Some Models of Heterogeneous and Distributed Specifications based on Universal Constructions

Edward Hermann Haeusler   Alfio Martini   Uwe Wolter

Abstract

In the last few years, it has been shown that much of the modern architectural software engineering can be suitably modelled with standard tools of category theory. In [9, 20], it is presented how universal constructions (limits and colimits) can, under certain mathematical conditions, help in providing modular tools to interconnect heterogeneous distributed systems described as specifications within arbitrary logical systems. In this paper, we revisit the theme of software interoperability and investigate how to extend these ideas to situations where the mathematical conditions in [9, 20] are not (fully) satisfied.

Keywords: category theory, universal constructions, heterogeneous systems, heterogeneous specifications, institution comorphisms.

MSC classification: 03B70,03C95,18A15,18C50

1 Introduction

The specification and design of a complex system can usually take advantage of a wide range of (rigorous) descriptions of diverse aspects or views of the system. This includes, for instance, both executable and non-executable formal specifications, models of concurrency and interaction, and specifications of security, dependability and real-time requirements, to name a few. This means that we must have clear mathematical foundations with which we can integrate and interoperate in a coherent and mathematically rigorous way these possibly diverse (architectural) descriptions.

Software Architecture (SA) is usually taken as the discipline of building systems from components in a uniform way. This uniformity is strongly related to the use of a small set of basic concepts and operations. Amongst the basic concepts we can find communication, hierarchy, inheritance, subsumption, instantiation, and superposition. Concerning operations, one can find concurrent composition, sequential composition, synchronous/asynchronous message passing, and parallel composition. Research developed in the 90's concluded that almost all composition mechanisms above cited can be obtained by the basic concepts of *components*, *glue* and *role*. This was demonstrated by the works of Garlan, Shaw and Adler on ADLs (Architecture Description Language) [1, 2, 23] with different flavors. In fact their works take different views of SA, but the ways the components are composed in order to define a new one is essentially the same. On the other hand, people with a strong mathematical/logical background have proposed a precise foundation for SA. Amongst the main influential works we must cite the early papers of Goguen and Burstall [5, 6, 12] on algebraic structuring of theories and the later works of Fiadeiro, Wermelinger, and Lopes [11, 14] on the use of category theory and the very concept of colimit in order to make precise the concepts of gluing and role in a component. Thus, a SA is taken as similar to a (structurally composed) algebraic theory as possible.

Some of the works cited here took their motivation from industrial projects. In this context, the project ARTS (SIEMENS/PUC-Rio), as described in [10, 22], delivered a platform for the support of the whole life-cycle of a systems development, based on object orientation (OO) for real-time applications in the domain of telecommunications. The platform integrated a specification layer, based on real-time modal logic well-suited to reason on timed-transition systems. A system designed in the context of ARTS consisted of structurally composed real-time transition systems. The composition followed the rules of OO-Design in the form of patterns. Those patterns are based on categorical concepts. Indeed, synchrony, parallel and concurrent composition are taken as colimits, following the foundation mentioned above. So to say, a system design in ARTS is given by a corresponding configuration diagram, i.e., the diagram constituted by the system components and the imposed relations between them. The system itself is the colimit of this diagram of timed-transition system components from the extensional point-of-view, and, a presentation of a (structured) RETOOL (Real Time Object Oriented Language) theory, from an intentional point-of-view. Of course the timed-transition system was used to formally derive C++ code, and, the RETOOL theory was used to formally prove the (real-time) requirements of the system.

However, this colimit must be preserved by the translations among the formalisms. Besides that, the translations among the different formal objects in ARTS are categorical functors, indeed, whilst each object inhabited its respective category. However, in praxis of refinement from the specification into the C++ code, the translation between the different formalism is componentwise. In fact the colimit picture of a SA is broken for the system components belonging to different categories. Thus the configuration diagram splits into a heterogeneous collection of small diagrams distributed over different categories. A worth task in the foundation of this approach is the demonstration of the soundness of this picture, here called interoperability via architecture.

Moreover, the formalisms involved in ARTS are either logical or programming languages abstractions. In either case they are institutions (Goguen and Burstall [12, 24, 26]), as already proven by the algebraic specification community.

RETOOL itself is a normal modal logic with a bring-it-about modality, C++ is a conventional programming language, and real-time transition systems are models for RETOOL theories. Thus, we are in the picture of a distributed, heterogeneous system design. The most fundamental questions in this case are, firstly, how the distributed collection of diagrams can nevertheless seen as a single diagram describing the overall system design. And, secondly, what are the possible colimits of such a heterogeneous configuration diagram or, in other words, what does it mean that such diagram is well-formed, i.e., constitutes a non-contradictory system design. This article is devoted to provide a positive pragmatic answer to these questions.

The paper is organized as follows: in section 2 we introduce institutions as our basic formal model for a logical formalism, where specifications (systems) can be described and related at a homogeneous level. Heterogeneous theories and their relations are captured with institution comorphisms and special Grothendieck constructions. In section 3, we show, under reasonable assumptions, how to build colimits of heterogeneous systems, modelled as diagrams in a suitable category. Section 4 introduces proper distributed specifications with ideas generalizing the notions of coherent families and morphisms [6, 7]. In section 5 we provide concluding remarks and points for further work.

Remark 1 (Notation) *With respect to notation, the collection of objects of a category* C *will be denoted* $|\mathsf{C}|$. *Given objects* $a, b \in |\mathsf{C}|$, *the collection (usually a set) of arrows from* a *to* b *is denote* $\mathsf{C}(a, b)$. *The category* Set *is the category of sets and total functions, while* Cat *is the category of all categories and functors.*

We represent composition of maps (functors) in diagrammatic order. For instance if $F : \mathsf{A} \to \mathsf{B}$ *and* $G : \mathsf{B} \to \mathsf{C}$ *are functors and* $a \in |\mathsf{A}|$, *then* $F; G : \mathsf{A} \to \mathsf{C}$ *and* $(F; G)(a) \stackrel{def}{=} G(F(a))$ *is an object of* C, *i.e.,* $G(F(a)) \in |\mathsf{C}|$.

Also, different institutions, frames, logics, are identified with primed superscripts (e.g., $\mathcal{I}, \mathcal{I}', \mathcal{I}''$*,etc.), while different objects within an institution (frame, logic), as signatures, models, etc., are denoted with numbered subscripts (e.g.,* $\Sigma_1, \Sigma_2, \mathsf{M}_1, \mathsf{M}_2$*,etc.).*

Moreover, if $\alpha : F \Rightarrow G : \mathsf{A} \to \mathsf{B}$ *and* $\beta : G \Rightarrow H : \mathsf{A} \to \mathsf{B}$ *are natural transformations, then the vertical composition of* α *and* β *is denoted* $\alpha; \beta : F \Rightarrow H : \mathsf{A} \to \mathsf{B}$ *such that for each* $a \in |\mathsf{A}|, (\alpha; \beta)_a \stackrel{def}{=} \alpha_a; \beta_a$. *Also, if* $F : \mathsf{A} \to \mathsf{B}$, $G, G' : \mathsf{B} \to \mathsf{C}$, $H : \mathsf{C} \to \mathsf{D}$ *are functors and* $\alpha : G \Rightarrow G' : \mathsf{B} \to \mathsf{C}$ *is a natural transformation, then the horizontal compositions of* F *with* α, *and* α *with* H *are represented as* $(F \centerdot \alpha) : F; G \Rightarrow F; G' : \mathsf{A} \to \mathsf{C}$ *and* $(\alpha \centerdot H) : G; H \Rightarrow G'; H : \mathsf{B} \to \mathsf{D}$ *such that for each* $c \in |\mathsf{C}|, a \in |\mathsf{A}|, (F \centerdot \alpha)_c \stackrel{def}{=} \alpha_{F(c)}$ *whereas* $(\alpha \centerdot H)_a \stackrel{def}{=} H(\alpha_a)$.

$$
\begin{tikzcd}
\mathsf{A} \arrow[r, "F"] & \mathsf{B} \arrow[r, bend left, "G"] \arrow[r, bend right, "G'"'] \arrow[r, phantom, "\Downarrow \alpha"] & \mathsf{C} \arrow[r, "H"] & \mathsf{D}
\end{tikzcd}
$$

□

2 Basic Definitions

The concept of an *institution* introduced by Goguen and Burstall [12] formally captures the notion of logical systems and allowed them to reformulate and to generalize the work they had done in the 70's on structuring (equational) specification [5, 6] independently of the underlying logic. A similiar proposal of an abstract concept of a logic had been given already by Barwise [3].

