 \models^{\mathsf{env}} \mathcal{C}'_2$. Therefore, replacing, in the big brackets defining the contract composition, $\mathcal{C}_1$ by $\mathcal{C}'_1$ and $\mathcal{C}_2$ by $\mathcal{C}'_2$ can only increase the set $\mathbf{C}_{\mathcal{C}_1,\mathcal{C}_2}$. □

To conform to the usage, we say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *compatible* contracts if their composition $\mathcal{C}_1 \otimes \mathcal{C}_2$ is defined and compatible in the sense of Table I. The following properties are a direct corollary of Lemma 1:

Property 3 (independent implementability): For all contracts $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}'_1$ and $\mathcal{C}'_2$, if

1) *$\mathcal{C}_1$ is compatible with $\mathcal{C}_2$,*
2) *$\mathcal{C}'_1 \preceq \mathcal{C}_1$ and $\mathcal{C}'_2 \preceq \mathcal{C}_2$ hold,*

then $\mathcal{C}'_1$ is compatible with $\mathcal{C}'_2$ and $\mathcal{C}'_1 \otimes \mathcal{C}'_2 \preceq \mathcal{C}_1 \otimes \mathcal{C}_2$.

Thus, compatible contracts can be independently refined. This property holds in particular if $\mathcal{C}'_1$ and $\mathcal{C}'_2$ are singletons:

Corollary 1: Compatible contracts can be independently implemented.

Property 3 is fundamental, particularly in top-down design. Top-down incremental design consists in iteratively decomposing a system-level contract $\mathcal{C}$ into sub-system contracts $\mathcal{C}_i, i \in I$ for further independent development. To ensure that independent development will not lead to integration problems, it is enough to verify that $\bigotimes_{i\in I} \mathcal{C}_i \preceq \mathcal{C}$. We insist that, since contracts are purposely abstract and subsystems are not many, the composition of contracts $\mathcal{C}_i$ will not typically result in state explosion.[2]

The following property is essential as it states that contract composition can be performed in any order and changes in architecture (captured by changes in parenthesizing) are allowed:

Property 4 (associativity and commutativity): For all contracts $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are compatible, $\mathcal{C}_3$ and $\mathcal{C}_4$ are compatible and $\mathcal{C}_1 \otimes \mathcal{C}_2$ is compatible with $\mathcal{C}_3 \otimes \mathcal{C}_4$, then $\mathcal{C}_1$ is compatible with $\mathcal{C}_3$, $\mathcal{C}_2$ is compatible with $\mathcal{C}_4$, $\mathcal{C}_1 \otimes \mathcal{C}_3$ is compatible with $\mathcal{C}_2 \otimes \mathcal{C}_4$, and

$$(\mathcal{C}_1 \otimes \mathcal{C}_2) \otimes (\mathcal{C}_3 \otimes \mathcal{C}_4) \quad = \quad (\mathcal{C}_1 \otimes \mathcal{C}_3) \otimes (\mathcal{C}_2 \otimes \mathcal{C}_4) \quad (1)$$

Proof: To shorten notations, write $\mathcal{C}_{12}$ instead of $\mathcal{C}_1 \otimes \mathcal{C}_2$ and similarly for any subset of $\{1, 2, 3, 4\}$. By Assumption 2 and the associativity and commutativity of component composition, $\mathcal{C}_{1234}$ is characterized by the following two properties, where index i ranges over the set $1 \ldots 4$:

$$\begin{array}{rcl} M_i \models^{\mathsf{imp}} \mathcal{C}_i & \Rightarrow & M_1 \times \ldots \times M_4 \models^{\mathsf{imp}} \mathcal{C}_{1234} \\ E \models^{\mathsf{env}} \mathcal{C}_{1234} & \Rightarrow & E \times (\times_{j\neq i} M_j) \models^{\mathsf{env}} \mathcal{C}_i \end{array} \quad (2)$$

Observe that (2) is fully symmetric, which proves (1). Next, using the assumptions regarding compatibility, we derive the existence of at least one environment E satisfying the premise of the second implication of (2). Since (2) is fully symmetric, this proves the conclusions of Property 4 regarding compatibility. □

Property 5 (distributivity): If the following contract compositions are all well defined, then the following holds:

$$[(\mathcal{C}_{11} \wedge \mathcal{C}_{21}) \otimes (\mathcal{C}_{12} \wedge \mathcal{C}_{22})] \preceq [(\mathcal{C}_{11} \otimes \mathcal{C}_{12}) \wedge (\mathcal{C}_{21} \otimes \mathcal{C}_{22})] \quad (3)$$

[2] This is unlike in *compositional verification*, where $\times_{i\in I} M_i \models^{\mathsf{imp}} P$ is to be checked, where M_i are detailed implementations and P is a property. In this context, I may be a large set, and thus the composition $\times_{i\in I} M_i$ typically gives raise to state explosion. Techniques have thus been proposed to verify such properties in an incremental way [26], [9], [20], [1], [21].

Proof: By Lemma 1, $\mathbf{C}_{(\mathcal{C}_{11}\wedge\mathcal{C}_{21}),(\mathcal{C}_{12}\wedge\mathcal{C}_{22})} \supseteq \mathbf{C}_{\mathcal{C}_{11},\mathcal{C}_{12}}$. Taking the GLB of these two sets thus yields $[(\mathcal{C}_{11} \wedge \mathcal{C}_{21}) \otimes (\mathcal{C}_{12} \wedge \mathcal{C}_{22})] \preceq \mathcal{C}_{11} \otimes \mathcal{C}_{12}$ and similarly for $\mathcal{C}_{21} \otimes \mathcal{C}_{22}$. Thus, (3) follows. □

The use of distributivity is best illustrated in the following context. Suppose the system under design decomposes into two sub-systems labeled 1 and 2, and each subsystem has two viewpoints associated with it, labeled by another index with values 1 or 2. Contract $\mathcal{C}_{11} \wedge \mathcal{C}_{21}$ is then the contract associated with sub-system 1 and similarly for sub-system 2. Thus, the left hand side of (3) specifies the set of implementations obtained by, first, implementing each sub-system independently, and then, composing these implementations. Property 5 states that, by doing so, we obtain an implementation of the overall contract obtained by, first, getting the two global viewpoints $\mathcal{C}_{11} \otimes \mathcal{C}_{12}$ and $\mathcal{C}_{21} \otimes \mathcal{C}_{22}$, and, then, taking their conjunction. This property supports independent implementation for specifications involving multiple viewpoints. Observe that only refinement, not equality, holds in (3).

E. Issues of effectiveness

For some contract frameworks, all relations or operators listed in Table I can be effectively computed, see [6] for examples of such frameworks. In most cases, however, and particularly when data taking values in infinite domains are involved, this no longer holds. Two kinds of techniques can be used to overcome this. Testing and simulation techniques can be used to *disprove* properties; *observers* [6] adapt the principles of testing to contracts by providing negative semi-decision procedures. In this paper we complement observers by *abstractions* allowing to *prove* properties of contracts, thus providing positive semi-decision procedures.

III. Abstractions for contracts

An abstraction consists of an abstract domain of contracts—intended to be simple enough to support analysis—together with a mapping, from contracts (we call them "concrete contracts" in the sequel) to abstract contracts. The hope is that properties of contracts can be proved by taking abstractions thereof.

In this section we explain how to lift, to contracts, abstraction procedures available on components. In doing so, our objectives are the following:

1) Abstraction for contracts should allow proving refinement, consistency, or compatibility, for any contract or sets of contracts, based on their abstractions;
2) Properties of contracts should be deducible from their abstractions, compositionally with respect to both conjunction and parallel composition;
3) The mechanism of lifting abstractions, from components to contracts should be generic and instantiable for any concrete contract framework.

A large part of this agenda—though not all of it—is achievable, as we shall see now. Our starting point is thus a framework for abstracting components. This framework must be rich enough to support abstraction and its opposite operation

in a coherent way. A known formalization of this is the notion of Galois connection, which is key in the theory of Abstract Interpretation [10], [11], [12], [23].

A. Background on Galois connections

A *Galois connection* consists of two *concrete* and *abstract* partially ordered sets $(\mathcal{X}_\mathbf{c}, \sqsubseteq_\mathbf{c})$ and $(\mathcal{X}_\mathbf{a}, \sqsubseteq_\mathbf{a})$, and two total monotonic maps:[3]

$$\begin{array}{rcl} \alpha : \mathcal{X}_\mathbf{c} \mapsto \mathcal{X}_\mathbf{a} & : & \text{the } \textit{abstraction} \\ \gamma : \mathcal{X}_\mathbf{a} \mapsto \mathcal{X}_\mathbf{c} & : & \text{the } \textit{concretization} \end{array}$$

such that, for any two $X_\mathbf{c} \in \mathcal{X}_\mathbf{c}$ and $X_\mathbf{a} \in \mathcal{X}_\mathbf{a}$,

$$X_\mathbf{c} \sqsubseteq_\mathbf{c} \gamma(X_\mathbf{a}) \quad \text{if and only if} \quad \alpha(X_\mathbf{c}) \sqsubseteq_\mathbf{a} X_\mathbf{a} \tag{4}$$

Property (4) is equivalent to any of the following properties:

$$X_\mathbf{c} \sqsubseteq_\mathbf{c} \gamma \circ \alpha(X_\mathbf{c}) \quad ; \quad \alpha \circ \gamma(X_\mathbf{a}) \sqsubseteq_\mathbf{a} X_\mathbf{a} \tag{5}$$

where $\gamma \circ \alpha$ is the composition of the two referred maps: $\gamma \circ \alpha(X_\mathbf{c}) =_{\text{def}} \gamma(\alpha(X_\mathbf{c}))$.

The intent is that $\mathcal{X}_\mathbf{c}$ is the concrete domain of interest and $\mathcal{X}_\mathbf{a}$ is a simpler and coarser representation of the former, where concrete entities can be approximated. The two orders $\sqsubseteq_{\mathbf{c}/\mathbf{a}}$ are interpreted as "is more precise"—for example, if components are specified as sets of behaviors, the preciseness order is simply set inclusion. The Galois connection property (4) relates the preciseness orders in concrete and abstract domains.

