

Complex software systems: Formalization and Applications

Marc Aiguier, Pascale Le Gall, Mbarka Mabrouki

▶ **To cite this version:**

Marc Aiguier, Pascale Le Gall, Mbarka Mabrouki. Complex software systems: Formalization and Applications. International Journal On Advances in Software, 2009, 2 (1), pp.47-62. hal-00782879

HAL Id: hal-00782879

https://centralesupelec.hal.science/hal-00782879

Submitted on 11 Apr 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Complex software systems : Formalization and Applications *

Marc Aiguier, Pascale Le Gall and Mbarka Mabrouki
École Centrale Paris
Laboratoire de Mathématiques Appliqués 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 [1, 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, 2, 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 to the global system.

Following some works issued from other scientific disciplinaries 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

*This work is performed within the European project GENNETEC *(GENetic NeTworks: Emergence and Complexity)* STREP 34952.

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 $X^\bullet$, and $F(X) + F(Y) = (X^\bullet \cup Y^\bullet)^\bullet$. 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 $X^\bullet$, resp. $Y^\bullet$, 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 (Σ, Ax) where Σ is a signature and Ax is a (finite) set of formulas (axioms) over Σ). 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 Vanbremeersch'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 [4] 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 [3]. Here, this manuscript also extends the paper published in [3] 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 $\mathcal{I} = (Sig, Sen, Mod, \models)$ *consists of*

- *a category Sig, objects of which are called* signatures,
- *a functor $Sen : Sig \to Set$ giving for each signature a set, elements of which are called* sentences,
- *a contravariant functor $Mod : Sig^{op} \to Cat$ giving for each signature a category, objects and arrows of which are called* Σ-models *and* Σ*-morphisms respectively, and*
- *a $|Sig|$-indexed family of relations*

$$\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$$

called satisfaction relation,

such that the following property holds:
$\forall \sigma : \Sigma \to \Sigma',\ \forall \mathcal{M}' \in |Mod(\Sigma')|,\ \forall \varphi \in Sen(\Sigma)$,

$$\mathcal{M}' \models_{\Sigma'} Sen(\sigma)(\varphi) \Leftrightarrow Mod(\sigma)(\mathcal{M}') \models_\Sigma \varphi$$

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

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

This means that $\mathcal{M}_1$ and $\mathcal{M}_2$ 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 $\mathcal{I} = (Sig, Sen, Mod, \models)$ be an institution. Let Σ be a signature of $|Sig|$. Let T be a set of Σ-sentences (i.e. $T \subseteq Sen(\Sigma)$). Let us denote $Mod(T)$ the full sub-category of $Mod(\Sigma)$ whose objects are all Σ-models $\mathcal{M}$ such that for any $\varphi \in T$, $\mathcal{M} \models_\Sigma \varphi$, and $T^\bullet$ the subset of $Sen(\Sigma)$, so-called* semantic consequences of T*, defined as follows:*

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

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

$\varphi \in T^\bullet$ is also denoted by $T \models_\Sigma \varphi$.

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 Σ, the set of Σ-sentences is the least set of sentences finitely built over propositional variables in Σ and Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$. Given a signature morphism $\sigma : \Sigma \to \Sigma'$ associating to each propositional variable of Σ a propositional variable of Σ', $Sen(\sigma)$ translates Σ-formulas to Σ'-formulas by renaming propositional variables according to σ.
Given a signature Σ, the category of Σ-models is the category of mappings[1] $\nu : \Sigma \to \{0, 1\}$ with identities as morphisms. Given a signature morphism $\sigma : \Sigma \to \Sigma'$, the forgetful functor $Mod(\sigma)$ maps a Σ'-model ν' to the Σ-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 $S^* \times S$ and S^+ respectively.[2]
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 Σ-atoms are of two possible forms: $t_1 = t_2$ where [3] $t_1, t_2 \in T_F(X)_s$ ($s \in S$), and $p(t_1, \ldots, t_n)$ where $p : s_1 \times \ldots \times s_n \in P$ and $t_i \in T_F(X)_{s_i}$ ($1 \leq i \leq n$, $s_i \in S$). The set of Σ-sentences is the least set of formulas built over the set of Σ-atoms by finitely applying Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$ and quantifiers $\forall$ and $\exists$.
Given a signature $\Sigma = (S, F, P)$, a Σ-model $\mathcal{M}$ is a family $M = (M_s)_{s \in S}$ of sets (one for every $s \in S$), each one equipped with a function $f^{\mathcal{M}} : M_{s_1} \times \ldots \times M_{s_n} \to M_s$ 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 Σ'-model $\mathcal{M}'$, $Mod(\sigma)(\mathcal{M}')$ is the Σ-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 Σ in **FOL** is a Σ-sentence of the form $\Gamma \Rightarrow \alpha$ where Γ is a finite conjunction of Σ-atoms and α is a Σ-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 $\Sigma = (S, F)$, Σ-sentences are formulas of the form $\varphi : t_1 \to t'_1 \wedge \ldots \wedge t_n \to t'_n \Rightarrow t \to t'$ where $t_i, t'_i \in T_F(X)_{s_i}$ ($1 \leq i \leq n$, $s_i \in S$) and $t, t' \in T_F(X)_s$ ($s \in S$). Models of rewriting logic are preorder models, *i.e.* given a signature $\Sigma = (S, F)$, $Mod(\Sigma)$ is the category of Σ-algebras $\mathcal{A}$ such that for every $s \in S$, A_s is equipped with a preorder $\geq$. Hence, $\mathcal{A} \models \varphi$ if, and only if for every variable assignment $\nu : X \to A$, if each $\nu(t_i)^{\mathcal{A}} \geq \nu(t'_i)^{\mathcal{A}}$ then $\nu(t)^{\mathcal{A}} \geq \nu(t')^{\mathcal{A}}$ where $_^{A} : T_F(A) \to A$ is the mapping inductively defined by: $f(t_1, \ldots, t_n)^A = f^{\mathcal{A}}(t_1^A, \ldots, t_n^A)$.

2.2.3 Modal FOL (MFOL)

Signatures are couples (Σ, A) where Σ 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.

[1] $\{0, 1\}$ are the usual truth-values.
[2] S^+ is the set of all non-empty sequences of elements in S and $S^* = S^+ \cup \{\epsilon\}$ where ϵ denotes the empty sequence.
[3] $T_F(X)_s$ is the term algebra of sort s built over F with sorted variables in a given set X.