Definition 1 (Institution) *An Institution $\mathcal{I} = (\mathsf{Sign}, Sen, Mod, \models)$ consists of the following data and operations:*

- A CATEGORY OF ABSTRACT SIGNATURES Sign*;*
- A SYNTAX FUNCTOR $Sen : \mathsf{Sign} \to \mathsf{Set}$*;*
- A MODEL FUNCTOR $Mod : \mathsf{Sign}^{op} \to \mathsf{Cat}$*;*
- AN INDEXED FAMILY OF SATISFACTION RELATIONS $\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$*,* $\Sigma \in |\mathsf{Sign}|$ *such that the following institution condition*

$$Mod(\phi)(M_2) \models_{\Sigma_1} \varphi_1 \iff M_2 \models_{\Sigma_2} Sen(\phi)(\varphi_1)$$

holds for each $\phi : \Sigma_1 \to \Sigma_2$ *in* Sign*,* $M_2 \in |\mathsf{Mod}(\Sigma_2)|$*, and* $\varphi_1 \in Sen(\Sigma_1)$*.*

$$\begin{tikzcd}
\Sigma_1 \arrow[d, "\phi"'] & Mod(\Sigma_1) \arrow[r, leftrightarrow, "\models_{\Sigma_1}"] & Sen(\Sigma_1) \arrow[d, "Sen(\phi)"] \\
\Sigma_2 & Mod(\Sigma_2) \arrow[u, "Mod(\phi)"] \arrow[r, leftrightarrow, "\models_{\Sigma_2}"'] & Sen(\Sigma_2)
\end{tikzcd}$$

□

To keep the exposition short and accessible for a broader audience we concentrate on two well-known examples of institutions – equational logic and many-sorted equational logic. For other and for more elaborated examples we refer to the literature [11, 12, 15, 16, 21, 25, 26].

Example 1 (Institution $\mathcal{EQ}$ of Equational Logic) *This is the classical unsorted universal algebra and goes back to* Birkhoff *[4]. Signatures $\Sigma = (OP, ar)$ are given by a set OP of operation symbols and an arity function $ar : OP \to \mathbb{N}^*$. Signature morphisms $\phi : \Sigma_1 \to \Sigma_2$ are given by a map $\phi : OP_1 \to OP_2$ such that $ar_2(\phi(op_1)) = ar_1(op_1)$ for every $op_1 \in OP_1$. Signatures and signature morphisms form the category of signatures* $\mathsf{Sign}_{\mathcal{EQ}}$*.*

For any signature Σ and any set X of variables we define inductively the set $T(\Sigma, X)$ of Σ-terms over X. Thus we can assign to any signature Σ the set $Sen_{\mathcal{EQ}}(\Sigma)$ of Σ-equations $(l = r)$ where $l, r \in T(\Sigma, X)$. Every signature morphism $\phi : \Sigma_1 \to \Sigma_2$ induces inductively a translation $\phi : T(\Sigma_1, X) \to T(\Sigma_2, X)$ of Σ_1-terms over X into Σ_2-terms over X and provides a functor $Sen_{\mathcal{EQ}}(\phi) : Sen_{\mathcal{EQ}}(\Sigma_1) \to Sen_{\mathcal{EQ}}(\Sigma_2)$ where $Sen_{\mathcal{EQ}}(\phi)(l = r) = (\phi(l) = \phi(r))$. The sets $Sen_{\mathcal{EQ}}(\Sigma)$ and the translation functors $Sen_{\mathcal{EQ}}(\phi)$ constitute the syntax functor $Sen_{\mathcal{EQ}} : \mathsf{Sign}_{\mathcal{EQ}} \to \mathsf{Set}$.

For any signature Σ we define the category $Mod_{\mathcal{EQ}}(\Sigma)$ of Σ-algebras as follows. The objects are Σ-algebras A, i.e. there is a non- empty carrier set A and for every $op \in OP$, $ar(op) = n$ there is an operation $A(op) : A^n \to A$. The morphisms are Σ-homomorphisms $h : A \to B$ translating the carriers compatible with the operations, i.e. $h(A(op)(a_1, \ldots, a_n)) = B(op)(h(a_1), \ldots, h(a_n))$, for every $op \in OP$ and $a_i \in A$.

Given a signature morphism $\phi : \Sigma_1 \to \Sigma_2$ any Σ_2-algebra A_2 defines a Σ_1-algebra $Mod_{\mathcal{EQ}}(A_2)$ with the same carrier A_2 and with $Mod_{\mathcal{EQ}}(A_2)(op_1) = A_2(\phi(op_1))$ for every $op_1 \in OP_1$. This construction also applies to Σ_2-homomorphisms thus we obtain a forgetful functor $Mod_{\mathcal{EQ}}(\phi) : Mod_{\mathcal{EQ}}(\Sigma_2) \to Mod_{\mathcal{EQ}}(\Sigma_1)$. The categories $Mod_{\mathcal{EQ}}(\Sigma)$ and the forgetful functors $Mod_{\mathcal{EQ}}(\phi)$ constitute the model functor $Mod_{\mathcal{EQ}} : \mathsf{Sign}_{\mathcal{EQ}}^{op} \to \mathsf{Cat}$.

Given a signature Σ, a Σ-algebra A, and a set X of variables any variable assignment $\alpha : X \to A$ can be extended inductively to a term evaluation $\overline{\alpha} : T(\Sigma, X) \to A$. A Σ-equation $(l = r)$ is satisfied in A, $A \models_\Sigma (l = r)$ in symbols, iff $\overline{\alpha}(l) = \overline{\alpha}(r)$ for all assignments α.

Let be given a signature morphism $\phi : \Sigma_1 \to \Sigma_2$, a Σ_1-equation $(l = r)$, and a Σ_2-algebra A_2. The crucial technical result for proving the institution condition

$$Mod_{\mathcal{EQ}}(\phi)(A_2) \models_{\Sigma_1} (l = r) \quad \Leftrightarrow \quad A_2 \models_{\Sigma_2} (\phi(l) = \phi(r))$$

is that the assignments of X into $Mod_{\mathcal{EQ}}(\phi)(A_2)$ coincide with the assignements of X into A_2 and that we have $\overline{\alpha}(t) = \overline{\alpha}(\phi(t))$ for any assignment $\alpha : X \to A_2$ and for any $t \in T(\Sigma_1, X)$ (compare [26]). □

Example 2 (Institution $\mathcal{MEQ}$ of Many-Sorted Equational Logic) *Signatures are of the form $\Sigma = (S, OP, dom, cod)$ with S a set of sort symbols, OP a set of operation symbols, the domain function $dom : OP \to S^*$, and the codomain function $cod : OP \to S$. Signature morphisms $\phi : \Sigma_1 \to \Sigma_2$ translate sort and operation symbols compatible with the domain and codomain functions, i.e. $dom_2(\phi(op_1)) = \phi^*(dom_1(op_1))$ and $cod_2(\phi(op_1)) = \phi(cod_1(op_1))$ for every $op_1 \in OP_1$. Signatures and signature morphisms form the category of many-sorted signatures $\mathsf{Sign}_{\mathcal{MEQ}}$.*

For any signature Σ and any S-set $X = (X_s \mid s \in S)$ of variables we define inductively the S-set $T(\Sigma, X)$ of Σ-terms over X. Thus we can assign to any signature Σ the set $Sen_{\mathcal{MEQ}}(\Sigma)$ of Σ-equations $(X : l = r)$ where $l, r \in$

$T(\Sigma, X)(s)$, $s \in S$. Every signature morphism $\phi : \Sigma_1 \to \Sigma_2$ induces a translation of S_1-sets $X = (X(s_1) \mid s_1 \in S_1)$ of variables into S_2-sets $\phi(X)$ of variables where $\phi(X)(s_2) = \biguplus\{X(s_1) \mid \phi(s_1) = s_2\}$ for any $s_2 \in S_2$. This translation extends inductively to a family $\phi(s_1) : T(\Sigma_1, X)(s_1) \to T(\Sigma_2, \phi(X))(\phi(s_1))$, $s_1 \in S_1$ of translations of Σ_1-terms over X into Σ_2-terms over $\phi(X)$ and provides, finally, a functor $Sen_{\mathcal{MEQ}}(\phi) : Sen_{\mathcal{MEQ}}(\Sigma_1) \to Sen_{\mathcal{MEQ}}(\Sigma_2)$ where $Sen_{\mathcal{MEQ}}(\phi)(X : l = r) = (\phi(X) : \phi(s_1)(l) = \phi(s_1)(r))$. The sets $Sen_{\mathcal{MEQ}}(\Sigma)$ and the translation functors $Sen_{\mathcal{MEQ}}(\phi)$ constitute the syntax functor $Sen_{\mathcal{MEQ}} : \mathsf{Sign}_{\mathcal{MEQ}} \to \mathsf{Set}$.

For any signature Σ we define the category $Mod_{\mathcal{MEQ}}(\Sigma)$ of Σ-algebras as follows. The objects are Σ-algebras A, i.e. for every $s \in S$ there is a (possibly empty) carrier set $A(s)$ and for every $op : s_1 \dots s_n \to s$ in OP there is an operation $A(op) : A(s_1) \times \dots \times A(s_n) \to A(s)$. The morphisms are Σ-homomorphisms $h = (h(s) : A(s) \to B(s) \mid s \in S)$ translating the carriers compatible with the operations, i.e. $h(s)(A(op)(a_1, \dots, a_n)) = B(op)(h(s_1)(a_1), \dots, h(s_n)(a_n))$, for every $op : s_1 \dots s_n \to s$ in OP and $a_i \in A(s_i)$, $i = 1, \dots, n$.