The following tool, originally found in [11], is extremely convenient in getting Galois connections:

Theorem 1 (see Theorem 2.2.1 of [23]):

1) *If $(\mathcal{X}_\mathbf{c}, \sqsubseteq_\mathbf{c})$ has LUBs for arbitrary sets and α is a complete $\sqcup_\mathbf{c}$-morphism,[4] then there exists a unique concretization γ such that (α, γ) is a Galois connection from $(\mathcal{X}_\mathbf{c}, \sqsubseteq_\mathbf{c})$ to $(\mathcal{X}_\mathbf{a}, \sqsubseteq_\mathbf{a})$. It is given by*

$$\gamma(X_\mathbf{a}) \quad = \quad \sqcup_\mathbf{c}\{X_\mathbf{c} \mid \alpha(X_\mathbf{c}) \sqsubseteq_\mathbf{a} X_\mathbf{a}\}$$

2) *If $(\mathcal{X}_\mathbf{a}, \sqsubseteq_\mathbf{a})$ has GLBs for arbitrary sets and γ is a complete $\sqcap_\mathbf{a}$-morphism, then there exists a unique abstraction α such that (α, γ) is a Galois connection from $(\mathcal{X}_\mathbf{c}, \sqsubseteq_\mathbf{c})$ to $(\mathcal{X}_\mathbf{a}, \sqsubseteq_\mathbf{a})$. It is given by*

$$\alpha(X_\mathbf{c}) \quad = \quad \sqcap_\mathbf{a}\{X_\mathbf{a} \mid X_\mathbf{c} \sqsubseteq_\mathbf{c} \gamma(X_\mathbf{a})\}$$

B. Lifting abstractions, from components to contracts

Having the above notions at hand, our next step consists in systematically lifting a given Galois connection (α, γ) on components to an abstraction on contracts, as defined in Table I. Since contracts are defined as pairs consisting of a set of valid environments and a set of valid components, our first task is to lift Galois connections, from sets to powersets.

[3] $f : X \to Y$, where $(X, \leq_X)$ and $(Y, \leq_Y)$ are two ordered sets, is monotonic if $x' \leq_X x$ implies $f(x') \leq_Y f(x)$, and strictly monotonic if $x' <_X x$ implies $f(x') <_Y f(x)$, where $< =_{\text{def}} \leq \cap \neq$.

[4] Meaning that α preserves existing LUBs: $\alpha(\sqcup_\mathbf{c} X) = \sqcup_\mathbf{a}\{\alpha(x) \mid x \in X\}$.

Our construction will be using the notion of inverse map, which we recall next. For X and Y two sets, $f : X \to Y$ a partial function, and $Z \subseteq Y$, define

$$f^{-1}(Z) = \{x \in X \mid f(x) \text{ is defined and } f(x) \in Z\}$$

The following holds:

$$\begin{array}{rcl} f^{-1}(Z_1 \cap Z_2) & = & f^{-1}(Z_1) \cap f^{-1}(Z_2) \\ f^{-1}(Z_1 \cup Z_2) & = & f^{-1}(Z_1) \cup f^{-1}(Z_2) \end{array} \tag{6}$$

Referring to the notations of Section III-A, we consider the sets $\mathcal{X}_\mathbf{c}^< \subseteq 2^{\mathcal{X}_\mathbf{c}}$ and $\mathcal{X}_\mathbf{a}^< \subseteq 2^{\mathcal{X}_\mathbf{a}}$ collecting all ideals[5] of $(\mathcal{X}_\mathbf{c}, \sqsubseteq_\mathbf{c})$ and $(\mathcal{X}_\mathbf{a}, \sqsubseteq_\mathbf{a})$, respectively. Equip $\mathcal{X}_\mathbf{c}^<$ and $\mathcal{X}_\mathbf{a}^<$ with their inclusion orders $\subseteq_\mathbf{c}$ and $\subseteq_\mathbf{a}$. The *canonical abstraction*

$$\widehat{\alpha} : \quad (\mathcal{X}_\mathbf{c}^<, \subseteq_\mathbf{c}) \to (\mathcal{X}_\mathbf{a}^<, \subseteq_\mathbf{a})$$

associated to Galois connection (α, γ) is defined by

$$\widehat{\alpha}(\chi_\mathbf{c}) \ =_{\text{def}} \ \gamma^{-1}(\chi_\mathbf{c}) \tag{7}$$

where $\chi_\mathbf{c}$ ranges over $\mathcal{X}_\mathbf{c}^<$. Definition (7) is sound since γ is monotonic. The following property follows by construction:

$$\forall \chi_\mathbf{c} \in \mathcal{X}_\mathbf{c}^< : \quad \chi_\mathbf{c} = \emptyset \implies \widehat{\alpha}(\chi_\mathbf{c}) = \emptyset \tag{8}$$

We now instantiate the generic construction (7) by substituting $\mathcal{X}_\mathbf{c} \leftarrow \mathcal{M}_\mathbf{c}$ and $\mathcal{X}_\mathbf{a} \leftarrow \mathcal{M}_\mathbf{a}$. To this end, we assume the following, which expresses that the preciseness orders fit our contract framework:

Assumption 3: For any concrete contract $\mathcal{C}_\mathbf{c} \in \mathbb{C}_\mathbf{c}$ with semantics $[\![\mathcal{C}_\mathbf{c}]\!] = \langle \mathcal{C}_\mathbf{c}^{\mathsf{env}}, \mathcal{C}_\mathbf{c}^{\mathsf{imp}} \rangle$, both $\mathcal{C}_\mathbf{c}^{\mathsf{env}}$ and $\mathcal{C}_\mathbf{c}^{\mathsf{imp}}$ are downward closed under $\sqsubseteq_\mathbf{c}$. The same holds for abstract contracts.

As we shall see in Section IV, Assumption 3 is very natural for known contract frameworks.

By (7) we inherit an abstraction $\widehat{\alpha}$ from $(\mathcal{M}_\mathbf{c}^<, \subseteq)$ to $(\mathcal{M}_\mathbf{a}^<, \subseteq)$. Since the semantics of a concrete generic contract $\mathcal{C}_\mathbf{c}$ is $[\![\mathcal{C}_\mathbf{c}]\!] = \langle \mathcal{C}_\mathbf{c}^{\mathsf{env}}, \mathcal{C}_\mathbf{c}^{\mathsf{imp}} \rangle \in \mathcal{M}_\mathbf{c}^< \times \mathcal{M}_\mathbf{c}^<$, we can define the *abstraction* $\overline{\alpha}(\mathcal{C}_\mathbf{c})$ of $\mathcal{C}_\mathbf{c}$, whose semantics is:

$$\begin{array}{rcl} [\![\overline{\alpha}(\mathcal{C}_\mathbf{c})]\!] & =_{\text{def}} & \langle \widehat{\alpha}\left(\mathcal{C}_\mathbf{c}^{\mathsf{env}}\right), \widehat{\alpha}\left(\mathcal{C}_\mathbf{c}^{\mathsf{imp}}\right) \rangle \\ & \in & \mathcal{M}_\mathbf{a}^< \times \mathcal{M}_\mathbf{a}^< \end{array} \tag{9}$$

Definition 2: $\overline{\alpha}$ defined by (9) is the canonical abstraction *on contracts associated to the Galois connection (α, γ) on components.*

C. Using abstractions for proofs

The following theorem achieves our first objectives regarding contract abstraction:

Theorem 2: Let $M_\mathbf{c}, E_\mathbf{c}, \mathcal{C}_\mathbf{c}$ be a concrete component, environment, and contract.

1) *If $\alpha(M_\mathbf{c}) \models_\mathbf{a}^{\mathsf{imp}} \overline{\alpha}(\mathcal{C}_\mathbf{c})$ holds, then $M_\mathbf{c} \models_\mathbf{c}^{\mathsf{imp}} \mathcal{C}_\mathbf{c}$ follows.*
   If $\alpha(E_\mathbf{c}) \models_\mathbf{a}^{\mathsf{env}} \overline{\alpha}(\mathcal{C}_\mathbf{c})$ holds, then $E_\mathbf{c} \models_\mathbf{c}^{\mathsf{env}} \mathcal{C}_\mathbf{c}$ follows.
2) *If $\mathcal{C}'_\mathbf{c} \preceq_\mathbf{c} \mathcal{C}_\mathbf{c}$ holds, then $\overline{\alpha}(\mathcal{C}'_\mathbf{c}) \preceq_\mathbf{a} \overline{\alpha}(\mathcal{C}_\mathbf{c})$ follows.*
3) *If $\overline{\alpha}(\mathcal{C}_\mathbf{c})$ is compatible or consistent, then so is $\mathcal{C}_\mathbf{c}$.*

Proof: Statement 3 follows immediately from (8).

[5] An *ideal* of $(\mathcal{X}, \sqsubseteq)$ is a $\sqsubseteq$-downward closed subset of $\mathcal{X}$.

Focus next on Statement 2. Since set abstraction $\widehat{\alpha}$ is monotonic with respect to set inclusion, we deduce that contract abstraction $\overline{\alpha}$ is monotonic for $\preceq_{\mathbf{c}/\mathbf{a}}$.

Regarding Statement 1, $\alpha(M_{\mathbf{c}}) \models_{\mathbf{a}}^{\mathsf{imp}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}})$ means that $\gamma(\alpha(M_{\mathbf{c}})) \in \mathcal{C}_{\mathbf{c}}^{\mathsf{imp}}$. By (5), $M_{\mathbf{c}} \sqsubseteq_{\mathbf{c}} \gamma(\alpha(M_{\mathbf{c}}))$, which, by Assumption 3, implies $M_{\mathbf{c}} \in \mathcal{C}_{\mathbf{c}}^{\mathsf{imp}}$, i.e., $M_{\mathbf{c}} \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}$. Similarly, $\alpha(E_{\mathbf{c}}) \models_{\mathbf{a}}^{\mathsf{env}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}})$ means that $\gamma(\alpha(E_{\mathbf{c}})) \in \mathcal{C}_{\mathbf{c}}^{\mathsf{env}}$. By (5), $E_{\mathbf{c}} \sqsubseteq_{\mathbf{c}} \gamma(\alpha(E_{\mathbf{c}}))$, which, by Assumption 3, implies $E_{\mathbf{c}} \in \mathcal{C}_{\mathbf{c}}^{\mathsf{env}}$, i.e., $E_{\mathbf{c}} \models_{\mathbf{c}}^{\mathsf{env}} \mathcal{C}_{\mathbf{c}}$. □

Observe that Statement 1 allows proving implementation and environment relations based on abstractions. Similarly, Statement 3 allows proving compatibility or consistency based on abstractions. In contrast, Statement 2 allows disproving refinement based on abstractions.

