Complex software systems: Formalization and Applications

Marc Aiguier, Pascale Le Gall and Mbarka Mabrouki
École Centrale Paris
Laboratoire de Mathématiques Appliquées aux Systèmes (MAS)
Grande Voie des Vignes - F-92295 Châtenay-Malabry

Programme d’Épigénomique
523, Place des Terrasses de l’Agora - F-91025 Evry
{marc.aiguier, pascale.legall}@ecp.fr, mabrouki@epigenomique.genopole.fr

Abstract

A mathematical denotation is proposed for the notion of complex software systems whose behavior is specified by rigorous formalisms. Complex systems are described in a recursive way as an interconnection of subsystems by means of architectural connectors. In order to consider the largest family of specification formalisms and architectural connectors, this denotation is essentially formalism, specification and connector independent. For this, we build our denotation on Goguen’s institution theory. In this abstract framework, we characterize complexity by the notion of property emergence.

This work is a revised and extended version of Aiguier, Le Gall and Mabrouki (3rd International Conference on Software Engineering Advanced (ICSEA), IEEE Computer Society Press, 2008).

Keywords- abstract specification language; abstract architectural connector; emergent property; institution; category theory; transition systems; modal first-order logic.

1 Introduction

A powerful approach to develop large software systems is to describe them in a recursive way as an interconnection of sub-systems. This has then made emerge the notion of architectural connector as a powerful tool to describe systems in terms of components and their interactions [6, 7, 16, 25]. Academic and industrial groups have defined and developed computer languages dedicated to the description of software architectures provided with architectural connectors, called Architectural Description Language (ADL), such as ACME/ADML [17], Wright [5] or Community [15, 16]. The interest of describing software systems as interconnected subsystems is that this promotes the reuse of components either directly taken in a library or adapted by slight modifications made on existing ones. The well-known difficulty with such systems is to infer the global behavior of the system from the ones of subsystems. Indeed, modern software systems are often open on the outside, that is they interact with the environment, composed of interacting subsystems (e.g. active objects which interact together concurrently [3, 27]) or defined by questioning requirements of subsystems (e.g. feature-oriented systems where each feature can modify the expected properties of pre-existing features [18, 4, 26]). Thus, such global systems may exhibit behaviors, that cannot be anticipated just from a complete knowledge of subsystems. Hence, what makes such software systems complex is they cannot be reduced to simple rules of property inference from subsystems towards the global system.

Following some works issued from other scientific disciplines such as biology, physics, economy or sociology [10, 13], let us make more precise what we mean by complex systems. A complex system is characterized by a holistic behavior, i.e. global: we do not consider that its behavior results from the combination of isolated behaviors of some of its components, but instead has to be considered as a whole. This is expressed by the appearance (emergence) of global properties which is very difficult, see impossible, to anticipate just from a complete knowledge of component behaviors. This notion of emergence seems to be the simplest way to define complexity. Succinctly, this could be expressed as follows: suppose a system XY composed of two subsystems X and Y. Let us also suppose we have a mathematical function F which gives all potential pieces of information on XY, X and Y, and an operation ‘+’ to combine potential pieces of information of subsystems. If we have that $F(XY) = F(X) + F(Y)$ then this means that
the system $XY$ integrates in a consistent manner the subsystems $X$ and $Y$ without either removing or adding pieces of information. Therefore, we can say that the system $XY$ is not complex (i.e. the system $XY$ is said to be modular). On the contrary, if there exists some $a \in F(X) + F(Y)$ such that $a \notin F(XY)$ or there exists some $a \in F(XY)$ such that $a \notin F(X) + F(Y)$, then there is reconsideration of some potential pieces of information of $X$ or $Y$ in the first case, and appearance of true emergence in the second case. The system $XY$ is then said complex.

In this paper, we will study the notion of complex software systems by using formal specifications, that is we will suppose that every part of systems have been specified in a given formalism from which we can infer properties. The system $XY$ will be built from subsystems $X$ and $Y$ by means of an architectural connector $c$ expliciting how the two subsystems are linked together to form the global system $c(X,Y) = XY$, the connector $c$ being implicit in the notation $XY$. Finally, the function $F$ will give for a specification its whole set of satisfied properties, the so-called semantic consequences of specifications usually noted $X^\ast$, and $F(X) + F(Y) = (X^\ast \cup Y^\ast)^\ast$. Roughly speaking, this last notation consists in saturating the property derivation mechanism, and then represents the fact that $F(X) + F(Y)$ denote the set of all properties which can be derived from the set of properties $X^\ast$, resp. $Y^\ast$, associated to $X$, resp. $Y$. The notion of complexity being based on the emergence of properties, a general framework dedicated to complex software systems can be defined independently of formalisms, specifications and architectural connectors. Hence, we investigate an abstract form of complexity, by following the paradigm “logical-system independency”. The interest here is simple. We can observe, whatever the formalism used to specify softwares, that the same set of notions underlies complexity. These notions are: architectural connector and emergent property. To formalize abstractly these elements, our approach will be based on previous works:

- we will use the general framework of institutions [20] which is recognized as well-adapted to generalize formalisms. The theory of institutions abstracts the semantical part of logical systems according to the needs of software specifications in which changes of signatures are taken into account. The abstraction of the different parts of logical systems is obtained by using some notions of the category theory such as the category of signatures and the two functors to denote respectively the set of sentences and the category of models over a signature (see Section 2 for the complete definition of institutions and some related notions);

- specifications will be defined following the generic approach of specification logics [14]. The interest of specification logics is they unify in the same framework heterogeneous forms of specifications by considering them as simple objects of a category $SPEC$, while handled specifications over institutions are mostly axiomatic (i.e. of the form $(\Sigma, Ax)$ where $\Sigma$ is a signature and $Ax$ is a (finite) set of formulas (axioms) over $\Sigma$). However, because we are interested by emergent properties, we will adapt/modify specification logics by defining them over institutions in order to focus on specification properties;

- abstract connectors will be defined by using notions of the category theory. The use of category theory has already been applied strikingly to model the architecture of software systems by Goguen [19] and Fiadeiro & al. [15]. It has also been applied to model complex natural systems such as biological, physical and social systems (e.g. Ehresman and Vannbreuwer’s works [13]).

Fiadeiro & al. [16] have proposed an abstract formal denotation of a class of architectural connectors in the style of Allen and Garlan [6], that is defined by a set of roles and a glue specification. Here, we will go beyond by not supposing any structure in the architectural connectors.

Over our abstract notions of specification and architectural connector, we will define the notion of emergent properties according to the two following classes:

1. the ones we will call true emergent properties that are properties which cannot be inferred from subsystem properties,

2. and the ones we will call non conformity properties that are subsystem properties which are not satisfied by the global system anymore.

In practice, properties of the first form, i.e. true emergent properties, combine knowledge inherited from subsystems. Thus, they are defined in a richer language than the ones associated to each subsystem, and the presence of such emergent properties is quite natural. Conversely, properties of the second form, i.e. non conformity properties, are often the consequences of bad interactions between subsystems. They characterize properties that are satisfied (resp. not satisfied) by a subsystem considered in isolation, but are not satisfied (resp. satisfied) by the global system incorporating the subsystem in question.

A software system will be then said complex when emergent properties can be inferred from it. The complexity of systems just means that we do not benefit from the complete knowledge of subsystems we have, to analyze the behavior of the large system. Hence, the recursive approach used to describe the system cannot be used to analyze its behavior. Complex systems can then be opposed to modular systems.
which by definition strictly preserve local properties at the global level (see [24] for a state-of-the-art on the modular approach).

The formalizations of system complexity and emergent properties are interesting if they are done in such way to support the characterization of general properties to guarantee when a system is or is not complex. To answer this point, we will give some conditions under which a system is modular. We will then establish two results: in the first one we will give a sufficient and necessary condition to ensure the absence of true emergent properties. In the second result, we will give sufficient conditions based on the categorical notion of adjunctness to ensure the absence of non-conformity properties.

As a result of our generalization defined in this paper, all the notions, results, and techniques established and defined in our abstract framework are de facto adaptable to any institution.

The paper is structured as follows: Section 2 reviews some concepts, notations and terminology about institutions. Section 3 defines an abstract notion of specifications over institutions. In Section 4, abstract architectural connectors are defined and classified as complex and modular. The notations of the category theory used in this paper are the standard ones and can be found in [15]. Although all the notions and results given in this manuscript are exemplified by many examples all along the paper, Section 5 illustrates more particularly the abstract framework developed in this paper to reactive component-based systems described by transition systems and combined together through the synchronous product operation.

Note: This manuscript extends the paper published in the proceedings of [1] with expanded definitions, new results and additional examples. Moreover, as an application of our approach, we will study reactive systems described by means of transition systems as components and of the usual synchronous product as architectural connector, and whose behavior is expressed by logical properties over a modal first-order logic. In this framework, we propose to study complexity of reactive systems through this notion of emergent properties. We will also give some conditions to guarantee when a system is lacking of non-conformity properties which have been recognized as being the cause of bad interaction between components. This last work has been published in the proceedings of [2]. Here, this manuscript also extends the paper published in [2] with complete proofs of the main results.

2 Institutions

The theory of institutions [20] is a categorical abstract model theory which formalizes the intuitive notion of logical system, including syntax, semantics, and the satisfaction between them. This emerged in computing science studies of software specification and semantics, in the context of the increasing number of considered logics, with the ambition of doing as much as possible at the level of abstraction independent of commitment to any particular logic. Now institutions have become a common tool in the area of formal specification, in fact its most fundamental mathematical structure.

2.1 Basic definitions

Definition. 1 (Institution) An institution \( I = (\text{Sig}, \text{Sen}, \text{Mod}, \vdash) \) consists of

- a category Sig, objects of which are called signatures,
- a functor Sen : Sig \( \rightarrow \) Set giving for each signature a set, elements of which are called sentences,
- a contravariant functor Mod : Sig\(^{op}\) \( \rightarrow \) Cat giving for each signature a category, objects and arrows of which are called \( \Sigma \)-models and \( \Sigma \)-morphisms respectively, and
- a \(|\text{Sig}|\)-indexed family of relations
  \( \vdash_{\Sigma} \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma) \)

called satisfaction relation,

such that the following property holds:

\[ \forall \sigma : \Sigma \rightarrow \Sigma', \forall M' \in |\text{Mod}(\Sigma')|, \forall \phi \in \text{Sen}(\Sigma), \]

\[ M' \vdash_{\Sigma'} \text{Sen}(\sigma)(\phi) \iff \text{Mod}(\sigma)(M') \vdash_{\Sigma} \phi \]

Here, we define some notions over institutions which will be useful thereafter.