A signature morphism $\phi : \Sigma_1 \to \Sigma_2$ defines an interpretation of the components of Σ_1 by suitable components of Σ_2. Analogously a Σ_2-algebra A_2 is given by an interpretation of the components of Σ_2 by suitable components of Set. Composing these two interpretations we obtain an interpretation of Σ_1 in Set, i.e. a Σ_1-algebra $Mod_{\mathcal{MEQ}}(A_2)$ with $Mod_{\mathcal{MEQ}}(A_2)(s_1) = A_2(\phi(s_1))$ for every $s_1 \in S_1$ and with $Mod_{\mathcal{MEQ}}(A_2)(op_1) = A_2(\phi(op_1))$ for every $op_1 \in OP_1$. This construction also applies to Σ_2-homomorphisms thus we obtain a forgetful functor $Mod_{\mathcal{MEQ}}(\phi) : Mod_{\mathcal{MEQ}}(\Sigma_2) \to Mod_{\mathcal{MEQ}}(\Sigma_1)$. The categories $Mod_{\mathcal{MEQ}}(\Sigma)$ and the forgetful functors $Mod_{\mathcal{MEQ}}(\phi)$ constitute the model functor $Mod_{\mathcal{MEQ}} : \mathsf{Sign}^{op}_{\mathcal{MEQ}} \to \mathsf{Cat}$.

Given a signature Σ, a Σ-algebra A, and an S-set $X = (X_s \mid s \in S)$ of variables any variable S-assignment $\alpha = (\alpha(s) : X(s) \to A(s) \mid s \in S)$ can be extended inductively to a term evaluation $\overline{\alpha} = (\overline{\alpha}(s) : T(\Sigma, X)(s) \to A(s) \mid s \in S)$. A Σ-equation $(X : l = r)$ is satisfied in A, $A \models_\Sigma (X : l = r)$ in symbols, iff $\overline{\alpha}(s)(l) = \overline{\alpha}(s)(r)$ for all assignments α. Note, that $(X : l = r)$ is vacuously satisfied if $X(s) \neq \emptyset$ and $A(s) = \emptyset$ for one $s \in S$.

Let be given a signature morphism $\phi : \Sigma_1 \to \Sigma_2$, a Σ_1-equation $(X : l = r)$, and a Σ_2-algebra A_2. The crucial technical result for proving the institution condition

$$Mod_{\mathcal{MEQ}}(\phi)(A_2) \models_{\Sigma_1} (X : l = r) \quad \Leftrightarrow \quad A_2 \models_{\Sigma_2} (\phi(X) : \phi(s_1)(l) = \phi(s_1)(r))$$

is that there is a one-to-one corresponcence between the assignments of X into $Mod_{\mathcal{MEQ}}(\phi)(A_2)$ and the assignments of $\phi(X)$ into A_2, respectively, and that there is a coincidence between the corresponding term evaluations. That is, for any S_1-assignment $\alpha : X \to Mod_{\mathcal{MEQ}}(\phi)(A_2)$ there is a S_2-assignment $\beta : \phi(X) \to A_2$ defined by $\beta(\phi(s_1))(x) = \alpha(s_1)(x)$ for all $s_1 \in S_1$ and $x \in X(s_1)$ such that $\overline{\alpha}(s_1)(t) = \overline{\beta}(\phi(s_1))(\phi(s_1)(t))$ for all $t \in T(\Sigma_1, X)(s_1)$ [26]. $\square$

Institutions are based on a pointwise assignment of signatures, sentences, and models. But, in design (programming) the relevant objects are not sentences (program lines), but, specifications (programs).

Definition 2 (Constructions for Specifications) *For a given institution* $\mathcal{I} = (\mathsf{Sign}, Sen, Mod, \models)$ *and any sets* $\Gamma, \Gamma_1, \Gamma_2 \subseteq Sen(\Sigma)$*, we fix the following:*

- *we write* $M \models_\Sigma \Gamma$ *if* $M \models_\Sigma \varphi$ *for all* $\varphi \in \Gamma$*,*
- *the semantical entailment* $\Gamma_1 \models_\Sigma \Gamma_2$ *holds if and only if* $M \models_\Sigma \Gamma_1$ *implies* $M \models_\Sigma \Gamma_2$*, for every model* $M \in |Mod(\Sigma)|$*.* □

Definition 3 (Category of Specifications) *For a given institution* $\mathcal{I}$*, the category* $\mathsf{Th(I)}$ *has as objects* **specifications***, i.e., pairs* $\langle \Sigma, \Gamma \rangle$*, where* $\Sigma \in |\mathsf{Sign}|$*,* $\Gamma \subseteq Sen(\Sigma)$ *and as arrows* **specification morphisms** $\phi : \langle \Sigma_1, \Gamma_1 \rangle \to \langle \Sigma_2, \Gamma_2 \rangle$*, where* $\phi : \Sigma_1 \to \Sigma_2$ *in* Sign *and* $\Gamma_2 \models_{\Sigma_2} Sen(\phi)(\Gamma_1)$*. There is also a forgetful functor* $sign : \mathsf{Th(I)} \to \mathsf{Sign}$ *such that* $sign(\langle \Sigma, \Gamma \rangle) \stackrel{def}{=} \Sigma$*.* □

Remark 2 *Note that we write* $\mathsf{Th(I)}$ *instead of* $Th(\mathcal{I})$ *to stress the fact that the former is a category.* □

The institution condition ensures that the model functor can be extended to specifications.

Proposition 1 (Generalized Model Functor) *Let be given an institution* $\mathcal{I} = (\mathsf{Sign}, Sen, Mod, \models)$*.*

- *For any specification* $\langle \Sigma, \Gamma \rangle$ *in* $\mathsf{Th(I)}$ *we denote by* $Mod_\models(\langle \Sigma, \Gamma \rangle)$ *the full subcategory of* $Mod(\Sigma)$ *given by all* Σ*-models* M *such that* $M \models_\Sigma \Gamma$*.*
- *For any specification morphism* $\phi : \langle \Sigma_1, \Gamma_1 \rangle \to \langle \Sigma_2, \Gamma_2 \rangle$ *we get a functor* $Mod_\models(\phi) : Mod_\models(\langle \Sigma_2, \Gamma_2 \rangle) \to Mod_\models(\langle \Sigma_1, \Gamma_1 \rangle)$ *as a restriction of the model functor* $Mod(\phi)$*.*

$$\begin{tikzcd}
\langle \Sigma_1, \Gamma_1 \rangle \arrow[d, "\phi"'] & Mod_\models(\langle \Sigma_1, \Gamma_1 \rangle) \arrow[r, hook] & Mod(\Sigma_1) \\
\langle \Sigma_2, \Gamma_2 \rangle & Mod_\models(\langle \Sigma_2, \Gamma_2 \rangle) \arrow[r, hook] \arrow[u, "Mod_\models(\phi)"] & Mod(\Sigma_2) \arrow[u, "Mod(\phi)"']
\end{tikzcd}$$

- *The assignments* $\langle \Sigma, \Gamma \rangle \mapsto Mod_\models(\langle \Sigma, \Gamma \rangle)$ *and* $\phi \mapsto Mod_\models(\phi)$ *define a* **generalized model functor** $Mod_\models : \mathsf{Th(I)}^{op} \to \mathsf{Cat}$ □

The following concept is due to Meseguer [16], but we adopt the up-to-date terminology from [13]. It formalizes how a target logic $\mathcal{I}'$ can code a source logic $\mathcal{I}$. It requires that the syntax of $\mathcal{I}'$ is rich enough to define the subclasses of models of $\mathcal{I}'$ which can be understood as models of $\mathcal{I}$.