D. Compositionality of abstraction

The second part of our agenda is about compositionality of abstraction, with respect to both conjunction and parallel composition. Observe first that Statement 2 of Theorem 2 implies $\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \wedge \mathcal{C}_{\mathbf{c}}^2) \preceq_{\mathbf{a}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \wedge \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2)$, etc. Using, however, the fact that abstraction and concretizations for powersets arise from inverse maps, we can in fact get equalities:

Theorem 3: The following equalities hold:

$$\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \wedge \mathcal{C}_{\mathbf{c}}^2) \quad = \quad \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \wedge \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2) \tag{10}$$

Proof: By definition,

$$\begin{aligned}
\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \wedge \mathcal{C}_{\mathbf{c}}^2) &= (\widehat{\alpha}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{env}} \cup (\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{env}}), \\
&\quad \widehat{\alpha}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{imp}} \cap (\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{imp}})) \\
\text{(by (9))} &= (\gamma^{-1}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{env}} \cup (\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{env}}), \\
&\quad \gamma^{-1}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{imp}} \cap (\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{imp}})) \\
\text{(by (6))} &= (\gamma^{-1}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{env}}) \cup \gamma^{-1}((\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{env}}), \\
&\quad \gamma^{-1}((\mathcal{C}_{\mathbf{c}}^1)^{\mathsf{imp}}) \cap \gamma^{-1}((\mathcal{C}_{\mathbf{c}}^2)^{\mathsf{imp}})) \\
&= \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \wedge \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2)
\end{aligned}$$

which finishes the proof. □

The last property in our agenda concerns parallel composition of contracts. We wish to relate $\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \otimes \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2)$ and $\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \otimes \mathcal{C}_{\mathbf{c}}^2)$. Unlike previous properties, this does not come for free. We first need an additional property for the concretization of components γ:

Definition 3: γ is called sub-multiplicative *if*

$$\gamma(X_{\mathbf{a}}^1 \times_{\mathbf{a}} X_{\mathbf{a}}^2) \quad \sqsubseteq_{\mathbf{c}} \quad \gamma(X_{\mathbf{a}}^1) \times_{\mathbf{c}} \gamma(X_{\mathbf{a}}^2) \tag{11}$$

and multiplicative *if equality holds in* (11).

Theorem 4:

1) *If γ is sub-multiplicative, then*

$$\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \otimes \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2) \quad \preceq_{\mathbf{a}} \quad \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \otimes \mathcal{C}_{\mathbf{c}}^2) \tag{12}$$

2) *If, in addition, γ is multiplicative, then the two contracts $\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \otimes \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2)$ and $\overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1 \otimes \mathcal{C}_{\mathbf{c}}^2)$ possess identical sets of implementations—their sets of valid environments, however, may differ.*

3) *If, in addition to the condition stated in 2), $\mathcal{C}_{\mathbf{c}}^1$ and $\mathcal{C}_{\mathbf{c}}^2$ satisfy $X_{\mathbf{c}} \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i \implies \gamma \circ \alpha(X_{\mathbf{c}}) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i$, then, equality holds in* (12).

Proof: For $\mathcal{M}_1, \mathcal{M}_2 \subseteq \mathcal{M}$, set

$$\mathcal{M}_1 \bar{\times} \mathcal{M}_2 \quad = \quad \{M_1 \times M_2 \mid M_1 \in \mathcal{M}_1, M_2 \in \mathcal{M}_2\}$$

Using this notation we can rewrite as follows the formula defining, in Table I, the parallel composition of contracts:

$$\mathcal{C}_1 \otimes \mathcal{C}_2 \quad = \quad \min \left\{ \mathcal{C} \;\middle|\; \begin{array}{lcl} (\mathcal{C}_1^{\mathsf{imp}} \bar{\times} \mathcal{C}_2^{\mathsf{imp}}) & \subseteq & \mathcal{C}^{\mathsf{imp}} \\ (\mathcal{C}^{\mathsf{env}} \bar{\times} \mathcal{C}_2^{\mathsf{imp}}) & \subseteq & \mathcal{C}_1^{\mathsf{env}} \\ (\mathcal{C}^{\mathsf{env}} \bar{\times} \mathcal{C}_1^{\mathsf{imp}}) & \subseteq & \mathcal{C}_2^{\mathsf{env}} \end{array} \right\} \tag{13}$$

Statement 1 set $\mathcal{C}_{\mathbf{c}} = \mathcal{C}_{\mathbf{c}}^1 \otimes \mathcal{C}_{\mathbf{c}}^2$ and $\mathcal{C}_{\mathbf{a}} = \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^1) \otimes \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^2)$.

Consider (12) and focus on sets of implementations. We need to prove that $\mathcal{C}_{\mathbf{a}}^{\mathsf{imp}} \subseteq (\overline{\alpha}(\mathcal{C}_{\mathbf{c}}))^{\mathsf{imp}}$, i.e., using (9): $\mathcal{C}_{\mathbf{a}}^{\mathsf{imp}} \subseteq \widehat{\alpha}(\mathcal{C}_{\mathbf{c}}^{\mathsf{imp}})$, which is, by (7), equivalent to

$$\forall M_{\mathbf{a}} \;:\; M_{\mathbf{a}} \models_{\mathbf{a}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{a}} \quad \implies \quad \gamma(M_{\mathbf{a}}) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}} \tag{14}$$

By (13) it is enough to prove (14) when $M_{\mathbf{a}}$ has the restricted form $M_{\mathbf{a}} = M_{\mathbf{a}}^1 \times_{\mathbf{a}} M_{\mathbf{a}}^2$, where $M_{\mathbf{a}}^i \models_{\mathbf{a}}^{\mathsf{imp}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^i)$, which, by using (9), is equivalent to $\gamma(M_{\mathbf{a}}^i) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i$. Thus (14) amounts to proving, for any two abstract components $M_{\mathbf{a}}^1, M_{\mathbf{a}}^2$:

$$\gamma(M_{\mathbf{a}}^i) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i \quad \implies \quad \gamma(M_{\mathbf{a}}^1 \times_{\mathbf{a}} M_{\mathbf{a}}^2) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}} \tag{15}$$

By (13) $\gamma(M_{\mathbf{a}}^i) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i$ implies $\gamma(M_{\mathbf{a}}^1) \times_{\mathbf{a}} \gamma(M_{\mathbf{a}}^2) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^1 \otimes \mathcal{C}_{\mathbf{c}}^2 = \mathcal{C}_{\mathbf{c}}$. Since γ is sub-multiplicative, we have $\gamma(M_{\mathbf{a}}^1 \times_{\mathbf{a}} M_{\mathbf{a}}^2) \sqsubseteq_{\mathbf{c}} \gamma(M_{\mathbf{a}}^1) \times_{\mathbf{c}} \gamma(M_{\mathbf{a}}^2)$ and we deduce (15) by invoking Assumption 3. This proves the implementation part of (12).

Focus next on environments. We need to prove that $\mathcal{C}_{\mathbf{a}}^{\mathsf{env}} \supseteq (\overline{\alpha}(\mathcal{C}_{\mathbf{c}}))^{\mathsf{env}}$, i.e., using (9): $\mathcal{C}_{\mathbf{a}}^{\mathsf{env}} \supseteq \widehat{\alpha}((\mathcal{C}_{\mathbf{c}})^{\mathsf{env}})$, which is, by (7), equivalent to

$$\forall E_{\mathbf{a}} \;:\; E_{\mathbf{a}} \models_{\mathbf{a}}^{\mathsf{env}} \mathcal{C}_{\mathbf{a}} \quad \impliedby \quad \gamma(E_{\mathbf{a}}) \models_{\mathbf{c}}^{\mathsf{env}} \mathcal{C}_{\mathbf{c}} \tag{16}$$

By (13), the left hand side of (16) is equivalent to: for any $M_{\mathbf{a}}^i \models_{\mathbf{a}}^{\mathsf{imp}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^i)$, or, equivalently, $\gamma(M_{\mathbf{a}}^i) \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i$, the following holds: $E_a \times_{\mathbf{a}} M_{\mathbf{a}}^i \models_{\mathbf{a}}^{\mathsf{env}} \overline{\alpha}(\mathcal{C}_{\mathbf{c}}^j)$ where $j \neq i$, which is equivalent to $\gamma(E_a \times_{\mathbf{a}} M_{\mathbf{a}}^i) \models_{\mathbf{c}}^{\mathsf{env}} \mathcal{C}_{\mathbf{c}}^j$. On the other hand, the right hand side of (16) is equivalent to: for any $M_{\mathbf{c}}^i \models_{\mathbf{c}}^{\mathsf{imp}} \mathcal{C}_{\mathbf{c}}^i$, the following holds: $\gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} M_{\mathbf{c}}^i \models_{\mathbf{c}}^{\mathsf{env}} \mathcal{C}_{\mathbf{c}}^j$ where $j \neq i$. To summarize we need to prove the folllowing, for any abstract environment $E_{\mathbf{a}}$:

$$\begin{array}{c} \left[\begin{array}{ccc} \forall M_{\mathbf{c}}^i : M_{\mathbf{c}}^i & \models_{\mathbf{c}}^{\mathsf{imp}} & \mathcal{C}_{\mathbf{c}}^i \\ & \Downarrow & \\ \gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} M_{\mathbf{c}}^i & \models_{\mathbf{c}}^{\mathsf{env}} & \mathcal{C}_{\mathbf{c}}^j \end{array}\right] \\ \Downarrow \\ \left[\begin{array}{ccc} \forall M_{\mathbf{a}}^i : \gamma(M_{\mathbf{a}}^i) & \models_{\mathbf{c}}^{\mathsf{imp}} & \mathcal{C}_{\mathbf{c}}^i \\ & \Downarrow & \\ \gamma(E_a \times_{\mathbf{a}} M_{\mathbf{a}}^i) & \models_{\mathbf{c}}^{\mathsf{env}} & \mathcal{C}_{\mathbf{c}}^j \end{array}\right] \end{array} \tag{17}$$

We can restrict the quantification in the up side of (17) to the subset of $M_{\mathbf{c}}^i$ of the form $M_{\mathbf{c}}^i = \gamma(M_{\mathbf{a}}^i)$. Since γ is sub-multiplicative, we get $\gamma(E_a \times_{\mathbf{a}} M_{\mathbf{a}}^i) \sqsubseteq_{\mathbf{c}} \gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} \gamma(M_{\mathbf{a}}^i)$,

which, by Assumption 3, implies the down side of (17) and proves statement 1 of the theorem.