Given a **MFOL** signature (Σ, A) with $\Sigma = (S, F, P)$, (Σ, A)-atoms are either predicates $p(t_1, \ldots, t_n)$ or the symbol T (for True), and the set of (Σ, A)-formulas is the least set of formulas built over the set of (Σ, A)-atoms by finitely applying Boolean connectives in $\{\neg, \vee, \wedge, \Rightarrow\}$, quantifiers $\forall$ and $\exists$, and modalities in $\{\Box_a | a \in A\}$. For every $a \in A$, the intuitive meaning of $\Box_a$ is "always after the action a". Given a signature (Σ, A), a (Σ, A)-model (W, R), called Kripke frame, consists of a family $W = (W^i)_{i \in I}$ of Σ-models in **FOL** (the *possible worlds*) such that [4] $W^i_s = W^j_s$ 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 $\sigma : (\Sigma, A) \to (\Sigma', A')$ and a (Σ', A')-model $((W'^i)_{i \in I}, R')$, $Mod(\sigma)(((W'^i)_{i \in I}, R'))$ is the (Σ, A)-model $(Mod(\sigma)(W'^i)_{i \in I}, R)$ defined for every $a \in A$ by $R_a = R'_{\sigma(a)}$. A (Σ, A)-sentence φ is said to be satisfied by a (Σ, A)-model (W, R), noted $(W, R) \models_{(\Sigma, A)} \varphi$, if for every $i \in I$ we have $(W, R) \models^i_\Sigma \varphi$, where $\models^i_\Sigma$ is inductively defined on the structure of φ as follows:

- for every **FOL**-formula φ built over Σ-atoms, $(W, R) \models^i_\Sigma \varphi$ iff $W^i \models_\Sigma \varphi$
- $(W, R) \models^i_\Sigma \Box_a \varphi$ when $(W, R) \models^j_\Sigma \varphi$ 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 Set. Given a set A, the set of sentences is A^* and $Mod(A)$ is the category whose objects are all subsets of A^*. Given a signature morphism $\sigma : A \to A'$, $Mod(\sigma)$ is the functor which at $L' \subseteq A'^*$ associates the set $L = \{\alpha | \sigma(\alpha) \in L'\}$. Finally, given a signature $\Sigma \in Sig$, $\models_\Sigma$ is just the membership relation $\ni$. 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.

[4] In the literature, Kripke frames satisfying such a property are said *with constant domains*.

3 Specifications in institutions

Over institutions, specifications are usually defined either by logical theories or couples (Σ, Ax) where Σ is a signature and Ax a set (usually finite) of formulae (often called axioms) over Σ. 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 $\mathcal{I} = (Sig, Sen, Mod, \models)$.

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

- $Spec : Sig^{op} \to Set$ *is a functor. Given a signature Σ, elements in $Spec(\Sigma)$ are called* specifications over Σ.
- $Real = (Real_\Sigma)_{\Sigma \in |Sig|}$ *is a Sig-indexed family of mappings $Real_\Sigma : Spec(\Sigma) \to |Cat|$ such that for every $\Sigma \in |Sig|$, and every $Sp \in Spec(\Sigma)$, $Real_\Sigma(Sp)$ is a full subcategory of $Mod(\Sigma)$. Objects of $Real_\Sigma(Sp)$ are called* realizations of Sp.

Definition. 6 (Semantic consequences) *Let $\mathcal{SL} = (Spec, Real)$ be a specification language over $\mathcal{I}$. Let us define $_^\bullet = (_^\bullet_\Sigma)_{\Sigma \in Sig}$ the Sig-indexed family of mappings $_^\bullet_\Sigma : Spec(\Sigma) \to \mathcal{P}(Sen(\Sigma))$ that to every $Sp \in Spec(\Sigma)$, yields the set $Sp^\bullet_\Sigma = \{\varphi | \forall \mathcal{M} \in Real_\Sigma(Sp), \mathcal{M} \models_\Sigma \varphi\}$. $Sp^\bullet_\Sigma$ is called the* set of semantic consequences *of Sp or the* theory of Sp.

Definition 6 calls for some comments:

- We could expect that $Mod(Sp^\bullet) = Real(Sp)$ what would make unmeaning the existence of the mappings in $Real$ 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, $_^\bullet$ is a natural transformation from $Spec$ to[5] $\mathcal{P} \circ Sen^{op}$. However, most of times, it is not the case (see the examples in Section 3.2).

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

Hence, specification morphisms are arrows in $SPEC$ that further preserve semantic consequences. Commonly, the category of specifications over institutions have $\bigcup_{\Sigma \in |Sig|} Spec(\Sigma)$ 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, this gives rise to the functor $Spec : Sig^{op} \to Set$ which to every $\Sigma \in Sig$, associates the set of all Σ-theories T, and to every signature morphism $\sigma : \Sigma \to \Sigma'$, matches every Σ'-theory T' with the Σ-theory $T = \{\varphi | Sen(\sigma)(\varphi) \in T'\}$. Hence, $Spec(\Sigma) \subseteq \mathcal{P}(Sen(\Sigma))$. We naturally define $Real_\Sigma(T) = Mod(T)$. Moreover, specifications being saturated theories, this naturally leads to the identity function $_^\bullet_\Sigma : Spec(\Sigma) \to \mathcal{P}(Sen(\Sigma))$. It is easy to check that given a signature morphism $\sigma : \Sigma \to \Sigma'$, the following diagram commutes and then $_^\bullet$ is a natural transformation:

$$\begin{array}{ccc} Spec(\Sigma) & \xrightarrow{-^\bullet_\Sigma} & \mathcal{P}(Sen(\Sigma)) \\ \uparrow{\scriptstyle Spec(\sigma)} & & \uparrow{\scriptstyle \mathcal{P}(Sen^{op}(\sigma^*))} \\ Spec(\Sigma') & \xrightarrow{-^\bullet_{\Sigma'}} & \mathcal{P}(Sen(\Sigma')) \end{array}$$

(See Footnote 5 for the definition of σ^*)

3.2.2 Axiomatic specifications

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

$$x\ R_2\ y \Longrightarrow y\ R_2\ x$$
$$x\ R_1\ y \Longleftrightarrow x\ R_2\ y$$

Obviously, we prove from Ax' that R_1 is a symmetric relation.
However, $Spec(\sigma)((\Sigma', Ax')) = \emptyset$, and then $Spec(\sigma)((\Sigma', Ax'))^\bullet$ is restricted to tautologies while $\mathcal{P}(Sen^{op}(\sigma^*))(Ax')$ 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 α_n. Hence, a specification over an alphabet A is a set R of n-ary relations on A^*. Given a signature morphism $\sigma : A \to A'$ and a specification R' over A', the specification $Spec(\sigma)(R')$ over A is the set R of n-ary relation r such that there exists $r' \in R'$ and $r = \{(a_1, \ldots, a_n) | (\forall i, 1 \leq i \leq n, a_i \in A) \wedge (a_1, \ldots, a_n) \in r'\}$. Given a set of inference rules R over an alphabet A, $R^\bullet_A$ is the language L inductively generated from inference rules of R. Given a signature morphism $\sigma : A \to A'$ and a set of inference rules R' over A'. It is easy to show that $Spec(\sigma)(R')^\bullet_A = R'^\bullet_{A'} \cap A^*$ what proves that $_^\bullet$ is a natural transformation from $Spec$ to $\mathcal{P} \circ Sen^{op}$.