Definition 4 (Institution Comorphisms)

An Institution Comorphisms $\mu = (\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ consists of:

- *A* FUNCTOR $\Phi : \mathsf{Sign} \to \mathsf{Sign}'$;
- *A* NATURAL TRANSFORMATION $\alpha : Sen \Longrightarrow \Phi; Sen' : \mathsf{Sign} \to \mathsf{Set}$;
- *A* NATURAL TRANSFORMATION $\beta : \Phi^{op}; Mod' \Longrightarrow Mod : \mathsf{Sign}^{op} \to \mathsf{Cat}$, *such that the following comorphism condition:*

$$\beta(\Sigma)(M') \models_\Sigma \varphi \Longleftrightarrow M' \models'_{\Phi(\Sigma)} \alpha(\Sigma)(\varphi)$$

holds for each $\Sigma \in |\mathsf{Sign}|, \varphi \in Sen(\Sigma)$ and $M' \in |Mod'(\Phi(\Sigma)|$.

$$\begin{tikzcd} \mathsf{Sign} \arrow[d, "\Phi"] \\ \mathsf{Sign}' \end{tikzcd} \quad \begin{tikzcd} \Sigma \arrow[d, mapsto] \\ \Phi(\Sigma) \end{tikzcd} \quad \begin{tikzcd} Mod(\Sigma) \arrow[rr, leftrightarrow, "\models_\Sigma"] & & Sen(\Sigma) \arrow[d, "\alpha(\Sigma)"] \\ Mod'(\Phi(\Sigma)) \arrow[u, "\beta(\Sigma)"] \arrow[rr, leftrightarrow, "\models'_{\Phi(\Sigma)}"'] & & Sen'(\Phi(\Sigma)) \end{tikzcd}$$

Example 3 (Institution Comorphism from $\mathcal{MEQ}$ into $\mathcal{EQ}$) *The process of "omitting sorts" provides a comorphism from $\mathcal{MEQ}$ into $\mathcal{EQ}$. We map a many-sorted signature $\Sigma = (S, OP, dom, cod)$ to the signature $\Phi(\Sigma) = (OP, ar)$ such that $ar(op) = n$ iff $dom(op)$ has length n. This defines a functor $\Phi : \mathsf{Sign}_{\mathcal{MEQ}} \to \mathsf{Sign}_{\mathcal{EQ}}$.*

For any many-sorted signature Σ and any S-set $X = (X_s \mid s \in S)$ of variables we obtain the "unsorted" set $\biguplus X = \biguplus \{X(s) \mid s \in S\}$ of variables and for every $s \in S$ we have an inclusion $T(\Sigma, X)(s) \subseteq T(\Phi(\Sigma), \biguplus X)$. This means that we have a natural transformation $\alpha : Sen_{\mathcal{MEQ}} \Longrightarrow \Phi; Sen_{\mathcal{EQ}}$ with $\alpha(\Sigma)(X : l = r) = (l = r)$. Note, that in most cases $\biguplus\{T(\Sigma, X)(s) \mid s \in S\}$ is a proper subset of $T(\Phi(\Sigma), \biguplus X)$ because many-sortedness means essentially, to put, in addition to the arity constraints, further constraints on the construction of terms.

Moreover, given a many-sorted signature Σ we can associate to any unsorted $\Phi(\Sigma)$-algebra A a many-sorted Σ-algebra $\beta(\Sigma)(A)$ with carriers $\beta(\Sigma)(A)(s) = A$ for all $s \in S$ and operations $\beta(\Sigma)(A)(op) = A(op)$ for all $op \in OP$. This gives a functor $\beta(\Sigma) : Mod_{\mathcal{EQ}}(\Phi(\Sigma)) \to Mod_{\mathcal{MEQ}}(\Sigma)$, and globally defines a natural transformation $\beta : \Phi^{op}; Mod_{\mathcal{EQ}} \Longrightarrow Mod_{\mathcal{MEQ}}$.

In accordance with the institution conditions the comorphism condition

$$\beta(\Sigma)(A) \models_\Sigma (X : l = r) \quad \Leftrightarrow \quad A \models_{\Phi(\Sigma)} (l = r)$$

is due to a one-to-one correspondence between the assignments of X into $\beta(\Sigma)(A)$ and the assignments of $\biguplus X$ into A, respectively, and due to the coincidence between the corresponding term evaluations. That is, for any S-assignment $\alpha : X \to \beta(\Sigma)(A)$ there is an assignment $\beta : \biguplus X \to A$ defined by $\beta(x) = \alpha(s)(x)$ for all $s \in S$, $x \in X(s)$ such that $\overline{\alpha}(s)(t) = \overline{\beta}(t)$ for all $t \in T(\Sigma, X)(s)$. □

Definition 5 (Category of Institutions) *Institutions and institution comorphisms define a category* $\mathsf{InstCom}$*. The identities $id_{\mathcal{I}} : \mathcal{I} \to \mathcal{I}$ are given by $id_{\mathcal{I}} \stackrel{def}{=} (id_{\mathsf{Sign}}, id_{Sen}, id_{Mod})$. And, given comorphisms $\mu = (\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ and $\mu' = (\Phi', \alpha', \beta') : \mathcal{I}' \to \mathcal{I}''$, the composition $\mu; \mu' : \mathcal{I} \to \mathcal{I}''$ is defined as follows:*

$$\mu; \mu' \stackrel{def}{=} (\Phi; \Phi', \alpha; (\Phi \centerdot \alpha'), (\Phi^{op} \centerdot \beta'); \beta) : \mathcal{I} \to \mathcal{I}''.$$

$$\begin{tikzcd} \mathsf{Sign} \arrow[r, bend left, "Sen"] \arrow[r, bend right, "\Phi; Sen'"'] \arrow[r, phantom, "\Downarrow \alpha"] & \mathsf{Set} \end{tikzcd} \qquad \begin{tikzcd} \mathsf{Sign}^{op} \arrow[r, "\Phi^{op}"] & \mathsf{Sign}'^{op} \arrow[r, bend left, "\Phi'^{op}; Mod''"] \arrow[r, bend right, "Mod'"'] \arrow[r, phantom, "\Downarrow \beta'"] & \mathsf{Cat} \end{tikzcd}$$

$$\begin{tikzcd} \mathsf{Sign} \arrow[r, "\Phi"] & \mathsf{Sign}' \arrow[r, bend left, "Sen'"] \arrow[r, bend right, "\Phi'; Sen''"'] \arrow[r, phantom, "\Downarrow \alpha'"] & \mathsf{Set} \end{tikzcd} \qquad \begin{tikzcd} \mathsf{Sign}^{op} \arrow[r, bend left, "\Phi^{op}; Mod'"] \arrow[r, bend right, "Mod"'] \arrow[r, phantom, "\Downarrow \beta"] & \mathsf{Cat} \end{tikzcd}$$

□

An important observation here is that many other notions of institution morphisms have been proposed over the last two decades. For a thorough exposition and comparison of these different concepts we refer to [13, 15].

The translation of signatures and sentences induces a corresponding transformation of specifications and the comorphism condition ensures that the model functor can be extend to these transformations of specifications.

Proposition 2 (Functorial Maps for Specifications and Models) *Let $\mu = (\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ be an institution comorphism.*

- *The natural transformation $\alpha : Sen \Longrightarrow \Phi; Sen' : \mathsf{Sign} \to \mathsf{Set}$ and the functor $\Phi : \mathsf{Sign} \to \mathsf{Sign}'$ define a functor $\Phi_\alpha : \mathsf{Th}(\mathsf{I}) \to \mathsf{Th}(\mathsf{I}')$ by the assignments $\Phi_\alpha(\langle \Sigma, \Gamma \rangle)) \stackrel{def}{=} \langle \Phi(\Sigma), \alpha(\Sigma)(\Gamma) \rangle$, such that $\Phi_\alpha; sign' = sign; \Phi$.*

$$\begin{tikzcd} \mathsf{Th}(\mathsf{I}) \arrow[r, "\Phi_\alpha"] \arrow[d, "sign"'] & \mathsf{Th}(\mathsf{I}') \arrow[d, "sign'"] \\ \mathsf{Sign} \arrow[r, "\Phi"'] & \mathsf{Sign}' \end{tikzcd}$$

- *The natural transformation* $\beta : \Phi^{op}; Mod' \Longrightarrow Mod : \mathsf{Sign}^{op} \to \mathsf{Cat}$ *can be extended to a natural transformation* $\beta_{\models} : \Phi_{\alpha}^{op}; Mod'_{\models'} \Longrightarrow Mod_{\models} : \mathsf{Th(I)}^{op} \to \mathsf{Cat}$*, where for any specification* $\langle \Sigma, \Gamma \rangle$ *the functor* $\beta_{\models}(\langle \Sigma, \Gamma \rangle) : Mod'_{\models'}(\Phi_{\alpha}(\langle \Sigma, \Gamma \rangle)) \to Mod_{\models}(\langle \Sigma, \Gamma \rangle)$ *is as a restriction of the functor* $\beta(\Sigma) : Mod'(\Phi(\Sigma)) \to Mod(\Sigma)$*.*

$$\begin{tikzcd}
\langle \Sigma, \Gamma \rangle \arrow[d, mapsto] & Mod_{\models}(\langle \Sigma, \Gamma \rangle) \arrow[r, hook] & Mod(\Sigma) \\
\Phi_{\alpha}(\langle \Sigma, \Gamma \rangle) & Mod'_{\models'}(\langle \Phi(\Sigma), \alpha(\Sigma)(\Gamma) \rangle) \arrow[u, "\beta_{\models}(\langle \Sigma, \Gamma \rangle)"] \arrow[r, hook] & Mod'(\Phi(\Sigma)) \arrow[u, "\beta(\Sigma)"']
\end{tikzcd}$$

□

Proposition 3 (Heterogeneous Specification Functor) *The assignments* $\mathcal{I} \mapsto \mathsf{Th(I)}$ *and* $\mu : \mathcal{I} \to \mathcal{I}' \mapsto \Phi_{\alpha} : \mathsf{Th(I)} \to \mathsf{Th(I')}$ *define an indexed category, i.e., a functor* $Th : \mathsf{InstCom} \to \mathsf{Cat}$*.* □