Statement 2: With reference to (14), we now need to prove

$$\forall M_{\mathbf{a}} \;:\; M_{\mathbf{a}} \models^{\mathsf{imp}}_{\mathbf{a}} \mathcal{C}_{\mathbf{a}} \iff \gamma(M_{\mathbf{a}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}_{\mathbf{c}} \qquad (18)$$

Proof obligation (15) is then replaced by the following stronger one. For any two abstract components $M^1_{\mathbf{a}}, M^2_{\mathbf{a}}$:

$$\gamma(M^i_{\mathbf{a}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}^i_{\mathbf{c}} \iff \gamma(M^1_{\mathbf{a}} \times_{\mathbf{a}} M^2_{\mathbf{a}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}_{\mathbf{c}} \qquad (19)$$

By (13) $\gamma(M^i_{\mathbf{a}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}^i_{\mathbf{c}}$ if and only if $\gamma(M^1_{\mathbf{a}}) \times_{\mathbf{a}} \gamma(M^2_{\mathbf{a}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}^1_{\mathbf{c}} \otimes \mathcal{C}^2_{\mathbf{c}} = \mathcal{C}_{\mathbf{c}}$. Since γ is multiplicative, we have $\gamma(M^1_{\mathbf{a}} \times_{\mathbf{a}} M^2_{\mathbf{a}}) = \gamma(M^1_{\mathbf{a}}) \times_{\mathbf{c}} \gamma(M^2_{\mathbf{a}})$, which shows (19). This proves statement 2.

Statement 3: With reference to (16), we now need to prove

$$\forall E_{\mathbf{a}} \;:\; E_{\mathbf{a}} \models^{\mathsf{env}}_{\mathbf{a}} \mathcal{C}_{\mathbf{a}} \iff \gamma(E_{\mathbf{a}}) \models^{\mathsf{env}}_{\mathbf{c}} \mathcal{C}_{\mathbf{c}} \qquad (20)$$

Proof obligation (17) is then replaced by the following stronger one, for any abstract environment $E_{\mathbf{a}}$:

$$\begin{bmatrix} \forall M^i_{\mathbf{a}} : \gamma(M^i_{\mathbf{a}}) & \models^{\mathsf{imp}}_{\mathbf{c}} & \mathcal{C}^i_{\mathbf{c}} \\ & \Downarrow & \\ \gamma(E_a \times_{\mathbf{a}} M^i_{\mathbf{a}}) & \models^{\mathsf{env}}_{\mathbf{c}} & \mathcal{C}^j_{\mathbf{c}} \end{bmatrix} \iff \begin{bmatrix} \forall M^i_{\mathbf{c}} : M^i_{\mathbf{c}} & \models^{\mathsf{imp}}_{\mathbf{c}} & \mathcal{C}^i_{\mathbf{c}} \\ & \Downarrow & \\ \gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} M^i_{\mathbf{c}} & \models^{\mathsf{env}}_{\mathbf{c}} & \mathcal{C}^j_{\mathbf{c}} \end{bmatrix} \qquad (21)$$

If we were able to restrict the quantification over $M^i_{\mathbf{c}}$ to the subset of the form $\gamma(M^i_{\mathbf{a}})$, then having γ multiplicative would imply (21) and we would be done with statement 3. To justify this we invoke the special condition for this case: for any $M^i_{\mathbf{c}} \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}^i_{\mathbf{c}}$, $\gamma \circ \alpha(M^i_{\mathbf{c}}) \models^{\mathsf{imp}}_{\mathbf{c}} \mathcal{C}^i_{\mathbf{c}}$ follows and the right hand side of (21) implies $\gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} \gamma \circ \alpha(M^i_{\mathbf{c}}) \models^{\mathsf{env}}_{\mathbf{c}} \mathcal{C}^j_{\mathbf{c}}$. On the other hand, by (5), $M^i_{\mathbf{c}} \sqsubseteq_{\mathbf{c}} \gamma \circ \alpha(M^i_{\mathbf{c}})$ and, thus, by monotonicity of $\times_{\mathbf{c}}$, $\gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} M^i_{\mathbf{c}} \sqsubseteq_{\mathbf{c}} \gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} \gamma \circ \alpha(M^i_{\mathbf{c}})$ and, thus, by Assumption 3, $\gamma(E_{\mathbf{a}}) \times_{\mathbf{c}} M^i_{\mathbf{c}} \models^{\mathsf{env}}_{\mathbf{c}} \mathcal{C}^j_{\mathbf{c}}$ follows. This shows that, for this case, we can restrict the quantification over $M^i_{\mathbf{c}}$ to the subset of the form $\gamma(M^i_{\mathbf{a}})$, so we are done with the theorem. □

E. Concluding discussion

a) From having a Galois connection on components we inherit an abstraction on contracts that is monotonic with respect to the refinement orders. Consistency and compatibility can both be checked on abstractions, see Theorem 2. The reader may conjecture that it should be possible to construct a Galois connection for contracts. We are rather convinced that this is not achievable; obstructions exist and we argue about this in Section V.

b) Theorem 3 allows checking consistency and compatibility in a $\wedge$-modular way by using equality

$$\overline{\alpha}(\textstyle\bigwedge_{i\in I} \mathcal{C}^i_{\mathbf{c}}) \;=\; \bigwedge_{i\in I} \overline{\alpha}(\mathcal{C}^i_{\mathbf{c}})$$

c) If γ is sub-multiplicative, Theorem 4 allows checking consistency in a $\otimes$-modular way by using refinement

$$\textstyle\bigotimes_{i\in I} \overline{\alpha}(\mathcal{C}^i_{\mathbf{c}}) \;\preceq_{\mathbf{a}}\; \overline{\alpha}(\bigotimes_{i\in I} \mathcal{C}^i_{\mathbf{c}})$$

This inequality is in the wrong way, however, for checking compatibility in a $\otimes$-modular way. Regarding this theorem, Galois connections on components where concretization is multiplicative are quite natural. Thus, Properties 1) and 2) of Theorem 4 will be easy to have. In contrast, the special condition, needed to have Property 3), arises only in exceptional cases. We conjecture that Theorem 4 is the best achievable result regarding compositionality of abstraction with respect to $\otimes$.

IV. The case of A/G-contracts

We investigate the case of A/G-contracts when components are specified: first, as sets of behaviors, and, second, by using transition relations (which is more practical).

A. Components specified as sets of behaviors

In this section we consider components M specified as sets of behaviors. Component composition is by intersection: $M_1 \times M_2 =_{\text{def}} M_1 \cap M_2$.

An *A/G-contract* is a pair $\mathcal{C} = (A, G)$ of assertions, called the *assumptions* and the *guarantees*. The set $\mathcal{C}^{\mathsf{env}}$ of the legal environments for $\mathcal{C}$ collects all components E such that $E \subseteq A$. The set $\mathcal{C}^{\mathsf{imp}}$ of all components implementing $\mathcal{C}$ is defined by $A \times M \subseteq G$. Thus, any component M such that $M \leq G \cup \neg A$, where $\neg A$ denotes the complement of set A, is an implementation of $\mathcal{C}$ and $M_{\mathcal{C}} = G \cup \neg A$ is the maximal implementation. Observe that two contracts $\mathcal{C}$ and $\mathcal{C}'$ with identical input and output alphabets, identical assumptions, and such that $G' \cup \neg A' = G \cup \neg A$, possess identical sets of implementations: $\mathcal{C}^{\mathsf{imp}} = \mathcal{C}'^{\mathsf{imp}}$. According to our meta-theory, such two contracts are equivalent. Any contract $\mathcal{C} = (A, G)$ is equivalent to a contract in *saturated form* (A, G') such that $G' \supseteq \neg A$, or, equivalently, $G \cup A = \mathsf{T}$, the true assertion; to exhibit G', just take $G' = G \cup \neg A$. A saturated contract $\mathcal{C} = (A, G)$ is *consistent* if and only if $G \neq \emptyset$ and *compatible* if and only if $A \neq \emptyset$.

Next, for $\mathcal{C}$ and $\mathcal{C}'$ two saturated contracts with identical input and output alphabets,

$$\textit{refinement} \;\; \mathcal{C}' \preceq \mathcal{C} \;\; \text{holds iff} \; \begin{cases} A' \supseteq A \\ G' \subseteq G \end{cases} \qquad (22)$$

whence Assumptions 1 and 2 of the meta-theory hold for A/G contracts. *Conjunction* follows from the refinement relation: for $\mathcal{C}_1$ and $\mathcal{C}_2$ two saturated contracts with identical input and output alphabets: $\mathcal{C}_1 \wedge \mathcal{C}_2 = (A_1 \cup A_2, G_1 \cap G_2)$.

Focus now on contract *composition* $\mathcal{C} = \mathcal{C}_1 \otimes \mathcal{C}_2$. With reference to Table I, contract composition instantiates through the following formulas:

$$\begin{aligned} G &= G_1 \cap G_2 \\ A &= \max \left\{ A \;\middle|\; \begin{array}{l} A \cap G_2 \subseteq A_1 \\ \text{and} \\ A \cap G_1 \subseteq A_2 \end{array} \right\} \end{aligned} \qquad (23)$$

which satisfies Assumption 2 of the meta-theory. If, furthermore, contracts are saturated, then (23) reformulates as the

formulas originally proposed in [5]:

$$\begin{array}{rcl} G & = & G_1 \cap G_2 \\ A & = & (A_1 \cap A_2) \cup \neg(G_1 \cap G_2) \end{array} \tag{24}$$

Observe that the so obtained contract (A, G) is saturated: $G \cup A = (G_1 \cap G_2) \cup (A_1 \cap A_2) \cup \neg(G_1 \cap G_2) = \mathit{true}$.