Definition. 2 (Elementary equivalence) Let \( I = (\text{Sig}, \text{Sen}, \text{Mod}, \vdash) \) be an institution. Let \( \Sigma \) be a signature. Two \( \Sigma \)-models \( M_1 \) and \( M_2 \) are elementary equivalent, noted \( M_1 \equiv_{\Sigma} M_2 \) if, and only if the following condition holds: \( \forall \phi \in \text{Sen}(\Sigma), M_1 \vdash_{\Sigma} \phi \iff M_2 \vdash_{\Sigma} \phi \).

This means that \( M_1 \) and \( M_2 \) are undistinguishable with respect to the formula satisfaction.

Definition. 3 (Closed under isomorphism) An institution is closed under isomorphism if, and only if every two isomorphic models are elementary equivalent.

All reasonable logics (anyway all the logics classically used in mathematics and computer science) are closed under isomorphism.
Definition. 4 (Logical theory) Let $\mathcal{I} = (\text{Sig}, \text{Sen}, \text{Mod}, \models)$ be an institution. Let $\Sigma$ be a signature of $|\text{Sig}|$. Let $T$ be a set of $\Sigma$-sentences (i.e. $T \subseteq \text{Sen}(\Sigma)$). Let us denote $\text{Mod}(T)$ the full sub-category of $\text{Mod}(\Sigma)$ whose objects are all $\Sigma$-models $\mathcal{M}$ such that for any $\varphi \in T$, $\mathcal{M} \models_{\Sigma} \varphi$, and $T^*$ the subset of $\text{Sen}(\Sigma)$, so-called semantic consequences of $T$, defined as follows:

$$T^* = \{ \varphi \mid \forall \mathcal{M} \in |\text{Mod}(T)|, \mathcal{M} \models_{\Sigma} \varphi \}$$

$T$ is a logical theory if, and only if $T = T^*$.

$\varphi \in T^*$ is also denoted by $\models_{\Sigma} \varphi$.

2.2 Examples of institutions

2.2.1 Propositional Logic (PL)

Signatures and signature morphisms are sets of propositional variables and functions between them respectively. Given a signature $\Sigma$, the set of $\Sigma$-sentences is the least set of sentences finitely built over propositional variables in $\Sigma$ and Boolean connectives in $\{\neg, \lor, \land, \Rightarrow\}$. Given a signature morphism $\sigma : \Sigma \to \Sigma'$ associating to each propositional variable of $\Sigma$ a propositional variable of $\Sigma'$, $\text{Sen}(\sigma)$ translates $\Sigma$-formulas to $\Sigma'$-formulas by renaming propositional variables according to $\sigma$.

Given a signature $\Sigma$, the category of $\Sigma$-models is the category of mappings $\nu : \Sigma \to \{0,1\}$ with identities as morphisms. Given a signature morphism $\sigma : \Sigma \to \Sigma'$, the forgetful functor $\text{Mod}(\sigma)$ maps a $\Sigma'$-model $\nu'$ to the $\Sigma$-model $\nu = \nu' \circ \sigma$.

Finally, satisfaction is the usual propositional satisfaction.

2.2.2 Many-sorted First Order Logic with equality (FOL)

Signatures are triples $(S, F, P)$ where $S$ is a set of sorts, and $F$ and $P$ are sets of function and predicate names respectively, both with arities in $S^* \times S$. Signature morphisms $\sigma : (S, F, P) \to (S', F', P')$ consist of three functions between sets of sorts, sets of functions and sets of predicates respectively, the last two preserving arities. Given a signature $\Sigma = (S, F, P)$, the $\Sigma$-atoms are of two possible forms: $t_1 = t_2$ where $\neg t_1, t_2 \in T_F(X)_s$ $(s \in S)$, and $\nu(t_1, \ldots, t_n)$ where $\nu$ denotes the empty sequence.

Given a signature $\Sigma = (S, F, P)$, the $\Sigma$-sentences is the least set of formulas built over the set of $\Sigma$-atoms by finitely applying Boolean connectives in $\{\neg, \lor, \land, \Rightarrow\}$ and quantifiers $\forall$ and $\exists$.

Given a signature $\Sigma = (S, F, P)$, a $\Sigma$-model $\mathcal{M}$ is a family $M = \{M_s\}_{s \in S}$ of sets. Fix each one equipped with a function $f^\mathcal{M} : M_{s_1} \times \ldots \times M_{s_n} \to M_t$ for every $f : s_1 \times \ldots \times s_n \to s \in F$ and with a n-ary relation $p^\mathcal{M} \subseteq M_{s_1} \times \ldots \times M_{s_n}$ for every $p : s_1 \times \ldots \times s_n \in P$. Given a signature morphism $\sigma : \Sigma = (S, F, P) \to \Sigma' = (S', F', P')$ and a $\Sigma'$-model $\mathcal{M}'$, $\text{Mod}(\sigma)(\mathcal{M}')$ is the $\Sigma$-model $\mathcal{M}$ defined for every $s \in S$ by $M_s = M'_{\sigma(s)}$, and for every function name $f \in F$ and predicate name $p \in P$, by $f^\mathcal{M} = \sigma(f)^{\mathcal{M}'}$ and $p^\mathcal{M} = \sigma(p)^{\mathcal{M}'}$. Finally, satisfaction is the usual first-order satisfaction.

Many other important logics can be obtained as FOL restrictions such as:

- **Horn Clause Logic (HCL).** An universal Horn sentence for a signature $\Sigma$ in FOL is a $\Sigma$-sentence of the form $\Gamma \Rightarrow \alpha$ where $\Gamma$ is a finite conjunction of $\Sigma$-atoms and $\alpha$ is a $\Sigma$-atom. The institution of Horn clause logic is the sub-institution of FOL whose signatures and models are those of FOL and sentences are restricted to the universal Horn sentences.

- **Equational Logic (EQL).** An algebraic signature $(S, F)$ simply is a FOL signature without predicate symbols. The institution of equational logic is the sub-institution of FOL whose signatures and models are algebraic signatures and algebras respectively.

- **Conditional equational logic (CEL).** The institution of conditional equational logic is the sub-institution of EQL whose sentences are universal Horn clauses for algebraic signatures.

- **Rewriting Logic (RWL).** Given an algebraic signature $\Sigma = (S, F)$, $\Sigma$-sentences are formulas of the form $\varphi : t_1 \Rightarrow t'_1 \land \ldots \land t_n \Rightarrow t'_n \Rightarrow t \Rightarrow t'$ where $t_i, t'_i \in T_F(X)_s$ $(1 \leq i \leq n, s_i \in S)$ and $t, t' \in T_F(X)_s$ $(s \in S)$. Models of rewriting logic are preorder models, i.e. given a signature $\Sigma = (S, F)$, $\text{Mod}(\Sigma)$ is the category of $\Sigma$-algebras $A$ such that for every $s \in S$, $A_s$ is equipped with a preorder $\geq$. Hence, $A \models \varphi$ if, and only if for every variable assignment $\nu : X \to A$, each $\nu(t_i)^A \geq \nu(t'_i)^A$ then $\nu(t)^A \geq \nu(t')^A$ where $A \models T_F(A) \Rightarrow A$ is the mapping inductively defined by: $f(t_1, \ldots, t_n)^A = f(A(t_1), \ldots, A(t_n))$.

2.2.3 Modal FOL (MFOL)

Signatures are couples $(\Sigma, A)$ where $\Sigma$ is a FOL-signature and $A$ is a set of actions, and morphisms are couples of FOL-signature morphisms and total functions on sets of actions. In the sequel, we will note by the same name both MFOL-signature and each of its components.
Given a MFOL signature \((\Sigma, A)\) with \(\Sigma = (S, F, P)\), \((\Sigma, A)\)-atoms are either predicates \(p(t_1, \ldots, t_n)\) or the symbol \(T\) (for \(T\)rue), and the set of \((\Sigma, A)\)-formulas is the least set of formulas built over the set of \((\Sigma, A)\)-atoms by finitely applying Boolean connectives in \(\{\neg, \lor, \land, \Rightarrow\}\), quantifiers \(\forall\) and \(\exists\), and modalities in \(\boxdot a \mid a \in A\). For every \(a \in A\), the intuitive meaning of \(\boxdot a\) is “always after the action \(a\)”.

Given a signature \((\Sigma, A)\), a \((\Sigma, A)\)-model \((W, R)\), called Kripke frame, consists of a family \(W = (W_i)_{i \in I}\) of \(\Sigma\)-models in FOL (the possible worlds) such that \(W_i W_j = W_j W_i\) for every \(i, j \in I\) and \(s \in S\), and a \(A\)-indexed family of “accessibility” relations \(R_a \subseteq I \times I\). Given a signature morphism \(\sigma: (\Sigma, A) \rightarrow (\Sigma', A')\) and a \((\Sigma', A')\)-model \((\langle W'' \rangle_{i \in I}, R'')\), \(\text{Mod}(\sigma)(\langle (W^n_i)_{i \in I}, R\rangle)\) is the \((\Sigma, A)\)-model \((\langle W'' \rangle_{i \in I}, R''\rangle)\) defined for every \(a \in A\) by \(R'_a = R''_{\sigma(a)}\). A \((\Sigma, A)\)-sentence \(\varphi\) is said to be satisfied by a \((\Sigma, A)\)-model \((W, R)\), noted \((W, R) \models (\Sigma, A) \varphi\), if for every \(i \in I\) we have \((W_i, R) \models \Sigma' \varphi\), where \(\Sigma'\) is inductively defined on the structure of \(\varphi\) as follows:

- for every FOL-formula \(\varphi\) built over \(\Sigma\)-atoms, \((W, R) \models \Sigma \varphi\) iff \(W_i \models \varphi\) for every \(i \in I\);
- \((W, R) \models \Sigma \sqcap a \varphi\) when \((W, R) \models \Sigma \varphi\) for every \(j \in I\) such that \(i R_a j\).

2.2.4 More exotic institutions

The institution theory also enables to represent formalisms which are not logics strictly speaking.

**Formal languages (FL)** The institution of formal languages is defined by the category of signatures \(\text{Set}\). Given a set \(A\), the set of sentences is \(A^*\) and \(\text{Mod}(A)\) is the category whose objects are all subsets of \(A^*\). Given a signature morphism \(\sigma: A \rightarrow A'\), \(\text{Mod}(\sigma)\) is the functor which at \(L' \subseteq A'^*\) associates the set \(L = \{\alpha | \sigma(\alpha) \in L'\}\). Finally, given a signature \(\Sigma \in \text{Sig}\), \(\Sigma\) is just the membership relation \(\exists\). It is obvious to show that the satisfaction condition holds.