Up to now we have only relations between specifications within the same formalism. But, institution comorphisms allow to translate specifications from one formalism into other, thus we are able to relate specifications from different formalisms by means of the following

Definition 6 (Grothendieck construction) *Given an indexed category* $C : \mathsf{IND} \to \mathsf{Cat}$ *we define the category* $\mathsf{Flat(C)}$ *as follows:*

- objects: *are pairs* $\langle i, a \rangle$ *where* $i \in |\mathsf{IND}|$ *and* $a \in |\mathsf{C(i)}|$*.*
- arrows: *from* $\langle i, a \rangle$ *to* $\langle j, b \rangle$ *are pairs* (σ, f) *where* $\sigma : i \to j$ *is an arrow in* IND *and* $f : C(\sigma)(a) \to b$ *is an arrow in* $\mathsf{C(j)}$*.*
- composition:

$$\begin{tikzcd}
 & & c \\
 & b & C(\tau)(b) \arrow[u, "g"] \\
a & C(\sigma)(a) \arrow[u, "f"] & C(\tau)(C(\sigma)(a)) \arrow[u, "C(\tau)(f)"] \\
C(i) \arrow[r, "C(\sigma)"'] & C(j) \arrow[r, "C(\tau)"'] & C(k) \\
i \arrow[r, "\sigma"'] & j \arrow[r, "\tau"'] & k
\end{tikzcd}$$

Given arrows $\langle\sigma, f\rangle : \langle i, a\rangle \to \langle j, b\rangle$ *and* $\langle\tau, g\rangle : \langle j, b\rangle \to \langle k, c\rangle$ *in* $\mathsf{Flat(C)}$*, let* $\langle\sigma, f\rangle ; \langle\tau, g\rangle \stackrel{def}{=} \langle\sigma;\tau, C(\tau)(f);g\rangle$.

Moreover we obtain a projection *functor* $Proj_{\mathsf{Flat(C)}} : \mathsf{Flat(C)} \to \mathsf{IND}$ *with* $Proj_{\mathsf{Flat(C)}}(\langle i, a\rangle) = i$ *and* $Proj_{\mathsf{Flat(C)}}(\langle\sigma, f\rangle) = \sigma$ *for any* $\langle i, a\rangle$ *and any* $\langle\sigma, f\rangle$ *in* $\mathsf{Flat(C)}$. □

Definition 7 (Category of Heterogeneous Specifications) *We denote by* $\mathsf{Flat(Th)}$ *the category obtained from* $Th : \mathsf{InstCom} \to \mathsf{Cat}$ *via the Grothendieck construction above. The objects are called* ***heterogeneous specifications*** *and the arrows are called* ***(specification) bridges****. Note that we also have a projection* $Proj_{\mathsf{Flat(Th)}} : \mathsf{Flat(Th)} \to \mathsf{InstCom}$*, such that* $Proj_{\mathsf{Flat(Th)}}(\langle\mathcal{I}, \langle\Sigma, \Gamma\rangle\rangle) = \mathcal{I}$. □

Corollary 1 (Bridges) *A* ***(specification) bridge***

$$\langle\mu, \phi\rangle : \langle\mathcal{I}, \langle\Sigma, \Gamma\rangle\rangle \to \langle\mathcal{I}', \langle\Sigma', \Gamma'\rangle\rangle$$

is an arrow such that

- $\mu = (\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}'$ *is an institution comorphism and*
- $\phi : \Phi_\alpha(\langle\Sigma, \Gamma\rangle) \to \langle\Sigma', \Gamma'\rangle$ *is an arrow in* $\mathsf{Th(I')}$*, i.e., a signature morphism* $\phi : \Phi(\Sigma) \to \Sigma'$ *in* Sign' *satisfying* $\Gamma' \models'_{\Phi(\Sigma)} Sen(\phi)(\alpha(\Sigma)(\Gamma))$. □

3 Colimits of Heterogeneous Specifications

As indicated in the introduction, we are usually faced in practice with the following situation: the different aspects/views of a system are specified by means of different formalisms. That is, a system specification turns out to be a heterogeneous collection of specifications distributed over different formalisms, institutions. The question to answer is if this heterogeneous collection constitutes indeed a system specification or not. That is, what are the possible models of such a heterogeneous collection and under which conditions those models exist at all, i.e., what does it mean that such a collection is consistent.

Let us consider two specifications $\langle\Sigma', \Gamma'\rangle$ and $\langle\Sigma'', \Gamma''\rangle$ from two different institutions $\mathcal{I}'$ and $\mathcal{I}''$. Usually, both formalisms will overlap, i.e., there are system aspects that can be specified in each of the formalisms $\mathcal{I}'$ or $\mathcal{I}''$, respectively. Intuitively, the two specifications are consistent, if there are no contradictions concerning the common aspects covered by both specifications. It is hardly to expect that the overlapping of both formalisms are manifested by syntactical identities. According to the methodological imperative of Category Theory, we have to define instead the overlapping by pointing out a third institution $\mathcal{I}$, representing the intersection of both formalisms, together with two institution comorphisms $\mu' = (\Phi', \alpha', \beta') : \mathcal{I} \to \mathcal{I}'$and $\mu'' = (\Phi'', \alpha'', \beta'') : \mathcal{I} \to \mathcal{I}''$ indicating how the common part is coded within $\mathcal{I}'$ and $\mathcal{I}''$, respectively.

Now, the specifications $\langle \Sigma', \Gamma' \rangle$ and $\langle \Sigma'', \Gamma'' \rangle$ turn out to be consistent if they put the same requirements w.r.t. the common aspects, i.e., if there is a specification $\langle \Sigma, \Gamma \rangle$ in the shared institution $\mathcal{I}$ specifying the common aspects, and both specifications $\langle \Sigma', \Gamma' \rangle$ and $\langle \Sigma'', \Gamma'' \rangle$ appear as respective extensions of $\langle \Sigma, \Gamma \rangle$. Formally, this can be expressed by the existence of two specification bridges $\langle \mu', \phi' \rangle : \langle \mathcal{I}, \langle \Sigma, \Gamma \rangle \rangle \to \langle \mathcal{I}', \langle \Sigma', \Gamma' \rangle \rangle$ and $\langle \mu'', \phi'' \rangle : \langle \mathcal{I}, \langle \Sigma, \Gamma \rangle \rangle \to \langle \mathcal{I}', \langle \Sigma'', \Gamma'' \rangle \rangle$.

In terms of a stepwise, modular system development such a span of specification bridges arises quite naturally: First, we specify all $\mathcal{I}$-aspects of the system by $\langle \Sigma, \Gamma \rangle$, and than we specify independently the additional $\mathcal{I}'$-aspects and $\mathcal{I}''$-aspects by extending the specifications $\langle \Phi'(\Sigma), \alpha'(\Sigma)(\Gamma) \rangle$ and $\langle \Phi''(\Sigma), \alpha''(\Sigma)(\Gamma) \rangle$, respectively.

By a kind of "formal magic" (Grothendieck construction) our practical situation can be presented as a span (configuration diagram) in the category $\mathsf{Flat}(\mathsf{Th})$, and a natural idea would be to define the corresponding system as the colimit of this diagram in $\mathsf{Flat}(\mathsf{Th})$ or in a subcategory of $\mathsf{Flat}(\mathsf{Th})$ determined by the so-called *Grothendieck institution* of a diagram in $\mathsf{InstCom}$ involving only those institutions relevant for the actual application. [9, 20] provide conditions ensuring that those colimits exist in Grothendieck institution. The question addressed in this paper is: What can we do, if those colimits does not exist?

One idea could be to construct the pushout of the span

$$\begin{tikzcd}
 & \mathcal{I} \arrow[dl, "\mu'"'] \arrow[dr, "\mu''"] & \\
\mathcal{I}' & & \mathcal{I}''
\end{tikzcd}$$

of institutions. But, even if such a pushout should exist, it will be a hard and tedious task to construct actually such a pushout (as anybody knows who tried once to combine two fairly different formalisms into one single big one). The task will be even more unfeasible, if we bear in mind that there may be more institutions $\mathcal{I}'''$, ... involved, covering other relevant system aspects.