We are now ready to introduce abstractions for A/G-contracts. We assume two classes $\mathcal{M}_\mathbf{c}$ and $\mathcal{M}_\mathbf{a}$ of concrete and abstract components, seen as sets of behaviors. $\mathcal{M}_\mathbf{c}$ and $\mathcal{M}_\mathbf{a}$ are ordered by set inclusion. We assume a Galois connection

$$(\alpha, \gamma), \text{ from } (\mathcal{M}_\mathbf{c}, \subseteq_\mathbf{c}) \text{ to } (\mathcal{M}_\mathbf{a}, \subseteq_\mathbf{a}) \tag{25}$$

Concrete and abstract A/G-contracts are pairs $\mathcal{C}_\mathbf{c} = (A_\mathbf{c}, G_\mathbf{c}) \in \mathcal{M}_\mathbf{c} \times \mathcal{M}_\mathbf{c}$ and $\mathcal{C}_\mathbf{a} = (A_\mathbf{a}, G_\mathbf{a}) \in \mathcal{M}_\mathbf{a} \times \mathcal{M}_\mathbf{a}$ of (concrete and abstract) assumptions and guarantees. $E_\mathbf{c} \models_\mathbf{c}^{\mathsf{env}} \mathcal{C}_\mathbf{c}$ iff $E_\mathbf{c}$ satisfies $A_\mathbf{c}$, i.e., $E_\mathbf{c} \subseteq A_\mathbf{c}$. $M_\mathbf{c} \models_\mathbf{c}^{\mathsf{imp}} \mathcal{C}_\mathbf{c}$ iff $M_\mathbf{c} \cap A_\mathbf{c}$ satisfies $G_\mathbf{c}$, i.e., $M_\mathbf{c} \cap A_\mathbf{c} \subseteq G_\mathbf{c}$. Since preciseness order $\sqsubseteq_{\mathbf{c}/\mathbf{a}}$ is set inclusion, Assumption 3 holds. Using formulas (7) for this case yields, for $\chi_\mathbf{c}$ a downward closed set of concrete components:

$$\widehat{\alpha}(\chi_\mathbf{c}) \quad = \quad \gamma^{-1}(\chi_\mathbf{c})$$

Instantiating this for sets of environments and implementations associated to A/G-contracts $\mathcal{C}_\mathbf{c} = (A_\mathbf{c}, G_\mathbf{c})$ and $\mathcal{C}_\mathbf{a} = (A_\mathbf{a}, G_\mathbf{a})$ yields:

$$\begin{array}{rcl} \widehat{\alpha}(\mathcal{C}_\mathbf{c}^{\mathsf{env}}) & = & \gamma^{-1}(\mathcal{C}_\mathbf{c}^{\mathsf{env}}) \\ & = & \{E_\mathbf{a} \mid \gamma(E_\mathbf{a}) \subseteq A_\mathbf{c}\} \\ \text{(by using (4))} & = & \{E_\mathbf{a} \mid E_\mathbf{a} \subseteq \alpha(A_\mathbf{c})\} \end{array}$$

and

$$\begin{array}{rcl} \widehat{\alpha}(\mathcal{C}_\mathbf{c}^{\mathsf{imp}}) & = & \gamma^{-1}(\mathcal{C}_\mathbf{c}^{\mathsf{imp}}) \\ & = & \{M_\mathbf{a} \mid \gamma(M_\mathbf{a}) \cap A_\mathbf{c} \subseteq G_\mathbf{c}\} \\ \text{(by using (4))} & = & \{M_\mathbf{a} \mid M_\mathbf{a} \subseteq \alpha(G_\mathbf{c} \cup \neg A_\mathbf{c})\} \end{array}$$

To summarize:

$$\overline{\alpha}(A_\mathbf{c}, G_\mathbf{c}) \quad = \quad (\alpha(A_\mathbf{c}), \alpha(G_\mathbf{c} \cup \neg A_\mathbf{c})) \tag{26}$$

It remains to explain how to construct a Galois connection (25). We do this in two steps. To be able to apply Theorem 4, we are interested in knowing if γ is (sub)-multiplicative.

B. Components specified as transition systems

Here we assume that components specified as transition systems, i.e., through their initial condition and transition relation, specified as predicates over a set of variables. We thus assume a set V of variables $x \in V$ with domains D_x and we set $D = \prod_{x \in V} D_x$. Initial conditions and transition relations are predicates $I \subseteq D$ and $R \subseteq D \times D$, that is,

$$(I, R) \in \mathcal{N} \quad =_{\text{def}} \quad 2^D \times 2^{D \times D} \tag{27}$$

A component seen as a set of synchronous behaviors over V is a subset

$$M \subseteq \mathcal{M} \quad =_{\text{def}} \quad \mathbb{N} \to D \tag{28}$$

Having an initial condition and a transition relation uniquely determines a component. The association map

$$\Phi : \quad \mathcal{N} \ni (I, R) \quad \mapsto \quad \Phi(I, R) \in \mathcal{M} \tag{29}$$

is defined by

$$\begin{array}{rl} & v_0, v_1, \ldots, v_n, \cdots \in \Phi(I, R) \\ \overset{\text{def}}{\iff} & v_0 \in I \text{ and } \forall k \geq 1 : (v_{k-1}, v_k) \in R \end{array}$$

Observe that Φ is monotonic. Define

$$\Phi^\dagger : \quad \mathcal{M} \ni M \quad \mapsto \quad \Phi^\dagger(M) = (I, R) \in \mathcal{N} \tag{30}$$

by

$$\begin{array}{rcl} v \in I & \text{iff} & \exists v_0, v_1, \ldots, v_n, \cdots \in M : v = v_0 \\ (v', v) \in R & \text{iff} & \exists v_0, v_1, \ldots, v_n, \cdots \in M, \\ & & \exists k \geq 1 : (v', v) = (v_{k-1}, v_k) \end{array}$$

$\Phi^\dagger$ is monotonic and we have

$$\Phi \circ \Phi^\dagger \supseteq Id \quad \text{and} \quad \Phi^\dagger \circ \Phi \subseteq Id \tag{31}$$

expressing that we have a Galois connection $(\Phi^\dagger, \Phi)$ from $(\mathcal{M}, \subseteq)$ to $(\mathcal{N}, \subseteq)$, see (5). In addition, we have

$$\begin{array}{rcl} \Phi(I_1 \cap I_2, R_1 \cap R_2) & = & \Phi(I_1, R_1) \cap \Phi(I_2, R_2) \\ \Phi^\dagger(M_1 \cap M_2) & = & \Phi^\dagger(M_1) \cap \Phi^\dagger(M_2) \end{array} \tag{32}$$

So far we have established a Galois connection relating sets of behaviors to pairs of initial condition and transition relation.

The next step of our Abstract Interpretation framework is a Galois connection for pairs of initial condition and transition relation. We thus assume two underlying sets $V_\mathbf{c}$ and $V_\mathbf{a}$ of concrete and abstract variables with domains $D_\mathbf{c} = \prod_{x_\mathbf{c} \in V_\mathbf{c}} D_{x_\mathbf{c}}$ and $D_\mathbf{a} = \prod_{x_\mathbf{a} \in V_\mathbf{a}} D_{x_\mathbf{a}}$. Initial conditions are subsets $I_\mathbf{c} \subseteq D_\mathbf{c}$ and $I_\mathbf{a} \subseteq D_\mathbf{a}$. Transition relations are subsets of $R_\mathbf{c} \subseteq D_\mathbf{c} \times D_\mathbf{c}$ and $R_\mathbf{a} \subseteq D_\mathbf{a} \times D_\mathbf{a}$. The resulting concrete and abstract domains are $\mathcal{N}_\mathbf{c}$ and $\mathcal{N}_\mathbf{a}$ obtained by applying formula (27).

Regarding preciseness orders $\sqsubseteq$ used in Abstract Interpretation frameworks, we equip $2^{D_\mathbf{c}}$, $2^{D_\mathbf{a}}$, $2^{D_\mathbf{c} \times D_\mathbf{c}}$, and $2^{D_\mathbf{a} \times D_\mathbf{a}}$, with the inclusion order and we assume two Galois connections, for the transition relations and initial conditions, respectively:

$$\begin{array}{rcl} (\alpha, \gamma) & : & \text{from } (2^{D_\mathbf{c} \times D_\mathbf{c}}, \subseteq) \text{ to } (2^{D_\mathbf{a} \times D_\mathbf{a}}, \subseteq) \\ ((\alpha^I, \gamma^I) & : & \text{from } (2^{D_\mathbf{c}}, \subseteq) \text{ to } (2^{D_\mathbf{a}}, \subseteq) \end{array} \tag{33}$$

Using (29), any pair (α^I, α) of maps

$$\alpha^I : 2^{D_\mathbf{c}} \to 2^{D_\mathbf{a}} \quad \text{and} \quad \alpha : 2^{D_\mathbf{c} \times D_\mathbf{c}} \to 2^{D_\mathbf{a} \times D_\mathbf{a}}$$

induces a unique map $\widetilde{\alpha} : \mathcal{N}_\mathbf{c} \to \mathcal{N}_\mathbf{a}$ and any pair (γ^I, γ) of maps

$$\gamma^I : 2^{D_\mathbf{a}} \to 2^{D_\mathbf{c}} \quad \text{and} \quad \gamma : 2^{D_\mathbf{a} \times D_\mathbf{a}} \to 2^{D_\mathbf{c} \times D_\mathbf{c}}$$

induces a unique map $\widetilde{\gamma} : \mathcal{N}_\mathbf{a} \to \mathcal{N}_\mathbf{c}$. Let $\mathcal{M}_\mathbf{c}$ and $\mathcal{M}_\mathbf{a}$ be the domains of concrete and abstract components obtained by applying formula (28). Formulas (33) give raise to a pair of maps

$$\begin{array}{rcl} \langle\alpha\rangle & : & \mathcal{M}_\mathbf{c} \to \mathcal{M}_\mathbf{a} \\ \langle\alpha\rangle & = & \Phi_\mathbf{a} \circ \widetilde{\alpha} \circ \Phi_\mathbf{c}^\dagger \end{array} \qquad \begin{array}{rcl} \langle\gamma\rangle & : & \mathcal{M}_\mathbf{a} \to \mathcal{M}_\mathbf{c} \\ \langle\gamma\rangle & = & \Phi_\mathbf{c} \circ \widetilde{\gamma} \circ \Phi_\mathbf{a}^\dagger \end{array}$$

Finally, we equip the concrete and abstract component domains $\mathcal{M}_\mathbf{c}$ and $\mathcal{M}_\mathbf{a}$ with the order defined by set inclusion.