[5] Given a functor $F : \mathcal{C} \to \mathcal{D}$, $\mathcal{F}^{op} : \mathcal{C}^{op} \to \mathcal{D}^{op}$ is the dual of $\mathcal{F}$ defined as follows:

- $\forall o \in \mathcal{C}, F^{op}(o) = F(o)$
- f^* being the reverse arrow of f in $\mathcal{C}$, $\forall o, o' \in \mathcal{C}, \forall f \in Hom_\mathcal{C}(o, o'), F^{op}(f^*) = F(f)^*$

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

3.3 Properties of specifications

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

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

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

Definition. 8 (Compatible) *Let $\mathcal{SL} = (Spec, Real)$ be a specification language over $\mathcal{I}$. Let $\sigma : Sp \to Sp'$ be a specification morphism. $Real$ is said* compatible *with σ if, and only if we can define a functor $Real(\sigma) : Real(Sp') \to Real(Sp)$.*

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

Definition. 9 (Definable by specification) *Given an institution $\mathcal{I}$ and a specification language over $\mathcal{I}$, a Σ-theory T is said* definable by specification *or* definable *for being shorter if, and only if there exists $Sp \in Spec(\Sigma)$ such that $T = Sp^{\bullet}_{\Sigma}$.*

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 $\mathcal{SL}$ over $\mathcal{I}$, a specification morphism $\sigma : Sp \to Sp'$ is* liberal *if, and only if $Real$ is compatible with σ and $Real(\sigma) : Real(Sp') \to Real(Sp)$ has a left-adjunct $\mathcal{F}(\sigma) : Real(Sp) \to Real(Sp')$.*

Specifications defined by logical theories and axiomatic specifications over the institution **CEL** is liberal for every specification morphism σ. Indeed, let $\sigma : \Sigma = (S, F) \to \Sigma' = (S', F')$ be a signature morphism, and let Γ and Γ' be two sets of conditional equations over, respectively, Σ and Σ' such that $Sen(\sigma)(\Gamma) \subseteq \Gamma'$. We can build a functor $T_{\Gamma'/\Gamma} : \mathcal{A} \mapsto T_{\Gamma'/\Gamma}(\mathcal{A})$, from the category of Γ-algebras to the category of Γ'-algebras.
Let $\mathcal{A}$ be a Γ-algebras. $T_{\Gamma'/\Gamma}(A)$ is the quotient of $T_{F'}(\mathcal{A})$ by the congruence generated by the kernel of the Σ-morphism $T_F(A)$ in $\mathcal{A}$ extending the identity on X. [6] This algebra satisfies the following universal property: for every Γ'-algebra $\mathcal{B}$ and every Σ-morphism $\mu : \mathcal{A} \to Mod(\sigma)\mathcal{B}$, there exists a unique Σ'-morphism $\eta_{\mathcal{B}} : T_{\Gamma'/\Gamma}(\mathcal{A}) \to \mathcal{B}$ such that for every $a \in A$, $\eta_{\mathcal{B}}(a) = \mu(a)$. This universal property directly shows that the functor $T_{\Gamma'/\Gamma}$ is left-adjunct to $Mod(\sigma)$, i.e., for every Γ-algebra $\mathcal{A}$ there exists a universal morphism $\mu_{\mathcal{A}} : \mathcal{A} \to Mod(\sigma)(T_{\Gamma'/\Gamma}(\mathcal{A}))$. $\mu_{\mathcal{A}}$ is called the *adjunct 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 Sp_1, Sp_2, $\ldots Sp_n$ defined respectively over the signatures Σ_1, Σ_2, $\ldots \Sigma_n$, to build a new one, denoted $Sp = c(Sp_1, Sp_2, \ldots, Sp_n)$. We can then see the connector c as a mapping of arity n from $|SPEC|^n$ to $|SPEC|$. 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 $(Sp_1, Sp_2, \ldots, Sp_n)$ 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 Sp_1, Sp_2, $\ldots Sp_n$ 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 $SPEC$. In practice, for a given connector c, all the diagrams accepted as arguments by

[6] $T_{F'}(A)$ (resp. $T_F(A)$) is the term algebra built over F' (resp. F) with sorted variables in the carrier A of the Γ-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 Σ of Sp is the least one over the signatures Σ_1, Σ_2, ..., Σ_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 Σ_1, Σ_2 ... Σ_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 δ be a diagram of $\Delta_{(I,C)}$. Let us denote $\delta_{|_{I'}}$ the diagram of $\Delta_{(I',C)}$ obtained by restricting δ to I'.

Definition. 11 (Co-cone) *Given a diagram $\delta : I \to C$. A* co-cone *of δ consists of an object $c \in |C|$ and a I-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 δ 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 δ, 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 $\bullet \leftarrow \bullet \rightarrow \bullet$ 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 $\mathcal{SL}$ 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 $\mathcal{I} = (Sig, Sen, Mod, \models)$. Moreover, in Example 1, $SPEC$ is the category whose objects are specifications of the form (Σ, 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 $Sen(Sig(\delta(a)))(Ax) \subseteq Ax'$, and yields $Enrich(\delta) = (\Sigma', Ax')$ together with the co-cone $Sig(\delta(a))$ and $Id_{Sig(\delta(j))}$ which is the obvious co-limit of $Sig \circ \delta$. Observe that $\delta(a)$ and $Id_{\delta(j)}$ 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 $a_2 : i \to k$. 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 $Sen(Sig(\delta(a_1)))(Ax_0) \subseteq Ax_1$ and $Sen(Sig(\delta(a_2)))(Ax_0) \subseteq Ax_2$, and yields $Union(\delta) = (\Sigma, Ax)$ with the co-cone $p : Sig \circ \delta \to \Sigma$ which is the pushout of $Sig(\delta(a_1))$ and $Sig(\delta(a_2))$ and such that $Ax = Sen(p_j)(Ax_1) \cup Sen(p_k)(Ax_2)$. Observe that we can derive the co-cone $p_{SPEC} : \delta \to (\Sigma, Ax)$ such that $Sig \circ p_{SPEC} = p$, and p_{SPEC_j} and p_{SPEC_k} are specification morphisms.*

In [8], both above connectors have been brought down to two basic connectors: union with constant signatures $\bigcup$, and **translate _ by** *σ for every signature morphism σ. 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 $p : Sig \circ \delta \to \Sigma$ where p_i and p_j are the identity signature morphism for Σ, and such that $Ax = Ax_1 \cup Ax_2$.*

2. *Let I be the graph composed two nodes k and l. The connector* **translate _ by** *σ where $\sigma : \Sigma \to \Sigma'$ is a signature morphism, is defined for every diagram $\delta : I \to SPEC$ where $\delta(k) = (\Sigma, Ax)$ and $\delta(l) = (\Sigma', Sen(\sigma)(Ax))$, and yields* **translate _ by** *$\sigma(\delta) = (\Sigma', Sen(\sigma)(Ax))$.*

In [8], $\bigcup(\delta)$ and **translate _ by** *$\sigma(\delta)$ are respectively noted $\delta(i) \bigcup \delta(j)$ and* **translate** *$\delta(k)$* **by** *σ.*