Actually, our span of theory bridges is a heterogeneous distributed diagram. We have two specification morphisms in different institutions virtually connected via a specification in a third institution

$$\begin{tikzcd}
 & & \langle \Sigma, \Gamma \rangle \arrow[dl, mapsto] \arrow[dr, mapsto] & & \\
\langle \Sigma', \Gamma' \rangle & \Phi_{\alpha'}(\langle \Sigma, \Gamma \rangle) \arrow[l, "\phi'"'] & & \Phi_{\alpha''}(\langle \Sigma, \Gamma \rangle) \arrow[r, "\phi''"] & \langle \Sigma'', \Gamma'' \rangle
\end{tikzcd}$$

where $\Phi_{\alpha'}(\langle\Sigma,\Gamma\rangle) \stackrel{def}{=} \langle\Phi'(\Sigma),\alpha'(\Sigma)(\Gamma)\rangle$ and $\Phi_{\alpha''}(\langle\Sigma,\Gamma\rangle) \stackrel{def}{=} \langle\Phi''(\Sigma),\alpha''(\Sigma)(\Gamma)\rangle$. In fact, there would be no problem if we would have a kind of "universal" institution $\mathcal{I}_u$ where all the institutions used for our system design can be embedded in a commutative way (a situation we have, for example, in CafeOBJ [8]). That is, instead a pushout we assume a commutative diagram

$$\begin{tikzcd}
 & \mathcal{I} \arrow[dl, "\mu'"'] \arrow[dr, "\mu''"] & \\
\mathcal{I}' \arrow[dr, "\mu'_u"'] & & \mathcal{I}'' \arrow[dl, "\mu''_u"] \\
 & \mathcal{I}_u &
\end{tikzcd}$$

of institution comorphisms. Then the heterogeneous, distributed diagram gives rise to a homogeneous diagram in $\mathcal{I}_u$

$$\begin{tikzcd}
 & \langle\Sigma_u,\Gamma_u\rangle \arrow[dl, "\Phi'_u(\phi')"'] \arrow[dr, "\Phi''_u(\phi'')"] & \\
\langle\Phi'_u(\Sigma'),\alpha'_u(\Sigma')(\Gamma')\rangle & & \langle\Phi''_u(\Sigma''),\alpha''_u(\Sigma'')(\Gamma'')\rangle
\end{tikzcd}$$

where $\langle\Sigma_u,\Gamma_u\rangle = \langle\Phi'_u(\Phi'(\Sigma)),\alpha'_u(\alpha'(\Sigma))(\Gamma)\rangle = \langle\Phi''_u(\Phi''(\Sigma)),\alpha''_u(\alpha''(\Sigma))(\Gamma)\rangle$. And, as long as $\mathcal{I}_u$ has the necessary features, as a finitely complete category of signatures and a continuous model functor, we are back to the traditional approach and we take as the system the colimit of this diagram .

But often, as in our project, we don't have such a "universal" institution, and thus we have to look for other ways to define the semantics of heterogeneous, distributed system specifications. What we can fairly assume is that at least on the level of signatures we do have colimits.

Remark 3 *In what follows, we consider the subcategory* $\mathsf{InstCom}_{coc}$ *of* $\mathsf{InstCom}$ *where*

- *for each institution* $\mathcal{I}$*, the corresponding category of signatures* Sign *is* FINITELY COCOMPLETE*, and*
- *for each comorphism* $\mu = (\Phi,\alpha,\beta) : \mathcal{I} \to \mathcal{I}'$*, the corresponding functor* $\Phi : \mathsf{Sign} \to \mathsf{Sign}'$ *is* FINITELY COCONTINOUS*.* □

An important result is established in the following

Lemma 1 *The category* $\mathsf{InstCom}_{coc}$ *of cocomplete categories and cocontinous functors is closed under colimit constructions.*

Proof. Consider the forgetful functor U from $\mathsf{InstCom}_{coc}$ to Cat, this is an inclusion of categories indeed. For any category $\mathcal{C}$ the Yoneda embedding $y_{\mathcal{C}} : \mathcal{C} \to \mathsf{Set}^{\mathcal{C}^{op}}$ is universal in $\mathsf{InstCom}_{coc}$ (see [18] section I.5). This implies that $\mathsf{Set}^{\mathcal{C}^{op}}$ is the free cocomplete category over $\mathcal{C}$. The mapping from $\mathcal{C}$ to $\mathsf{Set}^{\mathcal{C}^{op}}$ can be extended to a functor which is left adjoint to U. As Cat is cocomplete and left adjoints preserve colimits, it follows that $\mathsf{InstCom}_{coc}$ is cocomplete. □

Note, that the assumptions in Remark 3 are part of the conditions in [9, 20], mentioned above.

We present three propositions. The first one is a simple application of the fact that Cat is cocomplete. The second uses the fact that we are working with cocomplete categories of signatures for each institution involved, as well as cocontinous functors amongst them. The third proposition uses the cocompleteness of Set, the realm of the sentences of any theory in any institution, to establish the syntactic world of the heterogeneous pushout and its specification. The use of the completeness of Cat allows us to derive the model category of the pushout specification.

Remark 4 (Basic Diagram in of Bridges) *In the following, we assume the basic diagram of bridges:*

$$\langle \mathcal{I}', \langle \Sigma', \Gamma' \rangle \rangle \xleftarrow{\langle \mu', \phi' \rangle} \langle \mathcal{I}, \langle \Sigma, \Gamma \rangle \rangle \xrightarrow{\langle \mu'', \phi'' \rangle} \langle \mathcal{I}'', \langle \Sigma'', \Gamma'' \rangle \rangle$$

with $\mu' = (\Phi', \alpha', \beta') : \mathcal{I} \to \mathcal{I}'$ *and* $\mu'' = (\Phi'', \alpha'', \beta'') : \mathcal{I} \to \mathcal{I}''$ *institution comorphisms in* $\mathsf{InstCom}_{coc}$.

Next we present a series of propositions, as a methodological approach to define the semantics of our heterogeneous, distributed diagram in $\mathsf{InstCom}_{coc}$. Firstly, we construct a colimit category of signatures $\mathsf{Sign}' +_{\mathsf{Sign}} \mathsf{Sign}''$ (proposition 4) Then, based on the above diagram, we define a colimit signature $\Sigma' +_{\Sigma} \Sigma''$, which denotes the colimit of the signatures above.

Proposition 4 (Colimit of Signature Categories) *The pushout of* $\Phi' : \mathsf{Sign} \to \mathsf{Sign}'$ *and* $\Phi'' : \mathsf{Sign} \to \mathsf{Sign}''$ *in* Cat *is given by:*

$$\begin{tikzcd}
\mathsf{Sign} \arrow[r, "\Phi''"] \arrow[d, "\Phi'"'] & \mathsf{Sign}'' \arrow[d, dotted, "\Psi''"] \\
\mathsf{Sign}' \arrow[r, dotted, "\Psi'"'] & \mathsf{Sign}' +_{\mathsf{Sign}} \mathsf{Sign}''
\end{tikzcd}$$

□

Note that, by Lemma 1, $\mathsf{Sign}' +_{\mathsf{Sign}} \mathsf{Sign}''$ is also cocomplete. This enables us to introduce the next

Proposition 5 (Colimit of Signatures) *The colimit signature $\Sigma' +_\Sigma \Sigma''$ is defined by a pushout in $\mathsf{Sign}' +_{\mathsf{Sign}} \mathsf{Sign}''$ according to:*

$$\begin{tikzcd}
\Psi'(\Phi'(\Sigma)) = \Psi''(\Phi''(\Sigma)) \arrow[r, "\Psi''(\phi'')"] \arrow[d, "\Psi'(\phi')"'] & \Psi''(\Sigma'') \arrow[d, dotted, "\sigma''"] \\
\Psi'(\Sigma') \arrow[r, dotted, "\sigma'"'] & \Sigma' +_\Sigma \Sigma''
\end{tikzcd}$$

$\square$

Proposition 6 (Merging of Bridges) *The "merging" of the bridges $\langle \mu', \phi' \rangle : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$, $\langle \mu'', \phi'' \rangle : \langle \Sigma, \Gamma \rangle \to \langle \Sigma'', \Gamma'' \rangle$, is defined by the diagram below:*

$$\begin{tikzcd}
\langle \Sigma, \Gamma \rangle \arrow[r, "\langle \mu'', \phi'' \rangle"] \arrow[d, "\langle \mu', \phi' \rangle"'] & \langle \Sigma'', \Gamma'' \rangle \arrow[d, dotted] \\
\langle \Sigma', \Gamma' \rangle \arrow[r, dotted] & \langle \Sigma' +_\Sigma \Sigma'', \Gamma' +_\Gamma \Gamma'' \rangle
\end{tikzcd}$$

where the "merging" of sentences $\Gamma' +_\Gamma \Gamma''$ is given by the following pushout diagram in Set.

$$\begin{tikzcd}
Sen(\Sigma) \arrow[r, "\alpha''(\Sigma)"] \arrow[d, "\alpha'(\Sigma)"'] & Sen''(\Phi''(\Sigma)) \arrow[r, "Sen''(\phi'')"] & Sen''(\Sigma'') \arrow[dd, dotted, "\gamma''"] \\
Sen'(\Phi'(\Sigma)) \arrow[d, "Sen'(\phi')"'] & & \\
Sen'(\Sigma') \arrow[rr, dotted, "\gamma'"] & & Sen
\end{tikzcd}$$

Then for each $\Gamma' \subseteq Sen'(\Sigma')$ *and* $\Gamma'' \subseteq Sen''(\Sigma'')$ *we define* $\Gamma' +_\Gamma \Gamma'' \stackrel{def}{=} \gamma'(\Gamma') \cup \gamma''(\Gamma'')$. □

Remark 5 (Interpretation) *Note that, by pushout construction in* Set*, the elements of* Sen *are either singleton sets* $\{\varphi'\}$, $\{\varphi''\}$ *with*

$$\varphi' \in Sen(\Gamma') \setminus Sen'(\phi')(\alpha'(\Sigma)(Sen(\Sigma)))$$

and

$$\varphi'' \in Sen(\Gamma'') \setminus Sen''(\phi'')(\alpha''(\Sigma)(Sen(\Sigma))),$$

or they are equivalence classes of sentences from

$$Sen'(\phi')(\alpha'(\Sigma)(Sen(\Sigma))) \cup Sen''(\phi'')(\alpha''(\Sigma)(Sen(\Sigma))).$$

In other words, we can interpret $\Gamma' +_\Gamma \Gamma''$ *as a triple consisting of* Γ *and the two independent extensions* $\Gamma' \setminus Sen'(\phi')(\alpha'(\Sigma)(\Gamma))$ *and* $\Gamma'' \setminus Sen''(\phi'')(\alpha''(\Sigma)(\Gamma))$.