**Programming languages (PLG)** The institution of a programming language [28] is built over an algebra of built-in data types and operations of a programming language. Signatures are FOL signatures and sentences are programs of the programming language over signatures; and models are algebraic structures in which functions are interpreted as recursive mappings (i.e. for each function symbol is assigned a computation (either diverging, or yielding a result) to any sequence of actual parameters). A model satisfies a sentence if, and only if it assigns to each sequence of parameters the computation of the function body as given by the sentence. Hence, sentences determine particular functions in the model uniquely. Finally, signature morphisms, model reductions and sentence translations are defined similarly to those in FOL.

3 Specifications in institutions

Over institutions, specifications are usually defined either by logical theories or couples \((\Sigma, Ax)\) where \(\Sigma\) is a signature and \(Ax\) a set (usually finite) of formulae (often called axioms) over \(\Sigma\). However, there is a large family of specification formalisms mainly used to specify concurrent, reactive and dynamic systems for which specifications are not expressed in this way. We can cite for instance process algebras, transition systems or Petri nets. Now, all of these kinds of specifications can be studied through the set of their semantic consequences expressed in an adequate formalism. This leads us up to define the notion of specifications over institutions.

3.1 Definitions

Let us now consider a fixed but arbitrary institution \(\mathcal{I} = (\text{Sig}, \text{Sen}, \text{Mod}, \models)\).

**Definition 5 (Specifications)** A specification language \(\mathcal{SL}\) over \(\mathcal{I}\) is a pair \((\text{Spec}, \text{Real})\) where:

- \(\text{Spec}: \text{Sig}^{op} \rightarrow \text{Set}\) is a functor. Given a signature \(\Sigma\), elements in \(\text{Spec}(\Sigma)\) are called specifications over \(\Sigma\).
- \(\text{Real} = \text{Sig}^{\Sigma} \text{Real}^\Sigma : \text{Spec}(\Sigma) \rightarrow |\text{Cat}|\) such that for every \(\Sigma \in |\text{Sig}|\), and every \(Sp \in \text{Spec}(\Sigma)\), \(\text{Real}^\Sigma(\text{Sp})\) is a full subcategory of \(\text{Mod}(\Sigma)\). Objects of \(\text{Real}^\Sigma(\text{Sp})\) are called realizations of \(Sp\).

**Definition 6 (Semantic consequences)** Let \(\mathcal{SL} = (\text{Spec}, \text{Real})\) be a specification language over \(\mathcal{I}\). Let us define \(\text{Sp}_\Sigma^\Sigma = \{\varphi \mid \forall M \in \text{Real}^\Sigma(\text{Sp}), M \models_\Sigma \varphi\}\). \(\text{Sp}_\Sigma^\Sigma\) is called the set of semantic consequences of \(Sp\) or the theory of \(Sp\).

Definition 6 calls for some comments:

- We could expect that \(\text{Mod}(\text{Sp}_\Sigma^\Sigma) = \text{Real}(\text{Sp})\) what would make unmeaning the existence of the mappings in \(\text{Real}\) in Definition 5. However, we can often be led up to make some restrictions on specification models. For instance, when dealing with axiom specifications expressed in equational logic, we can be interested by reachable or initial models to allow inductive proofs or for computability reasons.
• Sometimes, \( \_ \) is a natural transformation from \( \text{Spec} \) to \( P \circ \text{Sen}^{op} \). However, most of times, it is not the case (see the examples in Section 3.2).

**Definition. 7 (Category of specifications)** Let \( SL \) be a specification language over an institution \( I \). Denote \( \text{SPEC} \) the category of specifications over \( SL \) whose the objects are the elements in \( \bigcup_{\Sigma \in \text{Sig}} \text{Spec}(\Sigma) \), and morphisms are actually given by signature morphisms (i.e. for every \( Sp \in \text{Spec}(\Sigma) \) and every \( Sp' \in \text{Spec}(\Sigma') \), \( \sigma : Sp \rightarrow Sp' \in \text{SPEC} \) iff \( \sigma : \Sigma \rightarrow \Sigma' \in \text{Sig} \). If a morphism \( \sigma : Sp \rightarrow Sp' \) in \( \text{SPEC} \) further satisfies: \( \text{Sen}(\sigma)(Sp) \subseteq Sp' \), then \( \sigma \) is called specification morphism.

\( \text{Sig} : \text{SPEC} \rightarrow \text{Sig} \) is the functor which maps any specification \( Sp \in \text{Spec}(\Sigma) \) to the signature \( \Sigma \) and any morphism \( \sigma \) to the signature morphism \( \text{Sig}(\sigma) \).

Hence, specification morphisms are arrows in \( \text{SPEC} \) that further preserve semantic consequences. Commonly, the category of specifications over institutions have arrows \( \bigcup_{\Sigma \in \text{Sig}} \text{Spec}(\Sigma) \) as objects and specification morphisms as arrows [20, 29]. Here, the fact to consider just signature morphisms between specifications will be useful to define both architectural connectors and their combination.

### 3.2 Examples of specifications

We give three examples of specification languages that correspond to the usual forms of specifications over arbitrary institutions.

#### 3.2.1 Logical theories

Here, specifications are logical theories. To meet the requirements given in Definition 5, let this to the functor \( \text{Spec} : \text{Sig}^{op} \rightarrow \text{Set} \) which to every \( \Sigma \in \text{Sig} \), associates the set of all \( \Sigma \)-theories \( T \), and to every signature morphism \( \sigma : \Sigma \rightarrow \Sigma' \), matches every \( \Sigma' \)-theory \( T' \) with the \( \Sigma \)-theory \( T = \{ \varphi | \text{Sen}(\sigma)(\varphi) \in T' \} \). Hence, \( \text{Spec}(\Sigma) \subseteq \mathcal{P}(\text{Sen}(\Sigma)) \). We naturally define \( \text{Real}_{2}(T) = \text{Mod}(T) \). Moreover, specifications being saturated theories, this naturally leads to the identity function \( \Sigma : \text{Spec}(\Sigma) \rightarrow \mathcal{P}(\text{Sen}(\Sigma)) \).

---

\( ^{3} \) Given a function \( F : C \rightarrow \mathcal{D} \), \( F^{op} : C^{op} \rightarrow \mathcal{D}^{op} \) is the dual of \( F \) defined as follows:

- \( \forall o \in C, F^{op}(o) = F(o) \)
- \( f^{*} \) being the reverse arrow of \( f \) in \( C, \forall o, o' \in C, \forall f \in \text{Hom}_{C}(o, o'), F^{op}(f^{*}) = F(f)^{*} \)

The powerset functor \( \mathcal{P} : \text{Set}^{op} \rightarrow \text{Set} \) takes a set \( S \) to its powerset \( \mathcal{P}(S) \), and a set function \( f : S \rightarrow S' \) (i.e., an arrow from \( S' \) to \( S \) in \( \text{Set}^{op} \)) to the inverse image function \( f^{-1} : \mathcal{P}(S') \rightarrow \mathcal{P}(S) \) which associates to a subset \( A \subseteq S' \) the subset \( \{ s | f(s) \in A \} \) of \( S \).

\( \mathcal{P}(\text{Sen}(\Sigma)) \). It is easy to check that given a signature morphism \( \sigma : \Sigma \rightarrow \Sigma' \), the following diagram commutes and then \( \_ \) is a natural transformation:

\[
\begin{array}{ccc}
\text{Spec}(\Sigma) & \xrightarrow{\Sigma} & \mathcal{P}(\text{Sen}(\Sigma)) \\
\downarrow \text{Spec}(\sigma) & & \downarrow \mathcal{P}(\text{Sen}^{op}(\sigma^{*})) \\
\text{Spec}(\Sigma') & \xrightarrow{\Sigma} & \mathcal{P}(\text{Sen}(\Sigma'))
\end{array}
\]

(See Footnote 5 for the definition of \( \sigma^{*} \))

#### 3.2.2 Axiomatic specifications

In this case, specifications are defined by pairs \((\Sigma, Ax)\) where \( \Sigma \) is a signature and \( Ax \subseteq \text{Sen}(\Sigma) \), and given a signature morphism \( \sigma : \Sigma \rightarrow \Sigma' \), \( \text{Spec}(\sigma) \) matches every \( \Sigma' \)-specification \( Sp' = (\Sigma', Ax') \) to \( Sp = (\Sigma, \{ \varphi | \text{Sen}(\sigma)(\varphi) \in Ax' \}) \). By the satisfaction condition, we have that \( \text{Sen}(\sigma)(Ax') \subseteq Ax^{*} \). The functor \( \text{Spec} \) then associates to every signature \( \Sigma \) the set of pairs \((\Sigma, Ax)\), and \( (\Sigma, Ax)^{\Sigma} = Ax^{*} \). Observe that \( \_ \) is not a natural transformation. Indeed, let us set \( F \in \text{FOL} \), and consider the inclusion morphism \( \sigma : \Sigma \rightarrow \Sigma' \) where \( \Sigma' = (\{s\}, \emptyset, \{R_{1}, R_{2} : s \times s\}) \) and \( \Sigma = (\{s\}, \emptyset, \{R_{1} : s \times s\}) \). Let \( Ax^{*} \) be the set of axioms:

\[
x R_{2} y \Rightarrow y R_{2} x \\
x R_{1} x \Leftarrow x R_{2} y
\]

Obviously, we prove from \( Ax^{*} \) that \( R_{1} \) is a symmetric relation.

However, \( \text{Spec}(\sigma)((\Sigma', Ax')) = \emptyset \), and then \( \text{Spec}(\sigma)((\Sigma', Ax'))^{*} \) is restricted to tautologies while \( \mathcal{P}(\text{Sen}^{op}(\sigma^{*}))(Ax') \) contains at least \( x R_{1} y \Rightarrow y R_{1} x \).