Lemma 2: The so defined pair $(\langle\alpha\rangle, \langle\gamma\rangle)$ is a Galois connection from $(\mathcal{M}_\mathbf{c}, \subseteq)$ to $(\mathcal{M}_\mathbf{a}, \subseteq)$.

Proof: Since Φ, $\Phi^\dagger$, and $\widetilde{\alpha}$ are monotonic, so is $\langle\alpha\rangle$, with the same reasoning for the concretization map $\langle\gamma\rangle$. We then need to prove, for any $M_\mathbf{c} \in \mathcal{M}_\mathbf{c}$ and $M_\mathbf{a} \in \mathcal{M}_\mathbf{a}$:

$$M_\mathbf{c} \subseteq \langle\gamma\rangle(M_\mathbf{a}) \iff \langle\alpha\rangle(M_\mathbf{c}) \subseteq M_\mathbf{a} \tag{34}$$

Since $\Phi^{-\dagger}$ is strictly monotonic, the left hand side of (34) is equivalent to $\Phi^\dagger(M_\mathbf{c}) \subseteq \Phi^\dagger(\langle\gamma\rangle(M_\mathbf{a}))$, which decomposes as $I_\mathbf{c} \subseteq \gamma^I(I_\mathbf{a})$ and $R_\mathbf{c} \subseteq \gamma(R_\mathbf{a})$. The latter is equivalent to $\alpha^I(I_\mathbf{c}) \subseteq I_\mathbf{a}$ and $\alpha(R_\mathbf{c}) \subseteq R_\mathbf{a}$. Since Φ is monotonic, applying Φ to both sides yields the right hand side of (34). We have thus proved $\Rightarrow$ in (34). The proof of $\Leftarrow$ goes the same way but in opposite direction. □

Lemma 3: If γ is multiplicative, then so is $\langle\gamma\rangle$.

Proof: For both abstract and concrete domains, component composition $\times$ is by intersection

- of sets of behaviors, for components specified as sets of behaviors, and
- of transition relations and initial conditions, for components specified that way.

We thus assume that $\gamma(R_{\mathbf{a},1} \cap R_{\mathbf{a},2}) = \gamma(R_{\mathbf{a},1}) \cap \gamma(R_{\mathbf{a},2})$ holds and similarly for initial conditions, thus ensuring that $\widetilde{\gamma}$ is multiplicative too. We then get

$$\begin{array}{rcl}
\langle\gamma\rangle(M_{\mathbf{a},1} \cap M_{\mathbf{a},2}) & = & \Phi_\mathbf{c} \circ \widetilde{\gamma} \circ \Phi^\dagger_\mathbf{a}(M_{\mathbf{a},1} \cap M_{\mathbf{a},2}) \\
\text{(by (32))} & = & \Phi_\mathbf{c} \circ \widetilde{\gamma}\left(\Phi^\dagger_\mathbf{a}(M_{\mathbf{a},1}) \cap \Phi^\dagger_\mathbf{a}(M_{\mathbf{a},2})\right) \\
(\widetilde{\gamma} \text{ multiplicative}) & = & \Phi_\mathbf{c}\left(\widetilde{\gamma}\left(\Phi^\dagger_\mathbf{a}(M_{\mathbf{a},1})\right) \cap \widetilde{\gamma}\left(\Phi^\dagger_\mathbf{a}(M_{\mathbf{a},2})\right)\right) \\
\text{(by (32))} & = & \Phi_\mathbf{c} \circ \widetilde{\gamma} \circ \Phi^\dagger_\mathbf{a}(M_{\mathbf{a},1}) \cap \\
& & \Phi_\mathbf{c} \circ \widetilde{\gamma} \circ \Phi^\dagger_\mathbf{a}(M_{\mathbf{a},2}) \\
& = & \langle\gamma\rangle(M_{\mathbf{a},1}) \cap \langle\gamma\rangle(M_{\mathbf{a},2})
\end{array}$$

This proves the lemma. □

It remains to provide instances of Galois connections (33). We do this in the next section by using the classical predicate abstraction.

C. Using predicate abstraction

Consider the concrete domain $\mathcal{N}_\mathbf{c}$ defined in (27). A *predicate* over $\mathcal{N}_\mathbf{c}$ is an element $P_\mathbf{c} = (P^I_\mathbf{c}, P^R_\mathbf{c}) \in \mathcal{N}_\mathbf{c}$; its usual interpretation is the map $\mathcal{N}_\mathbf{c} \ni X_\mathbf{c} \to b_{P_\mathbf{c}}(X_\mathbf{c}) \in \mathbf{Bool} =_{\text{def}} \{ff, tt\}$ where $b(X_\mathbf{c}) = tt$ if and only if $X_\mathbf{c} \subseteq P_\mathbf{c}$. Equip $\mathbf{Bool}$ with the order $tt < ff$ and the product $\mathbf{Bool}^I$ with the product order denoted by $\leq$. For example, for the case of a single real variable x, the interval $[1, 2]$ for the initial condition over this variable satisfies the predicate $x > 0$.

Fix a finite set $\mathbf{I}$. A *predicate abstraction* is a finite tuple of predicates

$$\alpha = (P_{\mathbf{c},i})_{i \in \mathbf{I}} \;:\; \mathcal{N}_\mathbf{c} \to \mathbf{Bool}^\mathbf{I} =_{\text{def}} \mathcal{N}_\mathbf{a} \tag{35}$$

where $\mathbf{I}$ indexes the finite set of predicates. For $X_\mathbf{c} \in \mathcal{N}_\mathbf{c}$, $\alpha(X_\mathbf{c})$ returns the values for the $\mathbf{I}$ predicates, when tested over $X_\mathbf{c} \in \mathcal{N}_\mathbf{c}$. The coordinates of $X_\mathbf{a} \in \mathcal{N}_\mathbf{a}$ are denoted by $X^i_\mathbf{a}$. We equip the abstract domain $\mathcal{N}_\mathbf{a}$ with the order $\sqsubseteq_\mathbf{a} =_{\text{def}} \leq$. On the other hand, we equip $2^{D_\mathbf{c}}$ and $2^{D_\mathbf{c} \times D_\mathbf{c}}$ with set inclusion and $\mathcal{N}_\mathbf{c}$ with the resulting product order $\sqsubseteq_\mathbf{c}$. Abstraction $\alpha : (\mathcal{N}_\mathbf{c}, \sqsubseteq_\mathbf{c}) \to (\mathcal{N}_\mathbf{a}, \sqsubseteq_\mathbf{a})$ is then increasing.

Lemma 4: Abstraction α defined in (35) is a complete $\sqcup_\mathbf{c}$-morphism.

Proof: Select a set $\mathbf{X_c} \subset \mathcal{N}_\mathbf{c}$ of pairs $X_\mathbf{c} = (I, R)$ consisting of a concrete initial condition and a concrete transition relation. We have, for $i \in I$:

$$\begin{array}{rcl}
& & b_{P_{\mathbf{c},i}}(\sqcup_{\mathbf{X_c}}) = tt \\
\iff & & \left\{ \begin{array}{rcl} \bigcup_{X_\mathbf{c}=(I,R) \in \mathbf{X_c}} I & \subseteq & P^I_{\mathbf{c},i} \\ \bigcup_{X_\mathbf{c}=(I,R) \in \mathbf{X_c}} R & \subseteq & P^R_{\mathbf{c},i} \end{array} \right. \\
\iff & & \left\{ \begin{array}{rcl} \forall X_\mathbf{c} = (I,R) \in \mathbf{X_c} & . & I \subseteq P^I_{\mathbf{c},i} \\ \forall X_\mathbf{c} = (I,R) \in \mathbf{X_c} & . & R \subseteq P^R_{\mathbf{c},i} \end{array} \right. \\
\iff & & \forall X_\mathbf{c} \in \mathbf{X_c} \;\; . \;\; X_\mathbf{c} \sqsubseteq_\mathbf{c} P_{\mathbf{c},i} \\
\iff & & \forall X_\mathbf{c} \in \mathbf{X_c} \;\; . \;\; b_{P_{\mathbf{c},i}}(X_\mathbf{c}) = tt \\
\iff & & \left[\sqcup_{X_\mathbf{c} \in \mathbf{X_c}} b_{P_{\mathbf{c},i}}(X_\mathbf{c})\right] = tt
\end{array}$$

where the first occurrence of $\sqcup$ refers to $\mathcal{N}_\mathbf{c}$ whereas the second occurrence refers to $\mathcal{N}_\mathbf{a}$. □

By Theorem 1, the formula

$$\begin{array}{rcl}
\gamma(X_\mathbf{a}) & = & \sqcup_\mathbf{c}\{X_\mathbf{c} \mid \alpha(X_\mathbf{c}) \sqsubseteq_\mathbf{a} X_\mathbf{a}\} \\
& = & \bigcup \{X_\mathbf{c} \mid \forall i \in I.\; b_{P_{\mathbf{c},i}}(X_\mathbf{c}) \leq X^i_\mathbf{a}\} \\
& = & \bigcap_{X^i_\mathbf{a} = tt} P_{\mathbf{c},i}
\end{array} \tag{36}$$

defines γ such that (α, γ) is a Galois connection.

Equip $\mathcal{N}_\mathbf{a}$ with the product $\times_\mathbf{a} =_{\text{def}} \bigwedge_\mathbf{a}$, where $\bigwedge_\mathbf{a}$ refers to the order $\sqsubseteq_\mathbf{a}$. On $\mathcal{N}_\mathbf{c}$, product is by intersection.