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

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

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

Let us denote $c' \circ_{i'} c : |\Delta_{I' \circ_{i'} I, SPEC}| \to |SPEC|$ the architectural connector defined by : [7]

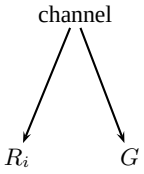
$$\delta \mapsto \begin{cases} c'(\delta_{|_{I'}}) & \text{if } c(\delta_{|_I}) \text{ is defined} \\ & \delta(i') = c(\delta_{|_I}) \\ & \text{and } \delta(q_i) \text{ is the morphism } r_i \text{ in } SPEC \\ & \text{whose the image by } Sig \text{ is the component} \\ & p_i \text{ of the co-limit } p \text{ associated to } c(\delta_{|_I}) \\ undefined & otherwise \end{cases}$$

Example. 2 *Enrichment can be removed and replaced by the following combination of* **translate** *and $\cup$ as follows: let δ be a diagram of $\Delta_{(I,SPEC)}$ where I is the index category of the connector $Enrich$, $\delta(i) = (\Sigma, Ax)$ and $\delta(j) = (\Sigma', Ax')$*

$$Enrich(\delta) = \bigcup \circ_i \textbf{translate_by} \delta'(p_i)(\delta')$$

where δ' is the diagram of $\Delta_{(I'' \circ_i I', SPEC)}$ for I'' (resp. I') the index category of the connector $\cup$ (resp. **translate***), defined by: $\delta'(k) = \delta(i)$, $\delta'(i) =$* **translate** *$\delta'(k)$* **by** *$\delta'(p_i) = (\Sigma', Sen(Sig(p_i))(Ax))$ and $\delta'(j) = (\Sigma', Ax' \setminus Ax)$.*

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 $c : |\Delta_{(I,SPEC)}| \to |SPEC|$ be an architectural connector. Let δ be a diagram of $\Delta_{(I,SPEC)}$ such that $c(\delta)$ is defined. c is said* complex for *δ if, and only if one of the two following properties fails:*

1. Conformity.
   $\forall i \in I, \forall \varphi \in Sen(Sig(\delta(i))), \varphi \in \delta(i)^{\bullet}_{Sig(\delta(i))} \iff Sen(p_i)(\varphi) \in c(\delta)^{\bullet}_{Sig(c(\delta))}$

[7] q_i is the arrow introduced in $Hom_{I' \circ_{i'} I}(i, i')$.

2. Non true emergence.

$$\forall \varphi \in c(\delta)^{\bullet}_{Sig(c(\delta))}, \bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}) \models_{Sig(c(\delta))} \varphi$$

A formula φ that makes fail the equivalence of both Point 1. and Point 2. is called emergent property.
If c is not complex for a diagram δ, then it is said modular.

Example. 3 *Here, we give a very simple example of specifications in which modularity fails. Let* **Nat** *be the specification in* **EQL** *defined as follows:*

Specification of Nat Sorts: $S_{Nat} = \{ nat \}$

Functions : $F_{Nat} =$
$\{0 :\rightarrow nat,$
$succ : nat \rightarrow nat,$
$_ + _ : nat \times nat \rightarrow nat \}$

Axioms: $Ax_{Nat} =$
$\{x + 0 = x$
$x + succ(y) = succ(x + y)\}$

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

Sorts: $S_{Stack} = \{ nat, stack \}$

Functions : $F_{Stack} = F_{Nat} \cup$
$\{empty :\rightarrow stack,$
$push : nat \times stack \rightarrow stack,$
$pop : stack \rightarrow stack,$
$top : stack \rightarrow nat,$
$high : stack \rightarrow nat\}$

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

If we suppose that realizations are either the initial model or reachable models [8] *of both specifications, then an example of emergent property is:*

$$\forall x, (x = 0) \vee (\exists y, x = succ(y))$$

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

[8] A model is reachable when any of its values is the result of the evaluation of a ground term.

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 δ comes to show that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$.

Theorem. 1 *Let c be an architectural connector and δ be a diagram such that $c(\delta)$ is defined. Then, we have:*

$(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$ if, and only if the set of true emergent properties is empty and each p_i is a specification morphism.

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

The if *part. As each p_i of p is a specification morphism, we have that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet} \subseteq c(\delta)^{\bullet}_{Sig(c(\delta)}$. Moreover, as the set of true emerging properties is empty, we have that $c(\delta)^{\bullet}_{Sig(c(\delta)} \subseteq (\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$. Hence, $c(\delta)^{\bullet}_{Sig(c(\delta)} = (\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$, and then $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$.*

By Theorem 1, the architectural connectors $Enrich$, $Union$, $\bigcup$ and **translate _ by** σ 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

$(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ is definable by $c(\delta)$ for a connector c and a diagram δ 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 $\Delta_{(I,SPEC)}$ satisfies the supplementary condition, so-called *Right Satisfaction Condition* (RSC) : $\forall \varphi \in Sen(Sig(\delta_i)), \forall \mathcal{M} \in Real(c(\delta)), Real(p_i)(\mathcal{M}) \models_{Sig(\delta_i)} \varphi \implies \mathcal{M} \models_{Sig(\delta(c))} Sen(p_i)(\varphi)$.

The interest of RSC is, realizations being a subset of models, some pruning on realizations in $Real(\delta(c))$ 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 δ be a diagram such that $c(\delta)$ is defined. Suppose that $(\bigcup_{i\in I} Sen(p_i)(\delta(i)^{\bullet}_{Sig(\delta(i))}))^{\bullet}$ 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 $\mathcal{M} \in Real(\delta(i))$, If each adjunct morphism $\mu_{\mathcal{M}} : \mathcal{M} \to Real(p_i)(\mathcal{F}(p_i)(\mathcal{M}))$ is an isomorphism, then the set of non-conformity properties is empty.*

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

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

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 $\mathscr{L} = (\Sigma, V, A)$ where: Σ is a* **FOL***-signature, V is a set of variables over Σ and A is a set whose elements are called* actions.

Definition. 18 (Side-effect) *Given a signature* $\mathscr{L} = (\Sigma, V, A)$ *where* $\Sigma = (S, F, P)$*, a* side-effect *over* $\mathcal{L}$ *is a pair of ground terms over* Σ (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* Σ*-atom (i.e. each* t_i *is a ground term) and* b *is equal to* $true$ *or* $false$*. In the sequel, a side-effect* (t, t') *will be noted* $t \mapsto t'$*.*
We note $\mathcal{SE}(\mathcal{L})$ *the set of side-effects over* $\mathcal{L}$*.*

A transition system is then defined as follows:

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

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

A small specification example is given in [3]. Transition systems are specifications of reactive systems. Given a signature morphism $\sigma : (\Sigma, A) \rightarrow (\Sigma', A')$ and a specification $\mathcal{S}' = (Q', \mathbb{T}')$ over (Σ', A'), $Spec(\sigma)(\mathcal{S}')$ is the specification $\mathcal{S} = (Q, \mathbb{T})$ over (Σ, A) such that $Q = Q'$ and $\mathbb{T} = \{(q, a, \varphi, \delta, q') | (q, \sigma(a), Sen(\sigma)(\varphi), \sigma(\delta), q') \in \mathbb{T}'\}$.