#### 3.2.3 Inference rules

In the framework of formal language, languages \( L \) over an alphabet \( A \) can be specified by inference rules, that is \( n \)-ary relations \( r \) on \( A^{*} \) and a tuple \( (\alpha_{1}, \ldots, \alpha_{n}) \in r \) means that if \( \alpha_{1}, \ldots, \alpha_{n-1} \) are words of the language, then so is \( \alpha_{n} \). Hence, a specification over an alphabet \( A \) is a set \( R \) of \( n \)-ary relations on \( A^{*} \). Given a signature morphism \( \sigma : A \rightarrow A' \) and a specification \( R' \) over \( A' \), the specification \( \text{Spec}(\sigma)(R') \) over \( A \) is the set \( R \) of \( n \)-ary relation \( r \) such that there exists \( r' \in R' \) and \( r = \{(a_{1}, \ldots, a_{n}) | (\forall i, 1 \leq i \leq n, a_{i} \in A) \land (a_{1}, \ldots, a_{n}) \in r' \} \). Given a set of inference rules \( R \) over an alphabet \( A, R_{A}^{*} \), is the language \( L \) inductively generated from inference rules of \( R \). Given a signature morphism \( \sigma : A \rightarrow A' \) and a set of inference rules \( R' \) over \( A' \), let that is easy to show that \( \text{Spec}(\sigma)(R')^{*} = R_{A}^{*} \cap A^{*} \) what proves that \( \_ \) is a natural transformation from \( \text{Spec} \) to \( \mathcal{P} \circ \text{Sen}^{op} \).
3.3 Properties of specifications

Proposition 1. Let \( \sigma : \text{Sp} \to \text{Sp}' \) be a specification morphism. Then, the functor \( \text{Mod}(\sigma) : \text{Mod}(\text{Sig}(\text{Sp}')) \to \text{Mod}(\text{Sig}(\text{Sp})) \) can be restricted to specification semantic consequences (i.e. \( \text{Mod}(\sigma) : \text{Mod}(\text{Sp}'^\Sigma) \to \text{Mod}(\text{Sp}^\Sigma) \) is a functor).

Proof. Let \( \varphi \in \text{Sp}'^\Sigma \) and \( \mathcal{M} \in \text{Mod}(\text{Sp}') \). As \( \sigma \) is a specification morphism, \( \mathcal{M} \models_{\text{Sig}(\text{Sp}')} \text{Sen}(\sigma)(\varphi) \). Therefore, by the satisfaction condition, we also have that \( \text{Mod}(\sigma)(\mathcal{M}) \models_{\text{Sig}(\text{Sp})} \varphi \).

We cannot state a similar result for the family of mappings \( \text{Real} \), i.e. we cannot define in a general way a functor of the form \( \text{Real}(\sigma) : \text{Real}(\text{Sp}') \to \text{Real}(\text{Sp}) \). The following notion of compatibility captures the existence of such a functor.

Definition 8 (Compatible). Let \( \mathcal{S} = (\text{Spec}, \text{Real}) \) be a specification language over \( \mathcal{T} \). Let \( \sigma : \text{Sp} \to \text{Sp}' \) be a specification morphism. \( \text{Real} \) is said compatible with \( \sigma \) if, and only if we can define a functor \( \text{Real}(\sigma) : \text{Real}(\text{Sp}') \to \text{Real}(\text{Sp}) \).

Here, we define two other notions that we will use after.

Definition 9 (Definable by specification). Given an institution \( \mathcal{T} \) and a specification language over \( \mathcal{T} \), a \( \Sigma \)-theory \( T \) is said definable by specification or definable for being shorter if, and only if there exists \( \text{Sp} \in \text{Spec}(\Sigma) \) such that \( T = \text{Sp}'^\Sigma \).

In the following definition, we now adapt the standard notion of liberal specification morphism [12] which will be useful in Section 4.3.

Definition 10 (Liberality). In any specification language \( \mathcal{S} = (\text{Spec}, \text{Real}) \) over \( \mathcal{T} \), a specification morphism \( \sigma : \text{Sp} \to \text{Sp}' \) is liberal if, and only if \( \text{Real} \) is compatible with \( \sigma \) and \( \text{Real}(\sigma) : \text{Real}(\text{Sp}') \to \text{Real}(\text{Sp}) \) has a left-adjunct \( \mathcal{F}(\sigma) : \text{Real}(\text{Sp}) \to \text{Real}(\text{Sp}') \).

Specifications defined by logical theories and axiomatic specifications over the institution CEL is liberal for every specification morphism \( \sigma \). Indeed, let \( \sigma : \Sigma = (S, F) \to \Sigma' = (S', F') \) be a signature morphism, and let \( \Gamma \) and \( \Gamma' \) be two sets of conditional equations over, respectively, \( \Sigma \) and \( \Sigma' \) such that \( \text{Sen}(\sigma)(\Gamma) \subseteq \Gamma' \). We can build a functor \( T_{\Gamma'/\Gamma} : \mathcal{A} \to T_{\Gamma'/\Gamma}(\mathcal{A}) \), from the category of \( \Gamma \)-algebras to the category of \( \Gamma' \)-algebras.

Let \( \mathcal{A} \) be a \( \Gamma \)-algebra. \( T_{\Gamma'/\Gamma}(\mathcal{A}) \) is the quotient of \( T_{\Gamma'}(\mathcal{A}) \) by the congruence generated by the kernel of the \( \Sigma \)-morphism \( T_{\Gamma'}(\mathcal{A}) \) in \( \mathcal{A} \) extending the identity on \( X \). This algebra satisfies the following universal property: for every \( \Gamma' \)-algebra \( B \) and every \( \Sigma \)-morphism \( \mu : \mathcal{A} \to \text{Mod}(\sigma)B \), there exists a unique \( \Sigma \)-morphism \( \eta_B : T_{\Gamma'/\Gamma}(\mathcal{A}) \to B \) such that for every \( a \in A \), \( \eta_B(a) = \mu(a) \). This universal property directly shows that the functor \( T_{\Gamma'/\Gamma} \) is left-adjunct to \( \text{Mod}(\sigma) \), i.e., for every \( \Gamma \)-algebra \( \mathcal{A} \) there exists a universal morphism \( \mu_A : \mathcal{A} \to \text{Mod}(\sigma)(T_{\Gamma'/\Gamma}(\mathcal{A})) \). \( \mu_A \) is called the adjoint morphism for \( \mathcal{A} \).

4 Architectural connector

4.1 Definitions

Succinctly, architectural connectors enable one to combine components (specifications) together to make bigger ones. However, depending on the used specification language, the way of combining components can be different. For instance, when specifications are logical theories then their combination is often based on the set theoretical union on signatures whereas the combination of specifications made of transition systems is based on some kinds of product. However, one can observe that most of existing connectors \( c \) have the following common features:

- a connector \( c \) gets as arguments a fixed number \( n \) of existing specifications \( \text{Sp}_1, \text{Sp}_2, \ldots, \text{Sp}_n \) defined respectively over the signatures \( \Sigma_1, \Sigma_2, \ldots, \Sigma_n \), to build a new one, denoted \( \text{Sp} = c(\text{Sp}_1, \text{Sp}_2, \ldots, \text{Sp}_n) \). We can then see the connector \( c \) as a mapping of arity \( n \) from \( \text{SPEC}^n \) to \( \text{SPEC} \). We will see in the examples that actually \( c \) may be a partial function, but often defined in a way sufficiently general to accept as arguments tuples \( (\text{Sp}_1, \text{Sp}_2, \ldots, \text{Sp}_n) \) with a large associated family of signature tuples \( (\Sigma_1, \Sigma_2, \ldots, \Sigma_n) \).

- as specifications will be recursively defined by means of connectors, the arguments \( \text{Sp}_1, \text{Sp}_2, \ldots, \text{Sp}_n \) of the connector \( c \) can be linked together by some constraints on elements present in specification signatures, expressed by signature morphisms. These constraints will be taken into account by the definition of the connector \( c \). Hence, the arguments of a connector \( c \) will not be a tuple of \( n \) specifications, but \( n \) specifications equipped with signature morphisms. This will be defined by a graph whose nodes are specifications and edges are signature morphisms. In our category theory based setting, such a graph is called a diagram of the specification category \( \text{SPEC} \). In practice, for a given connector \( c \), all the diagrams accepted as arguments by

---

6\( T_{\Gamma'}(\mathcal{A}) \) (resp. \( T_{\Gamma}(\mathcal{A}) \)) is the term algebra built over \( F' \) (resp. \( F \)) with sorted variables in the carrier \( A \) of the \( \Gamma \)-algebra \( \mathcal{A} \).
\(c\) have the same graph shape (i.e. the same organization between nodes and edges). Hence, our connectors will be built on the diagram category with the same shape over the category \(SPEC\).

- the signature \(\Sigma\) of \(SP\) is the least one over the signatures \(\Sigma_1, \Sigma_2, \ldots, \Sigma_n\). This expresses the fact that generally, a connector \(c\) does not explicitly introduce new elements to be specified, but on the contrary only combines the elements already present in one of the signatures \(\Sigma_1, \Sigma_2 \ldots \Sigma_n\). In the following definition of connectors, this will be expressed by the co-limit of the diagram, projected on signatures.

This then leads us up to formally define architectural connectors as follows:

**Notation. 1 (Diagram category)** Let \(I\) and \(C\) be two categories. Note \(\Delta_{(I,C)}\) the category of diagrams in \(C\) with shape \(I\), i.e. the category whose objects are all functors \(\delta : I \to C\), and morphisms are natural transformations between functors \(\delta, \delta' : I \to C\).

Let \(I'\) be a subcategory of a category \(I\). Let \(\delta\) be a diagram of \(\Delta_{(I,C)}\). Let us denote \(\delta_{i'}\), the diagram of \(\Delta_{(I',C)}\) obtained by restricting \(\delta\) to \(I'\).

**Definition. 11 (Co-cone)** Given a diagram \(\delta : I \to C\). A co-cone of \(\delta\) consists of an object \(c \in \mathcal{C}\) and a 1-indexed family of morphisms \(\alpha_i : \delta(i) \to c\) such that for each edge \(e : i \to i'\) in \(I\), we have that \(\alpha_{i'} \circ \delta(e) = \alpha_i\).

A co-limiting co-cone (co-limit) \((c, \{\alpha_i\}_{i \in I})\) can be understood as a minimal co-cone, that is:

**Definition. 12 (Co-limit)** A co-cone \((c, \{\alpha_i\}_{i \in I})\) of a diagram \(\delta\) is a co-limit if, and only if it has the property that for any other co-cone \((d, \{\beta_i\}_{i \in I})\) of \(\delta\), there exists a unique morphism \(\gamma : c \to d\) such that for every \(i \in I\), \(\gamma \circ \alpha_i = \beta_i\).

When \(I\) is the category \(\bullet \leftarrow \bullet \to \bullet\) with three objects and two non-identity arrows, the co-limit is called a pushout.

**Definition. 13 (Co-complete)** A category \(C\) is co-complete if for every shape category \(I\), every diagram \(\delta : I \to C\) has a co-limit.

In the sequel, we will then consider institutions whose the signature category is co-complete.