Remark 6 (Pullback of Models) *In such a way,* $\langle \Sigma' +_\Sigma \Sigma'', \Gamma' +_\Gamma \Gamma'' \rangle$ *is no longer an object in* $\mathsf{Flat}(\mathsf{Th})$*, since* $\Gamma' +_\Gamma \Gamma''$ *is not a set of* $\Sigma' +_\Sigma \Sigma''$ *sentences. That is, our answer to the question above is to extend* $\mathsf{Flat}(\mathsf{Th})$ *in an appropriate way. Appropriate means, especially, that the semantics of* $\langle \Sigma' +_\Sigma \Sigma'', \Gamma' +_\Gamma \Gamma'' \rangle$ *can nevertheless be defined by the following pullback diagram in* Cat

$$\begin{tikzcd}
Mod_{Lim}(\langle \Sigma' +_\Sigma \Sigma'', \Gamma' +_\Gamma \Gamma'' \rangle) \arrow[r, dotted, "Mod(\sigma'')"] \arrow[d, dotted, "Mod(\sigma')"'] & (\Phi_{\alpha''}; Mod''_{\models})(\langle \Sigma'', \Gamma'' \rangle) \arrow[d, "\beta''_{\models}"] \\
(\Phi_{\alpha'}; Mod'_{\models})(\langle \Sigma', \Gamma' \rangle) \arrow[r, "\beta'_{\models}"'] & Mod(\langle \Sigma, \Gamma \rangle)
\end{tikzcd}$$

In other words, the elements in $Mod_{Lim}(\langle \Sigma' +_\Sigma \Sigma'', \Gamma' +_\Gamma \Gamma'' \rangle)$ *are considered to be "distributed models", i.e., as triples* $\langle M', M, M'' \rangle$ *such that*

$$\beta'(\Sigma)(Mod'(\phi')(M')) = M = \beta''(\Sigma)(Mod''(\phi'')(M'')) \text{ and } M' \models' \Gamma', M'' \models' \Gamma''.$$

4 Towards a more Distributed View

The construction presented in the last section does not mix the syntax [1]. That is, each single sentence in a heterogeneous distributed specification is written in the syntax of a single signature from one of the involved institutions (compare

[1] There are, of course, also different mechanisms to combine the syntax of logical formalisms. See, for example [17, 19]

Remark 5). So, heterogeneous distributed specifications combine specifications from different institutions. Moreover, models are not single monolithic entities, but distributed coherent families of models (compare Remark 6). The crucial idea to formalize this kind of combination is to generalize the well-known comma category construction in a way, that we extend the concepts of "coherent families of objects and morphisms" in [7] (This concept was relaunched in [20] under the name "D-consistent family of models").

We consider again the basic diagram Δ of institution comorphisms

$$\mathcal{I}' \xleftarrow{\mu'=(\Phi',\alpha',\beta')} \mathcal{I} \xrightarrow{\mu''=(\Phi'',\alpha'',\beta'')} \mathcal{I}''$$

Since we are focussing on specifications we will work actually with the corresponding diagrams on the "fibred side" [27] according to proposition 2

$$\begin{tikzcd}
\mathsf{Th}(\mathsf{I}') \arrow[dr, "Mod'_{\models}"'] & \mathsf{Th}(\mathsf{I}) \arrow[l, "\Phi_{\alpha'}"'] \arrow[r, "\Phi_{\alpha''}"] \arrow[d, "Mod_{\models}"'] & \mathsf{Th}(\mathsf{I}'') \arrow[dl, "Mod''_{\models}"] \\
& \mathsf{Cat} &
\end{tikzcd}$$

$$\begin{tikzcd}
\mathsf{Sign}' & \mathsf{Sign} \arrow[l, "\Phi'"'] \arrow[r, "\Phi''"] & \mathsf{Sign}'' \\
\mathsf{Th}(\mathsf{I}') \arrow[u, "sign'"'] & \mathsf{Th}(\mathsf{I}) \arrow[l, "\Phi_{\alpha'}"'] \arrow[r, "\Phi_{\alpha''}"] \arrow[u, "sign"'] & \mathsf{Th}(\mathsf{I}'') \arrow[u, "sign''"']
\end{tikzcd}$$

Heterogeneous distributed specifications can be defined now straightforwardly by a generalization of the comma category construction

Definition 8 (Distributed Specifications) *We define the category* $\mathsf{Th}(\Delta)$ *of* Δ-distributed specifications *as follows: The objects are spans*

$$\langle \mathcal{I}', \langle \Sigma', \Gamma'' \rangle \rangle \xleftarrow{\langle \mu', \phi' \rangle} \langle \mathcal{I}, \langle \Sigma, \Gamma \rangle \rangle \xrightarrow{\langle \mu'', \phi'' \rangle} \langle \mathcal{I}'', \langle \Sigma'', \Gamma'' \rangle \rangle$$

of theory bridges and the morphisms are triples of theory bridges (specification *morphisms) such that the following diagram commutes*

$$\begin{tikzcd}
\langle \mathcal{I}', \langle \Sigma_1', \Gamma_1' \rangle\rangle \arrow[d, "\langle id_{\mathcal{I}'}, \psi' \rangle"'] & \langle \mathcal{I}, \langle \Sigma_1, \Gamma_1 \rangle\rangle \arrow[l, "\langle \mu', \phi_1' \rangle"'] \arrow[r, "\langle \mu'', \phi_1'' \rangle"] \arrow[d, "\langle id_{\mathcal{I}}, \phi \rangle"'] & \langle \mathcal{I}'', \langle \Sigma_1'', \Gamma_1'' \rangle\rangle \arrow[d, "\langle id_{\mathcal{I}''}, \psi'' \rangle"] \\
\langle \mathcal{I}', \langle \Sigma_2', \Gamma_2' \rangle\rangle & \langle \mathcal{I}, \langle \Sigma_2, \Gamma_2 \rangle\rangle \arrow[l, "\langle \mu', \phi_2' \rangle"'] \arrow[r, "\langle \mu'', \phi_2'' \rangle"] & \langle \mathcal{I}'', \langle \Sigma_2'', \Gamma_2'' \rangle\rangle
\end{tikzcd}$$

Remark 7 (Comma Categories) *That we generalize indeed the comma category construction becomes apparent by the fact that* $\mathsf{Th}(\Delta)$ *is actually a pullback of the following diagram*

$$\Phi_{\alpha'} \downarrow \mathsf{Th}(\mathsf{I}') \xrightarrow{\pi'_{Th}} \mathsf{Th}(\mathsf{I}) \xleftarrow{\pi''_{Th}} \Phi_{\alpha''} \downarrow \mathsf{Th}(\mathsf{I}'')$$

where $\Phi_{\alpha'} \downarrow \mathsf{Th}(\mathsf{I}')$ *and* $\Phi_{\alpha''} \downarrow \mathsf{Th}(\mathsf{I}'')$ *are comma categories and* π'_{Th} *and* π''_{Th} *are the corresponding projections into* Th. □

Remark 8 (Proper Distribution) *In* $\mathsf{Th}(\Delta)$ *we have only properly distributed specifications: An* $\mathcal{I}$*-specification* $\langle \Sigma, \Gamma \rangle$ *can be considered as a degenerated distributed specification because it can be presented by the span*

$$\langle \mathcal{I}', \Phi_{\alpha'}(\langle \Sigma, \Gamma \rangle)\rangle \xleftarrow{\langle \mu', id_{\Phi_{\alpha'}(\langle \Sigma, \Gamma \rangle)} \rangle} \langle \mathcal{I}, \langle \Sigma, \Gamma \rangle\rangle \xrightarrow{\langle \mu'', id_{\Phi_{\alpha''}(\langle \Sigma, \Gamma \rangle)} \rangle} \langle \mathcal{I}'', \Phi_{\alpha''}(\langle \Sigma, \Gamma \rangle)\rangle$$

Globally this coding provides a functor from $\mathsf{Th}(\mathsf{I})$ *into* $\mathsf{Th}(\Delta)$. *Contrarly, single specifications from* $\mathcal{I}'$ *or* $\mathcal{I}''$*, respectively, can be not represented in a natural way as* Δ*-distributed specifications. To do this, we need something like "the maximal* $\mathcal{I}$*-specification" contained in a given* $\mathcal{I}'$*- or* $\mathcal{I}''$*-specification. One may even take as the other components empty/initial specifications as long as they exist and are preserved.*

Similarly, we can define the category $\mathsf{Sign}(\Delta)$ of Δ-distributed signatures by the pullback of the following diagram

$$\Phi' \downarrow \mathsf{Sign}' \xrightarrow{\pi'_{\mathsf{Sign}}} \mathsf{Sign} \xleftarrow{\pi''_{\mathsf{Sign}}} \Phi'' \downarrow \mathsf{Sign}''$$

That is, as objects in $\mathsf{Sign}(\Delta)$ we take triples

$$\langle \Sigma' \stackrel{\phi'}{\leftarrow} \Phi'(\Sigma), \Sigma, \Phi''(\Sigma) \stackrel{\phi''}{\rightarrow} \Sigma'' \rangle$$

and morphisms are compatible triples of signature morphims. The pullback property will provided, in such a way, also a Δ-distributed functor $sign_\Delta : \mathsf{Th}(\Delta) \rightarrow \mathsf{Sign}(\Delta)$.