5.1.2 Semantics

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

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

- $\mathcal{W}$ *is a* I*-indexed set* $(\mathcal{W}^i)_{i \in I}$ *of* Σ*-models such that* $W^i_s = W^j_s$ *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 Σ-models. Therefore, side-effects will consist on moving from a Σ-model to another one by changing the semantics of functions according the assignments given in the set δ of transitions. Formally, this is defined as follow: if $\mathcal{A}$ is a Σ-model, then $_^{\mathcal{A}} : T_F \rightarrow \mathcal{A}$ is the Σ-morphism inductively defined by $f(t_1, \ldots, t_n) \mapsto f^{\mathcal{A}}(t_1^{\mathcal{A}}, \ldots, t_n^{\mathcal{A}})$

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

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

1. $(q, a, \varphi, \delta, q') \in \mathbb{T} \wedge \mathcal{W}^q \models \varphi \wedge \mathcal{W}^q \rightsquigarrow_\delta \mathcal{W}^{q'}$
$\Rightarrow q \; R_a \; q'$
2. $q \; R_a \; q'$
$\Rightarrow \exists (q, a, \varphi, \delta, q') \in \mathbb{T}, \mathcal{W}^q \models \varphi \wedge \mathcal{W}^q \rightsquigarrow_\delta \mathcal{W}^{q'}$

Hence, the way whose dynamic is dealt with in this paper follows the state-as-algebra style [21, 1] where states are Σ-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* $\mathcal{S}_i = (Q_i, \mathbb{T}_i)$ *be a transition system over a signature* $\mathscr{L}_i = (\Sigma_i, V_i, A_i)$ *with* $i = 1, 2$ *such that:*

- *for every transition* $(q_1, a, \varphi_1, \delta_1, q_1') \in \mathbb{T}_1$ *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* $(q_2, a, \varphi_2, \delta_2, q_2') \in \mathbb{T}_2$ *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* $\mathbb{T}_1$ *by* $\mathbb{T}_2$*,* δ_1 *by* δ_2 *and* δ_2 *by* δ_1.

The synchronous product *of* $\mathcal{S}_1$ *and* $\mathcal{S}_2$*, noted* $\mathcal{S}_1 \otimes \mathcal{S}_2$*, is the transition system* $(Q, \mathbb{T})$ *over* $\mathscr{L} = (\Sigma_1 \cup \Sigma_2, V_1 \cup V_2, A_1 \cup A_2)$ *defined as follows:*

- $Q = Q_1 \times Q_2$
- *if* $a \in A_1 \cap A_2$*,* $(q_1, a, \varphi_1, \delta_1, q_1') \in \mathbb{T}_1$ *and* $(q_2, a, \varphi_2, \delta_2, q_2') \in \mathbb{T}_2$ *then*
$((q_1, q_2), a, \varphi_1 \wedge \varphi_2, \delta_1 \cup \delta_2, (q_1', q_2')) \in \mathbb{T}$
- *if* $a \in A_1 \setminus A_2$ *and* $(q_1, a, \varphi_1, \delta_1, q_1') \in \mathbb{T}_1$ *then for every* $q_2 \in Q_2$*,* $((q_1, q_2), a, \varphi_1, \delta_1, (q_1', q_2)) \in \mathbb{T}$

- *if* $a \in A_2 \setminus A_1$ *and* $(q_2, a, \varphi_2, \delta_2, q_2') \in \mathbb{T}_2$ *then for every* $q_1 \in Q_1$, $((q_1, q_2), a, \varphi_2, \delta_2, (q_1, q_2')) \in \mathbb{T}$

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

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 $a_1 : i \to j$ and $a_2 : i \to k$. The connector $Sync$ is then defined for every diagram δ where $\delta(i)$ is the empty transition system over the signature $(\Sigma_\emptyset, A_i)$ where $\Sigma_\emptyset$ is the empty **FOL**-signature, $\delta(j) = (Q_j, \mathbb{T}_j)$ over the signature (Σ_j, A_j) and $\delta(k) = (Q_k, \mathbb{T}_k)$ over the signature (Σ_k, A_k), and yields $Sync(\delta) = \delta(j) \otimes \delta(k)$ over the signature (Σ, A) with the co-cone $p : Sig \circ \delta \to (\Sigma, A)$ 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 $\mathcal{S}_1 \otimes \mathcal{S}_2$ have generally true emergent properties. The reason is the set $Mod(Th(\mathcal{S}_1^\bullet \cup \mathcal{S}_2^\bullet))$ of Kripke frames may be greater than $Real(\mathcal{S}_1 \otimes \mathcal{S}_2)$. Indeed, Kripke frames in $Real(\mathcal{S}_1 \otimes \mathcal{S}_2)$ have to preserve the shape of the transition system $\mathcal{S}_1 \otimes \mathcal{S}_2$ unlike Kripke frames in $Mod(Th(\mathcal{S}_1^\bullet \cup \mathcal{S}_2^\bullet))$. Hence, properties in $(\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet$ may be more numerous than in $Th(\mathcal{S}_1^\bullet \cup \mathcal{S}_2^\bullet)$. 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 $\mathcal{S}_1$ and $\mathcal{S}_2$ are preserved by $\mathcal{S}_1 \otimes \mathcal{S}_2$. Let us suppose a $\mathcal{S}_1 \otimes \mathcal{S}_2$-model $(\mathcal{W}, R)$, and let us define a $\mathcal{L}_i$-model $(\mathcal{W}_i, R_i)$ for $i = 1, 2$ as follow:

- for every $q \in Q_i$, $\mathcal{W}_i^q = Mod(\Sigma_i \hookrightarrow \Sigma)(\mathcal{W}^{(q,q')})$ for any $q' \in Q_j$ with $j \neq i \in \{1, 2\}$
- $\forall a \in A_i$, $R_{i_a} = \{(q, q') | \exists \varphi \in Sen(\Sigma_i), \exists \delta \in \mathcal{SE}(\mathcal{L}),\ (q, a, \varphi, \delta, q') \in \mathbb{T}_i\}$

Let us note Γ_i for $i = 1, 2$, the set of all these $\mathcal{L}_i$-models.