Lemma 5: γ is multiplicative.

Proof: The following equalities hold:

$$\begin{array}{rcl}
\gamma(X_{\mathbf{a},1} \times_\mathbf{a} X_{\mathbf{a},2}) & = & \bigcap_{X^i_{\mathbf{a},1} = tt \,\vee\, X^i_{\mathbf{a},2} = tt} P_{\mathbf{c},i} \\
& = & \left[\bigcap_{X^i_{\mathbf{a},1} = tt} P_{\mathbf{c},i}\right] \bigcap \left[\bigcap_{X^i_{\mathbf{a},2} = tt} P_{\mathbf{c},i}\right] \\
& = & \gamma(X_{\mathbf{a},1}) \times_\mathbf{c} \gamma(X_{\mathbf{a},2})
\end{array}$$

which prove the lemma. □

V. Obstructions to getting stronger results

A. Why not Galois connections for contracts?

To parallel (7), one could consider defining a contract concretization $\widehat{\gamma} : (\mathcal{X}^<_\mathbf{a}, \subseteq_\mathbf{a}) \to (\mathcal{X}^<_\mathbf{c}, \subseteq_\mathbf{c})$ by

$$\widehat{\gamma}(\chi_\mathbf{a}) = \alpha^{-1}(\chi_\mathbf{a}) \tag{37}$$

This definition is sound since α is monotonic. Unfortunately, the pair $(\widehat{\alpha}, \widehat{\gamma})$ is *not* a Galois connection in general. Characteristic property (5) of Galois connections would translate here as:

$$\begin{array}{rcl}
\chi_\mathbf{c} \subseteq_\mathbf{c} \widehat{\gamma} \circ \widehat{\alpha}(\chi_\mathbf{c}) & = & \{x_\mathbf{c} \mid \alpha(x_\mathbf{c}) \in \widehat{\alpha}(\chi_\mathbf{c})\} \\
& = & \{x_\mathbf{c} \mid \gamma \circ \alpha(x_\mathbf{c}) \in \chi_\mathbf{c}\}
\end{array}$$

which is equivalent to

$$x \in \chi_\mathbf{c} \implies \gamma \circ \alpha(x_\mathbf{c}) \in \chi_\mathbf{c} \tag{38}$$

Since (α, γ) is a Galois connection we have

$$x_{\mathbf{c}} \quad \sqsubseteq_{\mathbf{c}} \quad \gamma \circ \alpha(x_{\mathbf{c}}) \tag{39}$$

On the other hand, we know that $\chi_{\mathbf{c}}$ is $\sqsubseteq_{\mathbf{c}}$-closed, which, together with (39), implies the *opposite* of the wanted implication (38).

This observation suggests a counter-measure that we have developed in a previous attempt: to get the right direction for the implication in (38), consider two mirroring Galois connections instead of a single one. A *Galois mirror* from $(\mathcal{X}_{\mathbf{c}}, \sqsubseteq_{\mathbf{c}})$ to $(\mathcal{X}_{\mathbf{a}}, \sqsubseteq_{\mathbf{a}})$ consists of two mirroring Galois connections:

$$(\alpha, \gamma): \quad \text{from} \quad (\mathcal{X}_{\mathbf{c}}, \sqsubseteq_{\mathbf{c}}) \text{ to } (\mathcal{X}_{\mathbf{a}}, \sqsubseteq_{\mathbf{a}}) \tag{40}$$

$$(\alpha, \breve{\gamma}): \quad \text{from} \quad (\mathcal{X}_{\mathbf{c}}, \sqsupseteq_{\mathbf{c}}) \text{ to } (\mathcal{X}_{\mathbf{a}}, \sqsupseteq_{\mathbf{a}}) \tag{41}$$

Galois mirrors can be obtained by invoking Theorem 1: If $(\mathcal{X}_{\mathbf{c}}, \sqsubseteq_{\mathbf{c}})$ has LUBs and GLBs for arbitrary sets and α is both a complete $\sqcup_{\mathbf{c}}$-morphism and a complete $\sqcap_{\mathbf{c}}$-morphism, then the two concretizations γ and $\breve{\gamma}$ associated to α with respect to orders $\sqsubseteq_{\mathbf{c/a}}$ and $\sqsupseteq_{\mathbf{c/a}}$, respectively, define, together with α, a Galois mirror.

Pursueing our attempt, we can state the following: We assume a Galois mirror $((\alpha, \gamma), (\alpha, \breve{\gamma}))$ from $(\mathcal{X}_{\mathbf{c}}, \sqsubseteq_{\mathbf{c}})$ to $(\mathcal{X}_{\mathbf{a}}, \sqsubseteq_{\mathbf{a}})$. The following formulas define a Galois connection from $(\mathcal{X}_{\mathbf{c}}^{<}, \subseteq_{\mathbf{c}})$ to $(\mathcal{X}_{\mathbf{a}}^{<}, \subseteq_{\mathbf{a}})$—both formulas are well defined since $\breve{\gamma}$ and α are monotonic:

$$\begin{array}{rcl} \widehat{\alpha}(\chi_{\mathbf{c}}) & = & \breve{\gamma}^{-1}(\chi_{\mathbf{c}}) \\ \widehat{\gamma}(\chi_{\mathbf{a}}) & = & \alpha^{-1}(\chi_{\mathbf{a}}) \end{array} \tag{42}$$

The proof of (42) is elegant. Suppose first that both $\chi_{\mathbf{c}}$ and $\chi_{\mathbf{a}}$ possess unique maxima c and a, so that $\chi_{\mathbf{c}} = \{c' \mid c' \sqsubseteq_{\mathbf{c}} c\} =_{\text{def}} c^{\downarrow}$ and similarly for $\chi_{\mathbf{a}}$. Then we have

$$\begin{array}{rcccl} \widehat{\alpha}(\chi_{\mathbf{c}}) & = & \breve{\gamma}^{-1}(c^{\downarrow}) & = & \{a' \mid \breve{\gamma}(a') \sqsubseteq_{\mathbf{c}} c\} \\ & & & \overset{\text{by}(41)}{=} & \{a' \mid a' \sqsubseteq_{\mathbf{a}} \alpha(c)\} \\ \widehat{\gamma}(\chi_{\mathbf{a}}) & = & \alpha^{-1}(a^{\downarrow}) & = & \{c' \mid \alpha(c') \sqsubseteq_{\mathbf{a}} a\} \\ & & & \overset{\text{by}(40)}{=} & \{c' \mid c' \sqsubseteq_{\mathbf{c}} \gamma(a)\} \end{array}$$

Consequently, we have, for such a pair $(\chi_{\mathbf{c}}, \chi_{\mathbf{a}})$:

$$\begin{array}{rcl} \widehat{\alpha}(\chi_{\mathbf{c}}) \subseteq \chi_{\mathbf{a}} & \Leftrightarrow & \alpha(c) \sqsubseteq_{\mathbf{a}} a \\ \chi_{\mathbf{c}} \subseteq \widehat{\gamma}(\chi_{\mathbf{a}}) & \Leftrightarrow & c \sqsubseteq_{\mathbf{c}} \gamma(a) \end{array}$$

Invoking again (40) proves $\widehat{\alpha}(\chi_{\mathbf{c}}) \subseteq \chi_{\mathbf{a}} \Leftrightarrow \chi_{\mathbf{c}} \subseteq \widehat{\gamma}(\chi_{\mathbf{a}})$ in this particular case.

For the general case we have, since $\chi_{\mathbf{a}}$ and $\chi_{\mathbf{c}}$ are downward closed and $\widehat{\alpha}$ and $\widehat{\gamma}$ are both complete $\cup$-morphisms,

$$\begin{array}{l} \widehat{\alpha}(\chi_{\mathbf{c}}) = \widehat{\alpha}(\bigcup_{c \in \chi_{\mathbf{c}}} c^{\downarrow}) = \bigcup_{c \in \chi_{\mathbf{c}}} \widehat{\alpha}(c^{\downarrow}) \\ \widehat{\gamma}(\chi_{\mathbf{a}}) = \widehat{\gamma}(\bigcup_{a \in \chi_{\mathbf{a}}} a^{\downarrow}) = \bigcup_{a \in \chi_{\mathbf{a}}} \widehat{\gamma}(a^{\downarrow}) \end{array}$$

Thus,

$$\bigcup_{c \in \chi_{\mathbf{c}}} \widehat{\alpha}(c^{\downarrow}) = \widehat{\alpha}(\chi_{\mathbf{c}}) \subseteq A = \bigcup_{a \in \chi_{\mathbf{a}}} a^{\downarrow}$$

holds if and only if, for every $c \in \chi_{\mathbf{c}}$, $\widehat{\alpha}(c^{\downarrow}) \subseteq \bigcup_{a \in \chi_{\mathbf{a}}} a^{\downarrow}$. But $\widehat{\alpha}(c^{\downarrow}) = \{a' \mid a' \sqsubseteq_{\mathbf{c}} \alpha(c)\}$, hence there exists some $a \in \chi_{\mathbf{a}}$ such that $\widehat{\alpha}(c^{\downarrow}) \subseteq a^{\downarrow}$, which implies $c^{\downarrow} \subseteq \widehat{\gamma}(a^{\downarrow})$ by the first part of this proof. Since this holds for every $c \in \chi_{\mathbf{c}}$, we derive $\chi_{\mathbf{c}} = \bigcup_{c \in \chi_{\mathbf{c}}} c^{\downarrow} \subseteq \bigcup_{a \in \chi_{\mathbf{a}}} \widehat{\gamma}(a^{\downarrow}) = \widehat{\gamma}(\bigcup_{a \in \chi_{\mathbf{a}}} a^{\downarrow}) = \widehat{\gamma}(\chi_{\mathbf{a}})$.