The main point is that we do not mix syntax by constructing $\mathsf{Sign}(\Delta)$, i.e., the concept of a Δ-sentence makes no sense. But, we can define a concept of Δ-distributed models (compare remark 6). Moreover, we can define a concept of Δ-distributed satisfaction for Δ-distributed models and Δ-distributed specifications thus arriving at an adequate model theory for heterogeneous distributed specifications.

Since the approach, sketched in this section, is based on well-behaved categorical constructions, there should be no severe problems to validate the outlined constructions in detail and to extend them to arbitrary diagrams Δ.

5 Conclusions and Future Work

In this paper we have investigated models for heterogeneous distributed (logical) specifications. In [9, 20] it has been shown that under certain mathematical conditions those models can be described in terms of the Grothendieck Institution related to the application in question.

We have also addressed here the question: "What can be the models for heterogeneous distributed (logical) specifications if the involved model functors are not assumed to be semi-exact?" We have investigated two different approaches. In the first one, as a more extensional approach, we have reported that a fundamental property for heterogeneous systems integration holds. The colimit built in $\mathsf{InstCom}_{coc}$ serves as a basis for step-wise colimits of the corresponding specifications. As one could expect, the constructions, being natural, are universal. The universality is used also in the colimit constrution inside $\mathsf{InstCom}_{coc}$, for building integration of heterogeneous systems in a natural way. Finally, this can be taken as a foundation for formal interoperability, since the objetcs in $\mathsf{InstCom}_{coc}$ are formal systems (specifications) represented in different institutions (specification formalisms). This is very important, because otherwise, we should have assumed a kind of universal institution where all the other logics could be faithfully embedded. However, as we all know, this is usually not the case in formal analysis, but the final implementation step.

The second approach has been only sketched and can be seen as a more structured revision of the first approach. The crucial idea is to model distributed specifications as first class citizens, based on the notions of coherent class of morphims (Δ-diagrams) and based objects. This is an essential development step in the model, since, as we have described in the first one, sentences of different specifications are actually never mixed in a colimit construction.

One should observe that this work should be seen as a project outline for having a theoretical model of distributed specifications which reflect the usual intentions of the working specifier and programmer. It seems a promising idea that can allow for the formalization of practical features in programming and design as "environments" and "libraries" in a distributed yet heterogeneous setting.

The next steps will be to develop the second approach in full detail, to apply it, and to relate it to other approaches. Especially we have to address the following issues: Does this approach recovers the idea of a “universal institution” $\mathcal{I}_u$ or not? Is this approach equivalent the Grothendieck Institution approach in case all the conditions in [9, 20] are satisfied?

References

[1] Richard M. Adler. Emerging standards for component software. *IEEE Computer*, pages 68–77, March, 1995.

[2] R. Allen and D. Garlan. Formalizing architectural connection. In *16th Int. Conference on Software Engineering*, pages 71–80, 1994.

[3] K. J. Bairwise. Axioms for Abstract Model Theory. *Annals of Mathematical Logic*, 7:221–265, 1974.

[4] G. Birkhoff. On the structure of abstract algebras. In *Proc. Cambridge Philos. Soc.*, pages 433–454, 1935.

[5] R. M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proc. Int. Conf. Artificial Intelligence*, 1977.

[6] R. M. Burstall and J. A. Goguen. The semantics of Clear, a specification language. In D. Bjøner, editor, *Abstract Software Specification, Proc. 1979 Copenhagen Winter School, LNCS 86*, pages 292–332. Springer, 1980.

[7] I. Classen. *Compositionality of Application Oriented Structuring for Algebraic Specification Languages with Initial Semantics.* PhD thesis, Technical University Berlin, Berlin, 1993.

[8] R. Diaconescu. Extra-theory morphisms in institutions: logical semantics for multi-paradigm languages. *Journal Applied Categorical Structures*, 1998.

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

[10] T.M. Maibaum J.L. Fiadeiro E. H. Haeusler, A. Haeberer. Arts: A formally supported environment for object oriented software development. In *Automating the Object-Oriented Software Development Workshop, ECOOP'98.*

[11] J. Fiadeiro. *Categories for Software Engineering.* Springer, Berlin, 2005.

[12] J. A. Goguen and R. M. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the ACM*, 39(1):95–146, January 1992.

[13] J. A. Goguen and G. Roşu. Institutions morphisms. *Formal Aspects of Computing*, 12(3-5):274–307, July 2002.

[14] M. Wermelinger J.L. Fiadeiro, A. Lopes. A mathematical semantics for architectural connectors. *Genneric Programming*, pages 178–221, 2003.

[15] A. Martini. *Relating Arrows between Institutions in a Categorical Framework*. PhD thesis, Technical University of Berlin, 1999.

[16] J. Meseguer. General logics. In H.-D. Ebbinghaus et. al., editor, *Logic colloquium '87*, pages 275–329. Elsevier Science Publishers B. V.,North Holland, 1989.

[17] J. Meseguer and U. Montanari. Mapping Tile Logic into Rewriting Logic. In F. Parisi Presicce, editor, *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Tarquinia, June 1997, Selected Papers*, pages 62–91. Springer Verlag, LNCS 1376, 1998.

[18] Ieke Moerdijk and Saunders Mac Lane. *Sheaves in Geometry and Logic*. Springer Berlag, New York, 1992.

[19] A. Mossakowski, T. Tarlecki and W. Pawlowski. Combining and Representing Logical Systems Using Model-Theoretic Parchments. In F. Parisi Presicce, editor, *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Tarquinia, June 1997, Selected Papers*, pages 349–364. Springer Verlag, LNCS 1376, 1998.

[20] T. Mossakowski. Comorphism-based Grothendieck logics . In *Mathematical Foundations of Computer Science 2002, Proceedings*, pages 593–604. 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Springer Verlag, LNCS 2420, 2002.

[21] T. Mossakowski. Foundations of Heterogeneous Specification. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Recent Trends in Algebraic Development Techniques, 16th International Workshop, WADT 2002, Frauenchiemsee, Germany, September 24-27, 2002, Revised Selected Papers*, pages 359–375. Springer Verlag, LNCS 2755, 2003.

[22] J.L. Fiadeiro S. E. R. Carvalho, E. H. Haeusler. A formal approach to real time object oriented software. In *Proceedings of the Workshop on Real-Time Programming*.

[23] Mary Shaw and David Garlan. Characteristics of higher-level languages for software architecture. Technical Report CMU-CS-94-210, Carnegie Mellon University, 1994.

[24] A. Tarlecki, R.M. Burstall, and J.A. Goguen. Some fundamental algebraic tools for the semantics of computation. Part III: Indexed categories. *TCS*, 91:239–264, 1991.

[25] U. Wolter, K. Didrich, F. Cornelius, M. Klar, R. Wessäly, and H. Ehrig. How to Cope with the Spectrum of Spectrum. Technical Report Bericht-Nr. 94-22, TU Berlin, FB Informatik, 1994.

[26] U. Wolter, M. Klar, R. Wessäly, and F. Cornelius. Four Institutions – A Unified Presentation of Logical Systems for Specification. Technical Report Bericht-Nr. 94-24, TU Berlin, Fachbereich Informatik, 1994.

[27] U. Wolter and A. Martini. Shedding New Light in the World of Logical Systems. In E. Moggi and G. Rosolini, editors, *Category Theory and Computer Science, 7th International Conference, CTCS'97*, pages 159–176. Springer, LNCS 1290, 1997.

Edward Hermann Haeusler
Department of Informatics
PUC-RJ - Rio de Janeiro - Brasil
e-mail: hermann@inf.puc-rio.br

Alfio Martini
Faculty of Informatics
PUCRS - Porto Alegre - Brasil
e-mail: alfio@inf.pucrs.br

Uwe Wolter
Department of Informatics
University of Bergen - Norway
e-mail: wolter@ii.uib.no