**Definition. 14 (Architectural connector)** Let \(SL\) be a specification language over an institution \(\mathcal{I}\) for which the category \(Sig\) is co-complete. An architectural connector \(c : [\Delta_{(I,SPEC)}] \to [SPEC]\) is a partial mapping such that every \(\delta \in \Delta_{(I,SPEC)}\) for which \(c(\delta)\) is defined, is equipped with a co-cone \(p : Sig \circ \delta \to Sig(c(\delta))\) co-limit of \(Sig \circ \delta\).

**Example. 1 (Enrichment and union)** Enrichment and union of specifications have surely been the first primitives architectural connectors (so-called structuring primitives) to be formally defined and studied especially when dealing with specifications defined as axiomatic specifications. They even received an abstract formalization in institutions [8]. In our framework, both structuring primitives are defined as follows: we consider an institution \(\mathcal{I} = (\text{Sig}, \text{Sen}, \text{Mod}, \models)\). Moreover, in Example 1, \(SPEC\) is the category whose objects are specifications of the form \((\Sigma, Ax)\) over a given institution \(\mathcal{I}\) and morphisms are any \(\sigma : (\Sigma, Ax) \to (\Sigma', Ax')\) s.t. \(\sigma : \Sigma \to \Sigma'\) is a signature morphism.

**Enrichment** Let \(I\) be the graph composed of two nodes \(i\) and \(j\) and one arrow \(a : i \to j\). The connector Enrich for axiomatic specifications is defined for every diagram \(\delta : I \to SPEC\) where \(\delta(i) = (\Sigma, Ax)\) and \(\delta(j) = (\Sigma', Ax')\) such that \(\text{Sen}((\text{Sig}(\delta(a)))(Ax)) \subseteq Ax'\), and yields \(\text{Enrich}(\delta) = (\Sigma', Ax')\) together with the co-cone \(\text{Sig}(\delta(a))\) and \(I_d(\text{Sig}(\delta(a)))\) which is the obvious co-limit of \(\text{Sig} \circ \delta\). Observe that \(\delta(a)\) and \(I_{d}(\delta(a))\) are further specification morphisms.

**Union** Let \(I\) be the graph composed of three nodes \(i, j\), and \(k\) and two arrows \(a_1 : i \to j\) and \(a_2 : j \to k\). The connector Union is defined for every diagram \(\delta : I \to SPEC\) where \(\delta(i) = (\Sigma_0, Ax_0)\), \(\delta(j) = (\Sigma_1, Ax_1)\) and \(\delta(k) = (\Sigma_2, Ax_2)\), and such that \(\text{Sen}((\text{Sig}(\delta(a_1)))(Ax_0)) \subseteq Ax_1\) and \(\text{Sen}((\text{Sig}(\delta(a_2)))(Ax_0)) \subseteq Ax_2\), and yields \(\text{Union}(\delta) = (\Sigma, Ax)\) with the co-cone \(p : (\Sigma, Ax) \to \Sigma\) which is the pushout of \(\text{Sig}(\delta(a_1))\) and \(\text{Sig}(\delta(a_2))\) and such that \(Ax = \text{Sen}(p_j)(Ax_1) \cup \text{Sen}(p_k)(Ax_2)\). Observe that we can derive the co-cone \(p_{SPEC} : \delta \to (\Sigma, Ax)\) such that \(\text{Sig} \circ \delta \to \Sigma\) where \(p_i\) and \(p_j\) are the identity signature morphism for \(\Sigma\), and such that \(Ax = Ax_1 \cup Ax_2\).

In [8], both above connectors have been brought down to two basic connectors: union with constant signatures \(\bigcup\), and translate \(\text{by}\) \(\sigma\) for every signature morphism \(\sigma\). They are defined by:

1. Let \(I\) be the graph composed of two nodes \(i\) and \(j\) and without arrows between \(i\) and \(j\). The connector \(\bigcup\) is defined for every diagram \(\delta : I \to SPEC\) where \(\delta(i) = (\Sigma, Ax_1)\) and \(\delta(j) = (\Sigma, Ax_2)\), and yields \(\bigcup(\delta) = (\Sigma, Ax)\) with the obvious co-limit \(p : \text{Sig} \circ \delta \to \Sigma\) where \(p_i\) and \(p_j\) are the identity signature morphism for \(\Sigma\), and such that \(Ax = Ax_1 \cup Ax_2\).

2. Let \(I\) be the graph composed two nodes \(k\) and \(l\). The connector translate \(\text{by}\) \(\sigma\) where \(\sigma : \Sigma \to \Sigma'\) is a signature morphism, is defined for every diagram \(\delta : I \to SPEC\) where \(\delta(k) = (\Sigma, Ax)\) and \(\delta(l) = (\Sigma', \text{Sen}(\sigma)(Ax))\), and yields translate \(\text{by}\) \(\sigma(\delta) = (\Sigma', \text{Sen}(\sigma)(Ax))\).
In [8], \(\bigcup(\delta)\) and translate _ by \(\sigma(\delta)\) are respectively noted \(\delta(i)\bigcup\delta(j)\) and translate \(\delta(k)\) by \(\sigma\).

Architectural connectors can be combined to deal with specifications in the large.

**Definition. 15 (Connector combination)** Let \(c : |\Delta_{I,SPEC}| \rightarrow |SPEC|\) and \(c' : |\Delta_{I',SPEC}| \rightarrow |SPEC|\) be two architectural connectors. Let \(I' \in |I'|\) be an object. Let \(I' \circ_{\nu} I\) be the category defined by:

- \(|I' \circ_{\nu} I| = |I| \coprod |I'|
- the sets \(\text{Hom}_{I' \circ_{\nu} I}(k,l)\) for every \(k,l \in |I' \circ_{\nu} I|\) are inductively defined as follows:
  - \(k,l \in |I'| \Rightarrow \text{Hom}_{I'}(k,l) \subseteq \text{Hom}_{I' \circ_{\nu} I}(k,l)\)
  - \(k,l \in |I| \Rightarrow \text{Hom}_{I}(k,l) \subseteq \text{Hom}_{I' \circ_{\nu} I}(k,l)\)
- for every \(i \in |I|\), we introduce the arrow \(q_i\) in \(\text{Hom}_{I' \circ_{\nu} I}(i,i')\).
- \(\text{Hom}_{I' \circ_{\nu} I}\) is closed under composition.

Let us denote \(c' \circ_{\nu} c : |\Delta_{I' \circ_{\nu} I,SPEC}| \rightarrow |SPEC|\) the architectural connector defined by :

\[
\begin{array}{ll}
\delta' & \text{if } \delta(c(\delta)) \text{ is defined} \\
\delta & \text{and } \delta(q_i) \text{ is the morphism } r_i \text{ in } SPEC \\
\end{array}
\]

where the image by \(\text{Sig}\) is the component \(p_i\) of the co-limit \(p\) associated to \(c(\delta)\).

**Example. 2** Enrichment can be removed and replaced by the following combination of translate and \(\cup\) as follows: let \(\delta\) be a diagram of \(\Delta(I,SPEC)\) where \(I\) is the index category of the connector \(\text{Enrich}\), \(\delta(i) = (\Sigma_i, Ax_i)\) and \(\delta(j) = (\Sigma_j, Ax_j)\).

\[
\text{Enrich}(\delta) = \bigcup_{\sigma(\delta)} \text{translate}_\delta(p)(\delta')
\]

where \(\delta'\) is the diagram of \(\Delta(I',I',SPEC)\) for \(I'\) (resp. \(I\)) the index category of the connector \(\cup\) (resp. translate), defined by: \(\delta'(k) = \delta(i), \delta'(i) = \text{translate}_\delta(p)(\delta_i) = (\Sigma_i, \text{Sen}(\Sigma(p_i))(Ax_i))\) and \(\delta'(j) = (\Sigma_j, Ax_j)\).

The reader accommodated to the terminology and to the concepts of software architecture can be disappointed by the way connectors are interpreted here, i.e. by functions that take components and produce systems. Indeed, connectors are typically viewed as forms of communicating components. Such connectors can also be formalized in our framework. For instance, in Community [15, 16], in the style of Allen and Garlan [6], a connector consists of \(n\) roles \(R_i\) and one glue \(G\) stating the interaction between roles (i.e. the way roles communicate together). Roles and glue are programs defined over signatures (see [16] for a complete definition of programs). In our framework, programs denote specifications from which we can observe temporal properties. Each role and the glue are interconnected by a channel to denote via signature morphisms shared attributes and actions. This gives rise to a diagram defined as the interconnection on the glue \(G\) of basic diagrams of the form:

In Community, the mathematical meaning of a connector is then defined by the colimit of such diagrams. This can be easily defined in our framework by considering a connector \(c\) defined for every diagram of the previous form over the category \(PROG\) (defined in [16]) taken as the category \(SPEC\).

### 4.2 Complex structuring

As already explained in the introduction of the paper, an architectural connector will be considered as complex when:

1. The global system does not preserve the complete behavior of some subsystems. We will then talk about non-conformity properties.

2. Some global properties cannot be deduced from a complete knowledge of these components. We will then talk about true emergent properties.

This is expressed by comparing the set of semantic consequences of subsystems with the ones of the global system up to signature morphisms.

**Definition. 16 (Complex connector)** Let \(c : |\Delta(I,SPEC)| \rightarrow |SPEC|\) be an architectural connector. Let \(\delta\) be a diagram of \(\Delta(I,SPEC)\) such that \(c(\delta)\) is defined. \(c\) is said complex for \(\delta\) if, and only if one of the two following properties fails:

1. **Conformity.**
   \[
   \forall i \in I, \forall \varphi \in \text{Sen}(\Sigma(\delta(i))), \varphi \in \delta(i) \iff \text{Sen}(p_i)(\varphi) \in c(\delta) \bullet \text{Sig}(c(\delta))
   \]
2. Non true emergence.
\[ \forall \varphi \in c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))} \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \models_{\text{Sig}(c(\delta))} \varphi \]

A formula \( \varphi \) that makes fail the equivalence of both Point 1. and Point 2. is called emergent property.

If \( c \) is not complex for a diagram \( \delta \), then it is said modular.