Vice-versa

$$\bigcup_{c \in \chi_{\mathbf{c}}} c^{\downarrow} = \chi_{\mathbf{c}} \subseteq \widehat{\gamma}(\chi_{\mathbf{a}}) = \bigcup_{a \in \chi_{\mathbf{a}}} \widehat{\gamma}(a^{\downarrow})$$

holds if and only if, for every $c \in \chi_{\mathbf{c}}$, $c^{\downarrow} \subseteq \bigcup_{a \in \chi_{\mathbf{a}}} \widehat{\gamma}(a^{\downarrow})$. By the same argument as before, this is equivalent to the existence of some $a \in \chi_{\mathbf{a}}$ such that $c^{\downarrow} \subseteq \widehat{\gamma}(a^{\downarrow})$, which implies $\widehat{\alpha}(c^{\downarrow}) \subseteq a^{\downarrow}$. Since this holds for every $c \in \chi_{\mathbf{c}}$, we derive $\widehat{\alpha}(\chi_{\mathbf{c}}) = \widehat{\alpha}(\bigcup_{c \in \chi_{\mathbf{c}}} c^{\downarrow}) = \bigcup_{c \in \chi_{\mathbf{c}}} \widehat{\alpha}(c^{\downarrow}) \subseteq \bigcup_{a \in \chi_{\mathbf{a}}} a^{\downarrow} = \chi_{\mathbf{a}}$. This proves (42).

Having (42), we can derive a Galois connection for contracts, from $(\mathbb{C}_{\mathbf{c}}, \sqsubseteq_{\mathbf{c}})$ to $(\mathbb{C}_{\mathbf{a}}, \sqsubseteq_{\mathbf{a}})$, where the two concrete and abstract preciseness orders are the product of inclusion orders $\sqsubseteq_{\mathbf{c/a}} =_{\text{def}} \subseteq_{\mathbf{c/a}} \times \subseteq_{\mathbf{c/a}}$ on $\mathbb{C}_{\mathbf{c/a}}$, whose semantic domain is $\mathcal{M}_{\mathbf{c/a}} \times \mathcal{M}_{\mathbf{c/a}}$:

$$\begin{array}{rcl} \alpha(\mathcal{C}_{\mathbf{c}}) & =_{\text{def}} & \langle \widehat{\alpha}\left(\mathcal{C}_{\mathbf{c}}^{\mathsf{env}}\right), \widehat{\alpha}\left(\mathcal{C}_{\mathbf{c}}^{\mathsf{imp}}\right) \rangle \\ & = & \langle \breve{\gamma}^{-1}\left(\mathcal{C}_{\mathbf{c}}^{\mathsf{env}}\right), \breve{\gamma}^{-1}\left(\mathcal{C}_{\mathbf{c}}^{\mathsf{imp}}\right) \rangle \\ \gamma(\mathcal{C}_{\mathbf{a}}) & =_{\text{def}} & \langle \widehat{\gamma}\left(\mathcal{C}_{\mathbf{a}}^{\mathsf{env}}\right), \widehat{\gamma}\left(\mathcal{C}_{\mathbf{a}}^{\mathsf{imp}}\right) \rangle \\ & = & \langle \alpha^{-1}\left(\mathcal{C}_{\mathbf{a}}^{\mathsf{env}}\right), \alpha^{-1}\left(\mathcal{C}_{\mathbf{a}}^{\mathsf{imp}}\right) \rangle \end{array}$$

Failure of the above approach:

1) Assumption 3 is needed because $\gamma \circ \alpha$ dominates Id for the preciseness order, but is not equal to Id. This assumption allows accomodating the inequality $M_{\mathbf{c}} \sqsubseteq_{\mathbf{c}} \gamma \circ \alpha(M_{\mathbf{c}})$. Assumption 3 cannot be reverted—with upward-closed replacing downward-closed. Reverting it would yield an assumption valid for no practical framework of contracts. To conclude on this point, no flexibility can be expected regarding this Assumption.
2) No variation of (40,41) seems able to imply (42); we have tried all combinations of orders on components and system of Galois connections. Only the above one worked.
3) All of the reasoning developed in this section looks nice but is flawed since the very beginning! Galois mirrors do not seem to exist! Just because it seems impossible to find nontrivial abstractions that are both $\sqcap_{\mathbf{c}}$- and $\sqcup_{\mathbf{c}}$-complete! (Try…)

B. Can we expect better results for the special case of A/G-contracts?

Getting a Galois connection for a pair of assumption and guarantee fails for the very same reasons we discussed in Section V-A. In fact, the refinement order is the product $\preceq = \supseteq \times \subseteq$ and getting a Galois connection for it would again require a Galois mirror…which does not exist.

Regarding Theorem 4, its statement 2) cannot be improved in that equality still does not hold in (12) in general. Indeed, the reasoning developed in the analysis of statement 3) cannot be circumvent for the special case of A/G-contracts.

VI. Conclusion

We have presented a systematic way of lifting an Abstract Interpretation framework (in the form of a Galois connection) on contracts, to an abstraction on contracts. This provides a machinery of abstraction for contracts with (nearly all)

compositionality properties, thus matching the rich compositionality apparatus of contracts in designing systems. Our abstraction provides positive semi-decision procedures for handling "non-computable" contracts (e.g., involving data of infinite domains). By "positive" we mean that properties of interest can be *proved*. In contrast, test-oriented techniques such as observers for contracts [6] offer negative semi-decision procedures by *disproving* such properties. Abstractions and observers thus complement each other for the analysis of undecidable contract frameworks.

REFERENCES

[1] Martín Abadi and Leslie Lamport. Conjoining specifications. *ACM Trans. Program. Lang. Syst.*, 17(3):507–534, 1995.

[2] Luca De Alfaro and Thomas A. Henzinger. Interface-based design. In *In Engineering Theories of Software Intensive Systems, proceedings of the Marktoberdorf Summer School*. Kluwer, 2004.

[3] Sebastian S. Bauer, Rolf Hennicker, and Martin Wirsing. Interface theories for concurrency and data. *Theor. Comput. Sci.*, 412(28):3101–3121, 2011.

[4] Sebastian S. Bauer, Kim Guldstrand Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. A modal specification theory for components with data. In Farhad Arbab and Peter Csaba Ölveczky, editors, *FACS*, volume 7253 of *Lecture Notes in Computer Science*, pages 61–78. Springer, 2011.

[5] Albert Benveniste, Benoît Caillaud, Alberto Ferrari, Leonardo Mangeruca, Roberto Passerone, and Christos Sofronis. Multiple viewpoint contract-based specification and design. In *Proceedings of the Software Technology Concertation on Formal Methods for Components and Objects, FMCO'07*, volume 5382 of *Lecture Notes in Computer Science*, pages 200–225. Springer, October 2008.

[6] Albert Benveniste, Benoit Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Thomas Henzinger, and Kim G. Larsen. Contracts for System Design. Rapport de recherche RR-8147, INRIA, November 2012. http://hal.inria.fr/hal-00757488/PDF/RR-8147.pdf.

[7] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, Marcin Jurdzinski, and Freddy Y. C. Mang. Interface compatibility checking for software modules. In *CAV*, pages 428–441, 2002.

[8] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. Synchronous and Bidirectional Component Interfaces. In *Proc. of the 14th International Conference on Computer Aided Verification (CAV'02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 414–427. Springer, 2002.

[9] Edmund M. Clarke, David E. Long, and Kenneth L. McMillan. Compositional model checking. In *LICS*, pages 353–362, 1989.

[10] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[11] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *J. Log. Program.*, 13(2&3):103–179, 1992.

[12] Patrick Cousot and Radhia Cousot. Abstract Interpretation Frameworks. *J. Log. Comput.*, 2(4):511–547, 1992.

[13] Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. The astreé analyzer. In *ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2005.

[14] Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, and Xavier Rival. Why does astrée scale up? *Formal Methods in System Design*, 35(3):229–264, 2009.

[15] Alexandre David, Kim. G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Timed I/O automata : A complete specification theory for real-time systems. In *Proc. of the 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC'10)*, pages 91–100. ACM, 2010.

[16] Luca de Alfaro, Leandro Dias da Silva, Marco Faella, Axel Legay, Pritam Roy, and Maria Sorea. Sociable Interfaces. In *Proc. of the 5th International Workshop on Frontiers of Combining Systems (FroCos'05)*, volume 3717 of *Lecture Notes in Computer Science*, pages 81–105. Springer, 2005.

[17] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*, pages 109–120. ACM Press, 2001.

[18] Benoît Delahaye, Benoît Caillaud, and Axel Legay. Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects. *Formal Methods in System Design*, 38(1):1–32, 2011.

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

[20] Orna Grumberg and David E. Long. Model checking and modular verification. *ACM Trans. Program. Lang. Syst.*, 16(3):843–871, 1994.

[21] Orna Kupferman and Moshe Y. Vardi. Modular model checking. In Willem P. de Roever, Hans Langmaack, and Amir Pnueli, editors, *COMPOS*, volume 1536 of *Lecture Notes in Computer Science*, pages 381–401. Springer, 1997.

[22] Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski. Modal i/o automata for interface and product line theories. In *Proc. of the 16th European Symposium on Programming (ESOP'07)*, volume 4421 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2007.

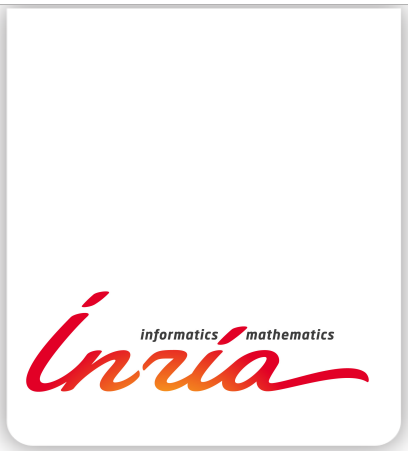
[23] Antoine Miné. Weakly Relational Numerical Abstract Domains. Phd, Ecole Normale Supérieure, département d'informatique, Dec 2004. http://www.di.ens.fr/~mine/these/these-color.pdf.

[24] Antoine Miné. Static analysis of run-time errors in embedded critical parallel c programs. In Gilles Barthe, editor, *ESOP*, volume 6602 of *Lecture Notes in Computer Science*, pages 398–418. Springer, 2011.

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

[26] Eugene W. Stark. A proof technique for rely/guarantee properties. In S. N. Maheshwari, editor, *FSTTCS*, volume 206 of *Lecture Notes in Computer Science*, pages 369–391. Springer, 1985.

[27] Reinhard Wilhelm and Björn Wachter. Abstract interpretation with applications to timing validation. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 22–36. Springer, 2008.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399