Theorem. 3 *Each* $(\mathcal{W}_i, R_i) \in \Gamma_i$ *is a* $\mathcal{S}_i$*-model.*

Proof. *The first condition of Definition 22 is obvious. To prove the second condition, let us suppose a transition* $(q, a, \varphi, q') \in \mathbb{T}_i$*. By construction, there exists a transition* $((q, q_j), a, \varphi', \delta', (q', q_j')) \in \mathbb{T}$ *such that either* $\varphi' = \varphi$ *and* $\delta' = \delta$*, or* $\varphi' = \varphi \wedge \varphi''$ *and* $\delta' = \delta \cup \delta''$*. In both cases, by hypothesis, we have that* $(\mathcal{W}^{(q,q_j)} \models \varphi'$*. Therefore, by the satisfaction condition for* **FOL** $\mathcal{W}_i^q \models \varphi$*. Moreover, by the condition on side-effects in Definition 23, we have that* $\mathcal{W}_i^q \rightsquigarrow_\delta \mathcal{W}_i^{q'}$

Proposition. 2 $\forall \iota : V \to W$,
$(\forall (\mathcal{W}_i, R_i) \in \Gamma_i,\ \forall q \in Q_i, (\mathcal{W}_i, R_i) \models_\iota^q \varphi)$
$\implies (\forall q_j \in Q_j,\ (\mathcal{W}, R) \models_\iota^{(q,q_j)} \varphi)$

Proof. *By induction on the structure of* φ*.*

Basic case. φ *is of the form* $p(t_1, \ldots, t_n)$*. Let* $q_j \in Q_j$*. By definition, there exists* $(\mathcal{W}_i, R_i) \in \Gamma_i$ *such that* $\mathcal{W}_i^q = Mod(\Sigma_i \hookrightarrow \Sigma)(\mathcal{W}^{(q,q_j)})$*. By hypothesis, we have* $\mathcal{W}_i^q \models_\iota p(t_1, \ldots, t_n)$*, and then* $\mathcal{W}^{(q,q_j)} \models_\iota p(t_1, \ldots, t_n)$*.*

General case. *Let us handle the case where* φ *is* $\Box_a \varphi'$*. Let us suppose that* $(\mathcal{W}, R) \models_\iota^{(q,q_j)} \varphi$*. Then, let us consider* (q', q_j) *such that* $(q, q_j)\ R_a\ (q', q_j')$*. By the hypothesis, we have for every* $(\mathcal{W}_i, R_i) \in \Gamma_i$ *that* $(\mathcal{W}_i, R_i) \models_\iota^q \varphi$*. By construction, we also have* $q\ R_{i_a}\ q'$ *for every* $(\mathcal{W}_i, R_i) \in \Gamma_i$*. Therefore, for every* $(\mathcal{W}_i, R_i) \in \Gamma_i$*,* $(\mathcal{W}_i, R_i) \models_\iota^q \varphi'$*, and then by the induction hypothesis, we have* $(\mathcal{W}, R) \models_\iota^{(q',q_j')} \varphi'$*, whence we can conclude* $(\mathcal{W}, R) \models_\iota^{(q',q_j')} \varphi$*.*

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

Theorem. 4 $\mathcal{S}_i^\bullet \subseteq (\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet$

Proof. *Let* $\varphi \in \mathcal{S}_i^\bullet$*, and let* $(\mathcal{W}, R) \in Real(\mathcal{S}_1 \otimes \mathcal{S}_2)$*. Let* $\iota : V \to W$ *be an interpretation. By Theorem 3, for every model* $(\mathcal{W}_i, R_i) \in \Gamma_i$*, we have* $(\mathcal{W}_i, R_i) \models \varphi$*, and then for every* $q \in Q_i$ *we also have* $(\mathcal{W}_i, R_i) \models_\iota^q \varphi$*. Therefore, by Proposition 2, we have for every* $q_j \in Q_j$ *that* $(\mathcal{W}, R) \models_\iota^{(q,q_j)} \varphi$*, and then* $(\mathcal{W}, R) \models \varphi$*.*

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 $(\mathcal{W}, R)$ of transition system $\mathcal{S}$, for each $q \in Q$, $\mathcal{W}^q$ is now an algebra. Therefore, the logic for transition systems is the modal first-order logic defined as in Section 2 except that now Σ-atoms are restricted to Σ-equations.

Given two transition systems $\mathcal{S}_1$ and $\mathcal{S}_2$ over the signatures $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, and satisfying the above restriction, for $i \neq j \in \{1, 2\}$, and for every $(\mathcal{W}_j, R_j) \in Mod(\mathcal{S}_j)$ we define the mapping $\mathcal{F}_{(\mathcal{W}_j, R_j)} : Mod(\mathcal{S}_i) \to Mod(\mathcal{L})$ where $\mathcal{L}$ is the signature over which the transition system $\mathcal{S}_1 \otimes \mathcal{S}_2$ is built as follow: if we note for a Σ-algebra $\mathcal{A}$, $th(\mathcal{A}) = \{\varphi | \varphi : \textbf{CEL}\text{-formula}, \mathcal{A} \models \varphi\}$, then to every

$(\mathcal{W}_i, R_i), \mathcal{F}_{(\mathcal{W}_j,R_j)}((\mathcal{W}_i, R_i)) = (\mathcal{W}, R)$ such that $(\mathcal{W}, R)$ is the $\mathcal{L}$-model defined by [9]

- $\forall q \in Q_i,\ \forall q' \in Q_j,\ W^{(q,q')} = T_{\Gamma_i/\Gamma}(\mathcal{W}_i^q) \times T_{\Gamma_j/\Gamma}(\mathcal{W}_i^q)$
- $R_a = \{((q_1, q_1'), (q_2, q_2'))|\exists\varphi \in Sen(\Sigma), \exists\delta \in \mathcal{SE}(\mathcal{L}),\ ((q_1, q_1'), a, \varphi, (q_2, q_2')) \in \mathbb{T}\}$

where $\Gamma_i = th(\mathcal{W}_i^q)$, $\Gamma_j = th(\mathcal{W}_j^{q'})$, and $\Gamma = th(\mathcal{W}_i^q) \cup th(\mathcal{W}_j^{q'})$.

Theorem. 5 *For every $(\mathcal{W}_j, R_j) \in Mod(\mathcal{S}_j)$ and every $(\mathcal{W}_i, R_i) \in Mod(\mathcal{S}_i)$, $\mathcal{F}_{(\mathcal{W}_j,R_j)}((\mathcal{W}_i, R_i))$ is a $\mathcal{S}_1 \otimes \mathcal{S}_2$-model.*

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

1. *either of the form $\varphi' \wedge \varphi''$ with $\varphi' \in Sen(\Sigma_i)$ and $\varphi'' \in Sen(\Sigma_j)$ and $\delta' \cup \delta''$ with $\delta' \in \mathcal{SE}(\mathcal{L}_i)$ and $\delta'' \in \mathcal{SE}(\mathcal{L}_j)$,*
2. *or $\varphi \in Sen(\Sigma_i) \cup Sen(\Sigma_j)$ and $\delta \in \mathcal{SE}(\mathcal{L}_i) \cup \mathcal{SE}(\mathcal{L}_j)$.*

This then leads to the two following cases:

1. *Suppose that φ is of the form $\varphi' \wedge \varphi''$ and then $\delta = \delta' \cup \delta''$. This means by construction, that $(q_1, a, \varphi', \delta', q_1') \in \mathbb{T}_i$ and $(q_2, a, \varphi'', \delta'', q_2') \in \mathbb{T}_j$. By hypothesis, we have $\mathcal{W}_i^{q_1} \models \varphi'$ and $\mathcal{W}_j^{q_1'} \models \varphi''$. Therefore, we have that $T_{\Gamma_i/\Gamma}(\mathcal{W}_i^{q_1}) \models \varphi' \wedge \varphi''$ and $T_{\Gamma_j/\Gamma}(\mathcal{W}_j^{q_2}) \models \varphi' \wedge \varphi''$, and then so is $T_{\Gamma_i/\Gamma}(\mathcal{W}_i^{q_1}) \times T_{\Gamma_j/\Gamma}(\mathcal{W}_j^{q_2})$ (recall that conditional equations are preserved along the cartesian product of algebras). Moreover, by hypothesis, we also have that $\mathcal{W}_i^{q_1} \rightsquigarrow_{\delta'} \mathcal{W}_i^{q_2}$ and $\mathcal{W}_j^{q_2} \rightsquigarrow_{\delta''} \mathcal{W}_j^{q_2'}$. By definition, Γ_i (resp. Γ_j) contains the ground equational theory of $\mathcal{W}_i^{q_1}$ (resp. $\mathcal{W}_j^{q_2}$). If we note $\Gamma_i' = th(\mathcal{W}_i^{q_1'})$, $\Gamma_j' = th(\mathcal{W}_j^{q_2'})$ and $\Gamma' = th(\mathcal{W}_i^{q_1'}) \cup th(\mathcal{W}_j^{q_2'})$, then we have $T_{\Gamma_i/\Gamma}(\mathcal{W}_i^{q_1}) \rightsquigarrow_\delta T_{\Gamma_i'/\Gamma'}(\mathcal{W}_i^{q_2})$ and $T_{\Gamma_j/\Gamma}(\mathcal{W}_j^{q_2}) \rightsquigarrow_\delta T_{\Gamma_j'/\Gamma'}(\mathcal{W}_j^{q_2'})$.*
2. *The case where $\varphi \in Sen(\Sigma_i) \cup Sen(\Sigma_j)$ and $\delta' \in \mathcal{SE}(\mathcal{L}_i)$ and $\delta'' \in \mathcal{SE}(\mathcal{L}_j)$ is noticeably similar to the previous one.*

MFOL 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 $(\mathcal{W}_i, R_i) \in Mod(\mathcal{S}_i)$, every $(\mathcal{W}_j, R_j) \in Mod(\mathcal{S}_j)$, and every $q \in Q_i$ and every $q' \in Q_j$, the adjunct morphism $\mu_{\mathcal{W}_i^q} : \mathcal{W}_i^q \to Mod(\Sigma_i \hookrightarrow \Sigma)(T_{\Gamma_i \setminus \Gamma}(\mathcal{W}_i^q))$ is an isomorphism, then $(\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet \cap Sen(\mathcal{L}_i) \subseteq \mathcal{S}_i^\bullet$.*

Proof. *Let $\varphi \in (\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet \cap Sen(\mathcal{S}_i)$ and let $(\mathcal{W}_i, R_i) \in Mod(\mathcal{S}_i)$. By Theorem 5, for every $(\mathcal{W}_j, R_j) \in Mod(\mathcal{S}_j)$, we have that $\mathcal{F}_{(\mathcal{W}_j,R_j)}((\mathcal{W}_i, R_i)) \models \varphi$. As the adjunct morphism $\mu_{\mathcal{W}_i^q}$ is an isomorphism, for every $\iota : V \to W_i$ there exists $\iota' : V \to T_{\Gamma_i/\Gamma}(W_i) \times T_{\Gamma_j/\Gamma}(W_j)$ such that $\iota = p_i \circ \iota'$ where p_i is the i-th projection map $p_i : T_{\Gamma_i/\Gamma}(\mathcal{W}_i^q) \to T_{\Gamma_i/\Gamma}(\mathcal{W}_i^q) \otimes T_{\Gamma_j/\Gamma}(\mathcal{W}_j^{q'})$ for $q \in Q_i$ and $q' \in Q_j$. By hypothesis, for every $q \in Q_i$ and every $q' \in Q_j$, $\mathcal{F}_{(\mathcal{W}_j,R_j)}((\mathcal{W}_i, R_i)) \models_{\iota'}^{(q,q')} \varphi$. It is then easy to show by induction on the structure of φ that $(\mathcal{W}_i, R_i) \models_\iota^q \varphi$.*

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 the suppose for the below counter-example 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 $\mathcal{S}_1$ and $\mathcal{S}_2$ defined respectively over the two following signatures $\mathscr{L}_1$ and $\mathscr{L}_2$:*

[9] **Cartesian product and preservation results** Let Σ be a signature, I be a set and $(\mathcal{A}_i)_{i\in I}$ be a I-indexed family of Σ-algebras. Let us note $\prod_{i\in I} \mathcal{A}_i$ the Σ-algebra defined as follow:

- for every $s \in S$, its carrier of sort s is $\prod_{i\in I} (A_i)_s$,
- for every $f : s_1 \times \ldots \times s_n \to s \in F$, $f^{\prod_{i\in I} \mathcal{A}_i}$ is the mapping that to every $(a_1, \ldots, a_n) \in \prod_{i\in I} (A_i)_{s_1} \times \ldots \times \prod_{i\in I} (A_i)_{s_n}$, associates $(f^{\mathcal{A}_i}(a_1^i, \ldots, a_n^i)|i \in I)$ where given $a \in \prod_{i\in I} (A_i)_s$, a^i is the ith coordinate of a.

By construction, we can notice that:

$$\prod_{i\in I} \mathcal{A}_i \models_\iota \varphi \iff \forall i \in I,\ \mathcal{A}_i \models_{\iota^i} \varphi$$

where for every interpretation ι, ι^i is the interpretation defined by $x \mapsto a^i$ if $\iota(x) = a$. It is well-known that conditional equations are preserved by Cartesian product of algebras, that is, if for every $i \in I$, $\mathcal{A}_i \models \Gamma \Rightarrow \alpha$, then $\prod_{i\in I} \mathcal{A}_i \models \Gamma \Rightarrow \alpha$.