Example. 3 Here, we give a very simple example of specifications in which modularity fails. Let \( \text{Nat} \) be the specification in EQL defined as follows:

**Specification of Nat**

**Sorts:** \( S_{\text{Nat}} = \{ \text{nat} \} \)

**Functions:** \( F_{\text{Nat}} = \{ 0 : \text{nat}, \text{succ} : \text{nat} \to \text{nat}, + : \text{nat} \times \text{nat} \to \text{nat} \} \)

**Axioms:** \( Ax_{\text{Nat}} \)
\[ \{ x + 0 = x, x + \text{succ}(y) = \text{succ}(x + y) \} \]

Let us enrich this specification by adding operations and axioms to specify stacks of natural numbers. This leads to the following enrichment:

**Sorts:** \( S_{\text{Stack}} = \{ \text{nat}, \text{stack} \} \)

**Functions:** \( F_{\text{Stack}} = F_{\text{Nat}} \cup \{ \text{empty} : \to \text{stack}, \text{push} : \text{nat} \times \text{stack} \to \text{stack}, \text{pop} : \text{stack} \to \text{stack}, \text{top} : \text{stack} \to \text{nat}, \text{high} : \text{stack} \to \text{nat} \} \)

**Axioms:** \( Ax_{\text{Stack}} = Ax_{\text{Nat}} \cup \{ \text{pop}(\text{empty}) = \text{empty}, \text{pop}(\text{push}(e, P)) = P, \text{top}(\text{push}(e, P)) = e, \text{high}(\text{push}(e, P)) = \text{succ}(\text{high}(P)) \} \)

If we suppose that realizations are either the initial model or reachable models\(^8\) of both specifications, then an example of emergent property is:
\[ \forall x, (x = 0) \lor (\exists y, x = \text{succ}(y)) \]

This is because \( \text{high}(\text{empty}) \) has not been specified to be equal to 0. On the contrary, if we add this equation in \( Ax_{\text{Stack}} \), there is not emergent property anymore.

4.3 Conditions for modularity

As we have explained it in the introduction of this manuscript, complex software systems prevent to check their correctness with respect to their specification step by step by taking the benefit of their recursive structure. This leads to the important consequence that adding any component gives rise to a new systems whose the correctness has to be completely (re)checked. It is then important to study general properties that guarantee when a system is not complex (i.e. modular). This is what we propose to do with the two following results.

Theorem 1 states that showing the non-presence of true emergent properties for a connector \( c \) and a diagram \( \delta \) comes to show that \( \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \) is definable by \( c(\delta) \).

**Theorem. 1** Let \( c \) be an architectural connector and \( \delta \) be a diagram such that \( c(\delta) \) is defined. Then, we have:
\[ \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \] is definable by \( c(\delta) \).

**Proof.** The only if part. This obviously results from the fact that \( \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \) is definable by \( c(\delta) \). Indeed, we have:
\[ c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))} = \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}), \]
that is for every \( \varphi \in c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))} \), we have that:
\[ \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \models_{\text{Sig}(c(\delta))} \varphi. \]

The if part. As each \( p_i \) of \( p \) is a specification morphism, we have that:
\[ \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \subseteq c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))}. \]
Moreover, as the set of true emerging properties is empty, we have that:
\[ c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))} \subseteq \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}). \]

Hence, \( c(\delta)\mathrel{\downarrow}_{\text{Sig}(c(\delta))} = \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}). \) and then:
\[ \bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)\mathrel{\downarrow}_{\text{Sig}(\delta(i))}) \] is definable by \( c(\delta) \).

By Theorem 1, the architectural connectors \( \text{Enrich} \), \( \text{Union} \), \( \bigcup \) and \( \text{translate} \) by \( \sigma \) have no true emergence properties for any defined diagram.

As we could expect, modularity is a property which holds for some, but certainly not for all architectural connectors. More surprising, even under the condition that

\(^8\) A model is reachable when any of its values is the result of the evaluation of a ground term.
(\bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)_{\text{Sig}()}))^* is definable by c(\delta) for a connector c and a diagram \delta such that c(\delta) is defined, modularity can fail because of non-conformity properties (see Example 3).

In the next theorem, we give a supplementary condition based on the liberality of each p_i of the co-limit p, that leads to an empty set of non-conformity properties. For Theorem 2, we suppose the following conditions:

1. the institution under consideration is closed under isomorphism,
2. Real is compatible for every specification morphism p_i of the associated co-cone p, and
3. each p_i of the co-limit p associated to the connector c in \Delta, SPEC satisfies the supplementary condition, so-called Right Satisfaction Condition (RSC) : \forall \varphi \in \text{Sen}(\text{Sig}()), \forall M \in \text{Real}(\delta(c)), \text{Real}(p_i)(M) \models_{\text{Sig}()} \varphi \iff M \models_{\text{Sig}(\delta(c))} \text{Sen}(p_i)(\varphi).

The interest of RSC is, realizability being a subset of models, some pruning on realizations in \text{Real}(\delta(c)) have been allowed to be done, and then this direction of the satisfaction condition has been able to be brought into failure. For instance, this property does not hold when specifications are logical theories and realizations are restricted to reachable models (see Example 3). For the next theorem, we suppose that these three conditions hold.

**Theorem 2** Let c be an architectural connector and \delta be a diagram such that c(\delta) is defined. Suppose that (\bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)_{\text{Sig}()}))^* is definable by c(\delta). Real is compatible with each p_i and each p_i is liberal. Then, for every i \in I and every M \in \text{Real}(\delta(i)), If each adjunct morphism \mu_M : M \to \text{Real}(p_i)(\text{F}(p_i)(M)) is an isomorphism, then the set of non-conformity properties is empty.

**Proof.** Let \varphi \in \delta(i)_{\text{Sig}()}, and let M \in \text{Real}(\delta(c)). As (\bigcup_{i \in I} \text{Sen}(p_i)(\delta(i)_{\text{Sig}()}))^* is definable by c(\delta),\text{Real}(p_i)(M) \models_{\text{Sig}()} \varphi. Therefore, by the hypothesis that the truth of property is preserved for the functor Real through each signature morphism p_i, we have that M \models_{\text{Sig}(\delta(c))} \text{Sen}(p_i)(\varphi).

let \varphi \in \text{Sen}(\delta(i)) such that \text{Sen}(p_i)(\varphi) \in c(\delta)^* and let M \in \text{Real}(\delta(c)). As \text{F}(p_i) is left-adjunct to \text{Real}(p_i), we have \text{F}(p_i)(M) \models_{\text{Sig}(\delta(c))} \text{Sen}(p_i)(\varphi). As Real is compatible with each p_i, \text{Real}(\sigma)(\text{F}(p_i)(M)) \models_{\text{Sig}(\delta(c))} \varphi. As the adjunct morphism is an isomorphism and \mathcal{I} is stable under isomorphism, M and \text{Real}(\sigma)(\text{F}(p_i)(M)) are elementary equivalent, and then M \models_{\text{Sig}(\delta(c))} \varphi.

Theorem 2 generalizes to any architectural connectors the standard condition of modularity based on the two notions of hierarchical consistency and sufficient completeness [22], which has been stated for the enrichment connector in the algebraic specification framework (when specifications are conditional positive).

**5 Application to reactive systems**

In this section, we propose to exemplify our abstract framework to reactive system modeling. We will then give a rigorous and formal definition of emergent properties in the framework of reactive system modeling. We restrict ourselves to reactive systems described by means of the usual synchronous product of transition systems, and whose behavior is expressed by logical properties over MFOL. The reason is this is sufficient for the purpose of the study, and the results given in this paper could easily be adapted to temporal logics more classically used to reason on reactive systems and other composition connector whose the greatest number are based-on transition system product. In our setting, we will study some conditions under which non-conformity properties do not occur. The interest is this provides guidance in the design process. Indeed, the appearance of non-conformity properties leads to make a posteriori verification of the global system without benefiting from the decomposition of the system into components.

In Section 5.1, we introduce transition systems and their semantics, and define the synchronous product as means to compose them. Finally, Section 5.2 presents results ensuring the non-existence of non-conformity properties along synchronous product.

**5.1 Transition systems**

**5.1.1 Syntax**

As usual when considering automata, transition systems describe possible evolutions of system states. Elementary evolutions are represented by a transition relation between states. Each transition between two states is labeled by three elements: actions of the system, guards expressed here by formulas of FOL presented in Section 2, and side-effects on states defined by pairs of ground terms or of the form (p(t_1, \ldots, t_n), b) where p(t_1, \ldots, t_n) is a ground atom and b is equal to true or false. As usual, we start by defining the language, so-called signature, on which transition systems are built:

**Definition 17 (Signature)** A signature is a triple \( \mathcal{L} = (\Sigma, V, A) \) where: \( \Sigma \) is a FOL-signature, \( V \) is a set of variables over \( \Sigma \) and \( A \) is a set whose elements are called actions.
Definition. 18 (Side-effect) Given a signature \( \mathcal{L} = (\Sigma, V, A) \) where \( \Sigma = (S, F, P) \), a side-effect over \( \mathcal{L} \) is a pair of ground terms over \( (\Sigma, t, t') \) of the same sort (i.e. \( \exists s \in S, t, t' \in T_F \) or a couple \((p(t_1, \ldots, t_n), b)\) where \( p(t_1, \ldots, t_n) \) is a ground \( \Sigma \)-atom (i.e. each \( t_i \) is a ground term) and \( b \) is true or false. In the sequel, a side-effect \( (t, t') \) will be noted \( t \rightarrow t' \).

We note \( SE(\mathcal{L}) \) the set of side-effects over \( \mathcal{L} \).

A transition system is then defined as follows:

Definition. 19 (Transition system) Given a signature \( \mathcal{L} = (\Sigma, V, A) \), a transition system is a couple \((Q, T)\) where:

- \( Q \) is a set of states, and
- \( T \subseteq Q \times A \times Sen(\Sigma) \times 2^{SE(\mathcal{L})} \times Q \).

A small specification example is given in [2]. Transition systems are specifications of reactive systems. Given a signature morphism \( \sigma : (\Sigma, A) \rightarrow (\Sigma', A') \) and a specification \( S' = (Q', T') \) over \((\Sigma', A')\), \( Spec(\sigma)(S') \) is the specification \( S = (Q, T) \) over \((\Sigma, A)\) such that \( Q = Q' \) and \( T = \{(q, a, \varphi, \delta, q')|(q, \sigma(a), Sen(\sigma)(\varphi), \sigma(\delta), q') \in T'\} \).

5.1.2 Semantics

Semantics of transition systems are defined by Kripke frames themselves defined as follows:

Definition. 20 (Kripke frame) Given a signature \( \mathcal{L} = (\Sigma, V, A) \), an Kripke frame over \( \mathcal{L} \) or \( \mathcal{L} \)-model, is a couple \((W, R)\) where:

- \( W \) is a I-indexed set \((W_i)_{i \in I}\) of \( \Sigma \)-models such that \( W_i = W_j \) for every \( i, j \in I \) and \( s \in S \), and
- \( R \) is a \( A \)-indexed set of “accessibility” relations \( R_a \subseteq I \times I \).

Here, states are defined by \( \Sigma \)-models. Therefore, side-effects will consist on moving from a \( \Sigma \)-model to another one by changing the semantics of functions according the assignments given in the set \( \delta \) of transitions. Formally, this is defined as follow: if \( A \) is a \( \Sigma \)-model, then \( t^A : T_F \rightarrow A \) is the \( \Sigma \)-morphism inductively defined by \( f(t_1, \ldots, t_n) \mapsto f^A(t_1^A, \ldots, t_n^A) \).

Definition. 21 (Side-effect semantics) Let \( \mathcal{L} = (\Sigma, V, A) \) be a signature. Let \( A \) and \( B \) be two \( \Sigma \)-models. We note \( A \sim_B \) to mean that the state \( A \) is transformed into the state \( B \) along \( \delta \), if and only if \( B \) is defined as \( A \) except that for every \( t \mapsto t' \in \delta \) (resp. \( p(t_1, \ldots, t_n) \mapsto b) \), \( t'^B = t'^A \) (resp. \( (t_1^A, \ldots, t_n^A) \in p^B \) iff \( b = true \)).

Definition. 22 (Semantics of transition systems) Given a transition system \( S = (Q, T) \) over a signature \( \mathcal{L} \), the semantics for \( S \), noted \( Real(S) \), is the set of all the Kripke frames \((W, R)\) over \( \mathcal{L} \) such that the set of indexes \( I = Q \), and satisfying both implications:

1. \( (q, a, \varphi, \delta, q') \in T \Leftrightarrow \varphi \wedge W^q \sim_\delta W^{q'} \Rightarrow q R_a q' \)
2. \( q R_a q' \Rightarrow \exists(q, a, \varphi, \delta, q') \in T, W^q \Rightarrow \varphi \wedge W^q \sim_\delta W^{q'} \)

Hence, the way whose dynamic is dealt with in this paper follows the state-as-algebra style [21, 3] where states are \( \Sigma \)-models and state transformations are transitions from a state-model to another state-model.

5.1.3 Synchronous product

Synchronous product combines two transition systems into a single one by synchronizing transitions. Understandably, executions of synchronous product modelize system behavior as a synchronizing concurrent system. Hence, when an action \( a \) is “executed” in the product, then every component with \( a \) in its alphabet must execute a transition labeled with \( a \). Formally, the synchronous product of two transition systems is defined as follows:

Definition. 23 (Synchronous product) Let \( S_i = (Q_i, T_i) \) be a transition system over a signature \( \mathcal{L}_i = (\Sigma_i, V_i, A_i) \) with \( i = 1, 2 \) such that:

- for every transition \((q_1, a, \varphi_1, \delta_1, q_1') \in T_1 \) and every \( f(t_1, \ldots, t_n) \mapsto t_1' \in \delta_1 \) (resp. \( p(t_1, \ldots, t_n) \mapsto b \in \delta_1 \)), there does not exist a transition \((q_2, a, \varphi_2, \delta_2, q_2') \in T_2 \) and a side-effect \( t_2 \mapsto t_2' \in \delta_2 \) with \( t_2 \) of the form \( f(t_1', \ldots, t_n') \) (resp. \( p(t_1', \ldots, t_n') \mapsto b' \in \delta_2 \)).

- and conversely, that is this condition on side-effects has also to be satisfied by replacing \( T_1 \) by \( T_2 \), \( \delta_1 \) by \( \delta_2 \) and \( \delta_2 \) by \( \delta_1 \).

The synchronous product of \( S_1 \) and \( S_2 \), noted \( S_1 \otimes S_2 \), is the transition system \((Q, T)\) over \( \mathcal{L} = (\Sigma_1 \cup \Sigma_2, V_1 \cup V_2, A_1 \cup A_2) \) defined as follows:

- \( Q = Q_1 \times Q_2 \)
- if \( a \in A_1 \cap A_2 \), \((q_1, a, \varphi_1, \delta_1, q_1') \in T_1 \) and \((q_2, a, \varphi_2, \delta_2, q_2') \in T_2 \) then \( ((q_1, q_2), a, \varphi_1 \wedge \varphi_2, \delta_1 \cup \delta_2, (q_1', q_2')) \in T \)
- if \( a \in A_1 \setminus A_2 \) and \((q_1, a, \varphi_1, \delta_1, q_1') \in T_1 \) then for every \( q_2 \in Q_2 \), \((q_1, q_2), a, \varphi_1, \delta_1, (q_1', q_2')) \in T \)
• if \( a \in A_2 \setminus A_1 \) and \((q_2, a, \varphi_2, \delta_2, q'_2) \in T_2\) then for every \( q_1 \in Q_1 \), \((q_1, q_2), a, \varphi_2, \delta_2, (q_1, q'_2) \in T\)  

Both conditions on side-effects allow us to remove the case where for an identical function name \( f \) (resp. a predicate \( p \)) applied to an identical tuple of arguments yields different values, and then causes the functionality of \( f \) (resp. makes inconsistent the set of side-effects resting on \( p \)) to fail.

By following the notions of our abstract framework, the synchronous product gives rise to the connector \( Sync \). To define this connector, we consider the shape \( I \) composed of three nodes \( i, j \) and \( k \) and two arrows \( a_1 : i \rightarrow j \) and \( a_2 : i \rightarrow k \). The connector \( Sync \) is then defined for every diagram \( \delta \) where \( \delta(i) \) is the empty transition system over the signature \((\Sigma_0, A_i)\) where \( \Sigma_0 \) is the empty FOL-signature, \( \delta(j) = (Q_j, T_j) \) over the signature \((\Sigma_j, A_j)\), and \( \delta(k) = (Q_k, T_k) \) over the signature \((\Sigma_k, A_k)\), and yields \( Sync(\delta) = \delta(j) \circ \delta(k) \) over the signature \((\Sigma, A)\) with the co-cone \( p : Sig \circ \delta \rightarrow (\Sigma, A) \) which is the pushout of \( Sig(\delta(a_1)) \) and \( Sig(\delta(a_2)) \) in \( Sig \).

5.2 Results

The synchronous product of two transition systems \( S_1 \otimes S_2 \) have generally true emergent properties. The reason is the set \( Mod(Th(S_1^* \cup S_2^*)) \) of Kripke frames may be greater than \( Real(S_1 \otimes S_2) \). Indeed, Kripke frames in \( Real(S_1 \otimes S_2) \) have to preserve the shape of the transition system \( S_1 \otimes S_2 \) unlike Kripke frames in \( Mod(Th(S_1^* \cup S_2^*)) \). Hence, properties in \((S_1 \otimes S_2)^*\) may be more numerous than in \( Th(S_1^* \cup S_2^*) \). However, we can show under some conditions that non-conformity properties cannot occur along synchronous product. More precisely, we are going to show that the “only if” part of the conformity property is satisfied but the “if” part only holds when formulas that label transitions are conditional equations (i.e. expressed in the logic CEL).

Let us start by showing that the semantic consequences of \( S_1 \) and \( S_2 \) are preserved by \( S_1 \otimes S_2 \). Let us suppose a \( S_1 \otimes S_2 \)-model \((W, R)\), and let us define a \( L \)-model \((W_i, R_i)\) for \( i = 1, 2 \) as follow:

• for every \( q \in Q_i \), \( W_i^q = Mod(\Sigma_i \hookrightarrow \Sigma)(W(q, q')) \) for any \( q' \in Q_j \) with \( j \neq i \) in \( \{1, 2\} \)

• \( \forall a \in A_i, \ R_a = \{(q, q') \mid \exists \varphi \in Sen(\Sigma_i), \exists \delta \in SE(L), (q, a, \varphi, \delta, q') \in T_i \} \)

Let us note \( \Gamma_i \) for \( i = 1, 2 \), the set of all these \( L \)-models.

Theorem 3 Each \((W_i, R_i) \in \Gamma_i \) is a \( S_i \)-model.

Proof. The first condition of Definition 22 is obvious. To prove the second condition, let us suppose a transition \((q, a, \varphi, q') \in T_i \). By construction, there exists a transition \(((q, q_1), a, \varphi', (q', q_1')) \in T \) such that either \( \varphi' = \varphi \) and \( \delta' = \delta \), or \( \varphi' = \varphi \land \varphi'' \) and \( \delta' = \delta \lor \delta'' \). In both cases, by hypothesis, we have that \( \{W(q, q_i) \} \models \varphi \). Therefore, by the satisfaction condition for FOL \( \{W_i \} \models \varphi \). Moreover, by the condition on side-effects in Definition 23, we have that \( \{W_i \} \models \varphi \).

Proposition 2 \( \forall i : V \rightarrow W_i \), \( \{W_i, R_i \} \models \varphi \) if and only if \( \{W_i \} \models \varphi \).

Proof. By induction on the structure of \( \varphi \).

Basic case. \( \varphi \) is of the form \( p(t_1, \ldots, t_n) \). Let \( q_j \in Q_j \). By definition, there exists \((W_i, R_i) \in \Gamma_i \) such that \( W_i^q = Mod(\Sigma_i \hookrightarrow \Sigma)(W(q, q_i)) \). By hypothesis, we have \( \{W_i \} \models \varphi \) if and only if \( \varphi \).

General case. Let us handle the case where \( \exists a, \varphi' \).

Let us suppose that \((W, R) \models \varphi \). Then, let us consider \((q, q') \) such that \( (q, q_1) \models \varphi \). By the hypothesis, we have for every \((W_i, R_i) \in \Gamma_i \) that \( \{W_i \} \models \varphi \). By construction, we also have \( q_i, q' \) \( \varphi \) for every \((W_i, R_i) \in \Gamma_i \). Therefore, for every \((W_i, R_i) \in \Gamma_i \), \( (W_i, R_i) \models \varphi \), and then by the induction hypothesis, we have \((W, R) \models \varphi \).

The cases of Boolean connectives and quantifier are simpler and left to the interested reader.

Theorem 4 \((S_1^* \cup S_2^*)^* \subseteq (S_1 \otimes S_2)^* \).

Proof. Let \( \varphi \in S_1^* \), and let \((W, R) \in Real(S_1 \otimes S_2) \). Let \( i : V \rightarrow W \) be an interpretation. By Theorem 3, for every model \((W_i, R_i) \in \Gamma_i \), we have \((W_i, R_i) \models \varphi \), and then for every \( q \in Q_i \) we also have \((W_i, R_i) \models \varphi \). Therefore, by Proposition 2, we have for every \( q_j \in Q_j \) that \((W, R) \models \varphi \), and then \((W, R) \models \varphi \).

To show the “if” part of the conformity property, we need to make some restrictions on formulas that label transitions. Hence, we suppose that transition systems are built over the logic CEL, and then given a model \((W, R) \) of transition system \( S \), for each \( q \in Q, W^q \) is now an algebra. Therefore, the logic for transition systems is the modal first-order logic defined as in Section 2 except that now \( \Sigma \)-atoms are restricted to \( \Sigma \)-equations.