[10] A Σ-algebra is *reachable* if, and only if the unique Σ-morphism $\mu : T_F \to \mathcal{A}$ is surjective, that is all the values in A are denoted by the evaluation of a ground term.

$$\Sigma_1 = \left\{ \begin{array}{l} S = \{nat\}, \\ F = \left\{ \begin{array}{l} 0 :\rightarrow nat; \\ s : nat \rightarrow nat, \\ + : nat \times nat \rightarrow nat \end{array} \right\}, \\ P = \emptyset \end{array} \right.$$

$$\Sigma_2 = \left\{ \begin{array}{l} S = \{nat\}, \\ F = \{0 :\rightarrow nat; s, p : nat \rightarrow nat\}, \\ P = \emptyset \end{array} \right.$$

$A_1 = A_2 = \{a\}$

Let us define $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows:

- $\mathcal{S}_1 = (\{q_1, q_2\}, \{q_1 \stackrel{a,\varphi_1,\delta_1}{\longrightarrow} q_2\})$ *where* $\varphi_1 = (s(x) = s(y) \Rightarrow x = y) \wedge x + 0 = x \wedge x + s(y) = s(x + y)$ *and* $\delta_1 = \emptyset$. $\mathcal{S}_2 = (\{q'_1, q'_2\}, \{q'_1 \stackrel{a,\varphi_2,\delta_2}{\longrightarrow} q'_2\})$ *where* $\varphi_2 = (s(x) = s(y) \Rightarrow x = y) \wedge s(p(x)) = x \wedge p(s(x)) = x$ *and* $\delta_2 = \emptyset$.

By definition of $\mathcal{S}_1$ (resp. $\mathcal{S}_2$), the unique $\mathcal{S}_1$-model (resp. $\mathcal{S}_2$-model) is $(\mathcal{W}, R)$ where $\mathcal{W}_1^{q_1} = \mathcal{W}_1^{q_2} = \mathbb{N}$ (resp. $\mathcal{W}_2^{q'_1} = \mathcal{W}_2^{q'_2} = \mathbb{Z}$). On the contrary, by construction, in $\mathcal{S}_1 \otimes \mathcal{S}_2$, we have the transition $(q_1, q'_1) \stackrel{a,\varphi',\delta'}{\longrightarrow} (q_2, q'_2)$ where $\varphi' = \varphi_1 \wedge \varphi_2$ and $\delta' = \emptyset$, and then all the $\mathcal{S}_1 \otimes \mathcal{S}_2$-model satisfy $\mathcal{W}^{(q_1,q'_1)} = \mathcal{W}^{(q_2,q'_2)} = \mathbb{Z}$. Consequently, the modal formula $\varphi' \Rightarrow \Box_a(\forall x.\exists y.x + y = 0)$ belongs to $(\mathcal{S}_1 \otimes \mathcal{S}_2)^\bullet$ but not in $\mathcal{S}_1^\bullet$. The reason is $F_{(\mathcal{W}_2,R_2)}(\mathcal{W}_1^{q_1}) = \mathbb{Z}$. Therefore, the adjunct functor $\mu_{\mathcal{W}_1^{q_1}}$ is injective but not surjective, and then 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

[1] M. Aiguier. Étoile-specifications: An object-oriented algebraic formalism with refinement. *Journal of Logic and Computation*, 14(2):145–178, 2004.

[2] M. Aiguier, C. Gaston, and P. Le Gall. Feature logics and refinement. In *Proceedings of the 9th Asian Pacific Software Engeenering Conference (APSEC)*, pages 385–395. IEEE Computer Society Press, 2002.

[3] M. Aiguier, P. Le Gall, and M. Mabrouki. Emergent properties in reactive systems. *Proceedings of the 15th Asia-Pacific Software Engineering Conference, APSEC08*, pages 273–280, 2008.

[4] M. Aiguier, P. Le Gall, and M. Mabrouki. A formal definition of complex software. In *ICSEA*, pages 415–420. IEEE Computer Society, 2008.

[5] R. Allen. *A Formal Approach to Software Architecture*. PhD thesis, Carnegie Mellon, School of Computer Science, Junuary 1997. Issued as CMU Technical Report CMU-CS-97-144.

[6] R. Allen and D. Garlan. A formal basis for architectural connectors. *ACM TOSEM*, 6(3):213–249, 1997.

[7] L. Blass, P. Clements, and R. Kasman. *Software Architecture in Practice*. Addison Wesley, 1998.

[8] T. Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286:197–245, 2002.

[9] S. Coudert and P. Le Gall. A reuse-oriented framework for hierarchical specifications. In *AMAST '00: Proc. of the 8th Int. Conference on Algebraic Methodology and Software Technology*, pages 438–453, London, UK, 2000. Springer-Verlag.

[10] R.-I. Damper. Emergence and levels of abstraction. *International Journal of Systems Science*, 31(7):811–818, 2000. Editorial for the Special Issue on 'Emergent Properties of Complex Systems'.

[11] R. Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10(4):383–402, 2002.

[12] R. Diaconescu. Jewels of institution-independent model theory. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to J.-A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[13] A.-C. Ehresmann and J.-P. Vanbremeersch. *Memory Evolutive Systems: Hierarchy, Emergence, Cognition*. Elsevier Science, 2007.

[14] H. Ehrig, M. Balmadus, and F. Orejas. New concepts for amalgation and extension in the framework of specification logics. In *Algebraic Methodology and Software Technology (AMAST)*, Lecture Notes in Computer Science. Springer, 1991.

[15] J.-L. Fiadeiro. *Categories for Software Engineering*. Springer-Verlag, 2004.

[16] J.-L. Fiadeiro, A. Lopes, and M. Wermelinger. A mathematical semantics for architectural connectors. In R.-C. Backhouse and J. Gibbons, editors, *Generic Programming*, volume 2793 of *Lecture Notes in Computer Science*, pages 178–221. Springer-Verlag, 2003.

[17] D. Garlan, R.-T. Monroe, and D. Wile. Acme: An architecture description interchange language. In *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research (CASCON)*, pages 169–183. IBM Press, 1997.

[18] C. Gaston, M. Aiguier, and P. Le Gall. *Language Constructs for Decsribing Features*, chapter Algebraic treatment of feature-oriented systems, pages 105–125. Springer-Verlag, 2000.

[19] J. Goguen. *Advances in Cybernetics and Systems Research*, chapter Categorical Foundations for General Systems Theory, pages 121–130. Transcripta Books, 1973.

[20] J. Goguen and R.-M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[21] Y. Gurevich. Evolving algebras 1993: Lipari guide. In *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.

[22] J.-V. Guttag and J.-J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, pages 27–52, 1978.

[23] T. Mossakowski. Institutional 2-cells and grothendieck institutions. In *Essays Dedicated to Joseph A. Goguen*, volume 4060 of *Lecture Notes in Computer Science*, pages 124–149. Springer, 2006.

[24] F. Orejas. *Algebraic Foundations of Systems Specification*, chapter Structuring and Modularity, pages 159–201. IFIP State-of-the-Art Reports. Springer, 1999.

[25] D. Perry and A. Wolf. Foundations for the study of software architectures. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[26] M. Plath and M. Ryan. Feature integration using a feature construct. *Science of Computer Programming*, 41(1):53–84, 2001.

[27] A. Sernadas, C. Sernadas, and C. Caleiro. Denotational semantics of object specification. *Acta Informatica*, 35(9):729–773, 1998.

[28] A. Tarlecki. Moving between logical systems. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer Verlag, 1996.

[29] A. Tarlecki. *Algebraic Foundations of Systems Specification*, chapter Institutions: An abstract Framework for Formal Specifications, pages 105–131. IFIP State-of-the-Art Reports. Springer, 1999.