Given two transition systems \( S_1 \) and \( S_2 \) over the signatures \( L_1 \) and \( L_2 \), respectively, and satisfying the above restriction, for \( i \neq j \in \{1, 2\} \), and for every \((W_i, R_i) \in Mod(S_i) \) we define the mapping \( F_{(W_j, R_j)} : Mod(S_j) \rightarrow Mod(L) \) where \( L \) is the signature over which the transition system \( S_1 \otimes S_2 \) is built as follow: if we note for a \( \Sigma \)-algebra \( A \), \( th(A) = \{\varphi | \varphi \in CEL \text{-formula, } A \models \varphi \} \), then to every
(W_i, R_i), F(W_i, R_i)((W_i, R_i)) = (W, R) such that (W, R) is the L-model defined by

\[ \forall q \in Q_i, \forall q' \in Q_j, W(q',q) = T_{\Gamma_i,\Gamma_j}(W^q_i) \times T_{\Gamma_j,\Gamma_i}(W^q_j) \]

where \( \Gamma_i = th(W^q_i), \Gamma_j = th(W^q_j), and \Gamma = th(W^q_i) \cup \Gamma_j \).

**Theorem 5** For every \((W_i, R_i) \in Mod(S_i)\) and every \((W_j, R_j) \in Mod(S_j)\), \(F(W_i, R_i)((W_i, R_i)) \) is a \( S_1 \otimes S_2 \) model.

**Proof.** The first condition of Definition 20 is obvious. To prove the second condition, let us suppose a transition \( ((q_1, q_1'), (a, q_2, q_2')) \in \mathcal{T} \). By construction, \( \varphi \) and \( \delta \) are:

1. either of the form \( \varphi' \land \varphi'' \) with \( \varphi' \in Sen(S_j) \) and \( \varphi'' \in Sen(S_j) \) and \( \delta' \land \delta'' \in \mathcal{SE}(L_i) \) and \( \delta'' \in \mathcal{SE}(L_j) \).
2. or \( \varphi \in Sen(S_i) \cup Sen(S_j) \) and \( \delta \in \mathcal{SE}(L_i) \cup \mathcal{SE}(L_j) \).

This then leads to the two following cases:

1. Suppose that \( \varphi \) is of the form \( \varphi' \land \varphi'' \) and then \( \delta = \delta' \land \delta'' \). This means by construction, that \( (q_1, a, q_2, q_2) \in \mathcal{T}_1 \) and \( (q_2, a, q_1', q_1') \in \mathcal{T}_j \).

By hypothesis, we have \( W^q_i \models \varphi' \) and \( W^q_j \models \varphi'' \). Therefore, we have that \( T_{\Gamma_i,\Gamma_j}(W^q_i) \models \varphi' \land \varphi'' \) and \( T_{\Gamma_j,\Gamma_i}(W^q_j) \models \varphi' \land \varphi'' \), and then so is \( T_{\Gamma_i,\Gamma_j}(W^q_i) \times T_{\Gamma_j,\Gamma_i}(W^q_j) \) (recall that conditional equations are preserved along the cartesian product of algebras). Moreover, by hypothesis, we also have that \( W^q_i \sim_{\delta_i} W^q_j \sim_{\delta_j} W^{-q_j} \).

By definition, \( \Gamma_1 \) (resp. \( \Gamma_2 \)) contains the ground equational theory of \( W^q_i \) (resp. \( W^q_j \)). If we note \( \Gamma'_1 = th(W^q_i), \Gamma'_2 = th(W^q_j) \) and \( \Gamma' = \Gamma'_1 \cup \Gamma'_2 \), then we have \( T_{\Gamma'_1,\Gamma'_2}(W^q_i) \sim_{\delta} T_{\Gamma'_1,\Gamma'_2}(W^q_j) \) and \( T_{\Gamma'_1,\Gamma'_2}(W^q_i) \sim_{\delta} T_{\Gamma'_1,\Gamma'_2}(W^q_j) \).

2. The case where \( \varphi \in Sen(S_i) \cup Sen(S_j) \) and \( \delta' \in \mathcal{SE}(L_i) \) and \( \delta'' \in \mathcal{SE}(L_j) \) is noticeably similar to the previous one.

**MFO.** MFO is closed under isomorphism. Moreover, by Theorem 3, Real is compatible with each morphisms \( p_i \) of the co-cone \( p \) associated to the connector Sync. Finally, by Proposition 2 and Theorem 4, RSC is satisfied. Therefore, Theorem 6 is a specialization of Theorem 2.

**Theorem 6** If for every \((W_i, R_i) \in Mod(S_i)\), every \((W_j, R_j) \in Mod(S_j)\), and every \( q \in Q_i \) and every \( q' \in Q_j \), the adjunct morphism \( \mu_{W_i, W} : W_i \rightarrow Mod(S_i \rightarrow \Sigma)(T_{\Gamma_i,\Gamma_j}(W^q_i)) \) is an isomorphism, then \( (S_1 \otimes S_2)^* \cap Sen(L_i) \subseteq S_1^* \).

**Proof.** Let \( \varphi \in (S_1 \otimes S_2)^* \cap Sen(L_i) \) and let \((W_i, R_i) \in Mod(S_i)\). By Theorem 5, for every \((W_j, R_j) \in Mod(S_j)\), we have that \( F(W_i, R_i)((W_i, R_i)) \models \varphi \). As the adjunct morphism \( \mu_{W_i, W} \) is an isomorphism, for every \( \iota : V \rightarrow W \), there exists \( \tau : T_{\Gamma_i,\Gamma_j}(W^q_i) \times T_{\Gamma_j,\Gamma_i}(W^q_j) \) such that \( \iota = p_i \circ \tau \), where \( p_i \) is the i-th projection map \( p_i : T_{\Gamma_i,\Gamma_j}(W^q_i) \rightarrow T_{\Gamma_i,\Gamma_j}(W^q_i) \times T_{\Gamma_j,\Gamma_i}(W^q_j) \) for \( q \in Q_i \) and \( q' \in Q_j \). By hypothesis, for every \( q \in Q_i \) and every \( q' \in Q_j \), \( F(W_i, R_i)((W_i, R_i)) \models \varphi' \). It is then easy to show by induction on the structure of \( \varphi \) that \((W_i, R_i) \models \varphi \).

**Example 4** When dealing with formulas expressed in the logic CEL to label transitions, we often make restrictions on algebras denoting states. Indeed, to allow inductive proofs or for computability reasons, state-algebras are then restricted to reachable \( 10 \) or some quotients of the ground term algebra. Let us suppose the for the below counterexample of the conditions given in Theorem 6, that we restrict our approach to state-algebras defined by reachable algebras. Let us consider the two following transition systems \( S_1 \) and \( S_2 \) defined respectively over the two following signatures \( L_1 \) and \( L_2 \):

\[ A \Sigma \text{-algebra is reachable if, and only if the unique } \Sigma \text{-morphism } \mu : T_F \rightarrow A \text{ is surjective, that is all the values in } A \text{ are denoted by the evaluation of a ground term.} \]
\[
\begin{align*}
\Sigma_1 &= \left\{ \begin{array}{l}
S = \{ \text{nat} \}, \\
F = \{ 0 \mapsto \text{nat}; s : \text{nat} \to \text{nat}, \\
\quad + : \text{nat} \times \text{nat} \to \text{nat} \} , \\
P = \emptyset
\end{array} \right. \\
\Sigma_2 &= \left\{ \begin{array}{l}
S = \{ \text{nat} \}, \\
F = \{ 0 \mapsto \text{nat}; s, p : \text{nat} \to \text{nat} \}, \\
P = \emptyset
\end{array} \right. \\
A_1 = A_2 = \{ a \}
\end{align*}
\]

Let us define \( S_1 \) and \( S_2 \) as follows:

- \( S_1 = (\{ q_1, q_2 \}, \{ q_1 \overset{a_1, \varphi_1, \delta_1}{\rightarrow} q_2 \}) \) where \( \varphi_1 = (s(x) = s(y) \Rightarrow x = y) \land x + 0 = x \land s(x + y) = s(x + y) \) and \( \delta_1 = \emptyset \).
- \( S_2 = (\{ q'_1, q'_2, q''_1, q''_2 \}, \{ q'_1 \overset{a_2, \varphi_2, \delta_2}{\rightarrow} q''_2 \}) \) where \( \varphi_2 = (s(x) = s(y) \Rightarrow x = y) \land s(p(x)) = x \land p(s(x)) = x \) and \( \delta_2 = \emptyset \).

By definition of \( S_1 \) (resp. \( S_2 \)), the unique \( S_1 \)-model (resp. \( S_2 \)-model) is \((W, R)\) where \( W_{S_1} = W_{S_2} = \mathbb{N} \) (resp. \( W_{S_1} = W_{S_2} = \mathbb{Z} \)). On the contrary, by construction, in \( S_1 \otimes S_2 \), we have the transition \( (q_1, q'_1) \overset{a, \varphi', \delta'}{\rightarrow} (q_2, q''_2) \) where \( \varphi' = \varphi_1 \land \varphi_2 \) and \( \delta' = \emptyset \), and then all the \( S_1 \otimes S_2 \)-model satisfy \( W_{S_1} = W_{S_2} = \mathbb{Z} \). Consequently, the modal formula \( \varphi' \Rightarrow \square_{a}(\forall x, \exists y. x + y = 0) \) belongs to \( (S_1 \otimes S_2)^* \) but not in \( S_1^* \). The reason is that \( F(\mathcal{W}_2, r_2)(W_{S_1}^0) = \emptyset \). Therefore, the adjunct functor \( \mu_{W_{S_1}} \) is injective but not surjective, and then it is not an isomorphism.

6 Conclusion

In this paper, our main contribution is twofold. First, we have formally defined the notion of emergent properties independently of formalism, and of the form of both specifications and architectural connectors. Secondly, we have studied in this abstract framework, some general conditions that enable us to obtain two general properties that guarantee when a system is not complex. These conditions are based on the category theory of morphism conservativeness and adjunction. Finally, to illustrate our abstract framework, we have instantiated our abstract framework with reactive component-based systems described by transition systems and combined together through synchronous product, and we have applied our general results to obtain global systems lacking of non-conformity properties which have been recognized as being the cause of bad interactions between components.

An ongoing research that we are currently pursuing is to extend abstract connectors to heterogeneous abstract connectors, that is connectors defined on component specifications described in heterogeneous formalisms. For this purpose, we will take benefit from [11, 23] and from works that we made on hierarchical heterogeneous specifications [9].

References


