A Heterogeneous Characterisation of Component-Based System Design in a Categorical Setting

Carlos G. Lopez Pombo[1,3], Pablo F. Castro[2,3], Nazareno Aguirre[2,3], and Tomas S.E. Maibaum[4]

[1] Department of Computing, FCEyN, Universidad de Buenos Aires
[2] Department of Computing, FCEFQyN, Universidad Nacional de Río Cuarto
[3] Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICET)
[4] Department of Computing & Software, McMaster University

Abstract. In component-based design, components and communication mechanisms have a different nature; while the former represent the agents that cooperate to fulfill a certain goal, the latter formalise the communication mechanism through which these agents interact. A proper formalisation of the heterogeneity that arises from this difference requires one to employ the most adequate formalism for each of the parts of a specification and then proceed to merge the parts of the system specification characterised in different languages. The approach we propose in this paper is based on the notion of *institution*, and makes extensive use of *institution representations* in order to relate the specifications of components and communication mechanisms, each of which might be expressed in different formalisms. The contribution focuses on providing tools needed to engineer heterogeneous languages arising from particular choices for the specification of components and communication devices.

1 Introduction

Nowadays, software artefacts are ubiquitous in our lives, being an essential part of home appliances, cars, cell phones, and even in more critical activities like aeronautics and health sciences. In this context, software failures may produce enormous losses, either economical or, in the worst case, in human lives. In order to provide better guarantees for the correct functioning of software, various elements are necessary, among which *formal foundations*, that enable a precise reasoning about software, and *modularity*, that helps in dealing with software complexity, are crucial.

The importance of modularity has been acknowledged since the foundational work of Parnas, which promoted building software artefacts (and more specifically software specifications) modularly, enhancing reusability of modules and contributing to a better separation of concerns, and leading to an improved quality in specification and development. Modularisation is generally understood as the process of dividing a system specification, or implementation, into manageable parts: the *modules* or *components*. It leads to a structural view of systems, called *architecture*, as described in [16], in which the relevance of *component interaction* is brought out. Aside from its crucial relevance for managing the complexity of systems, a system's architectural structure also plays an important role in its functional and non-functional characteristics.

Given the relevance of software architecture, its formal foundations are essential to guarantee the correct functioning of component based systems. There exist various approaches to formally capture component based systems, which are either language-specific (e.g., formalisations of schema operators in Z or structuring mechanisms in B), making its results difficult to generalise to other component-based settings, or target specific ways of communicating components (e.g., formalising particular communication mechanisms, such as synchronisation in process algebra based approaches). Moreover, these approaches support communication mechanisms that are influenced (or defined) by the nature of the components they communicate (e.g., synchronisation in event based models, or shared memory in state based models).

In this work, we tackle the above described limitations of existing formalisations of component based systems by introducing an *abstract* and *heterogeneous* categorical characterisation of component-based systems. Our characterisation is presented in a logic or language independent setting, by making use of the notion of *institution*. Moreover, the approach is *heterogeneous*, favouring a more genuine separation of concerns in the specification of components, and that of the communication mechanisms. Finally, although there exist other abstract and heterogeneous approaches, most notably the work related to The Heterogeneous Tool Set HETS [26, 27], our approach differs from these in that it focuses on providing a formal characterisation of the elements of the domain of component-based software architecture (to be further discussed in Sec. 5).

The practical usefulness of heterogeneous specification formalisms in the context of component-based systems is acknowledged by the existence of languages such as Acme [17] (and others), designed with the aim of putting together specifications originating in different formalisms. Generally, heterogeneity arises from two different angles: it arises from the fact that the description of each component or module could be given in a different specification language; and as a consequence of components and communication mechanisms being of a different nature. The existing literature concentrates on the first kind of heterogeneity; we will devote this work to providing formal foundations for the second one.

The formal tools we use in this paper are those coming from the field of category theory, more specifically from the domain of algebraic specifications [6]. They have been shown to be useful for enabling the formal characterisation of different kinds of specification structuring mechanisms and refinement in different settings; a few examples are: [7, 9, 22, 33]. We employ a well established abstract definition of logical systems, known as *institutions* [18], to achieve generality (in the sense of the approach being language independent). In order to appropriately combine different formalisms, we make extensive use of *institution representations* [30]. These serve the purpose of relating and, consequently, combining different (abstract) logics used for different purposes in a given specification, e.g., those used for component and connector specifications.

In summary, the main contributions of this work are: *a*) providing a formal, and language independent, interpretation of the concepts arising from the field of software architecture, and *b*) providing formal foundations for the heterogeneity observed when components interact through communication channels.

2 Preliminaries

From now on, we assume that the reader has a nodding acquaintance with the basic definitions of category theory, including the concepts of category, functor, natural transformation, etc. We mainly follow the notation of [23]: given a category $\mathbf{C}$, $|\mathbf{C}|$ denotes its collection of objects, while $||\mathbf{C}||$ denotes its collection of arrows. $g \circ f : A \to C$ denotes the composition of arrows $f : A \to B$ and $g : B \to C$. Natural transformations will be indicated with the arrow $\dot{\to}$.

The theory of institutions [18] provides a formal definition of logical system, and of how specifications in a logical system can be put together. They also serve as a suitable framework for addressing the problem of heterogeneity [27, 32].

Definition 1. *An* institution *is a structure of the form* $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ *satisfying the following conditions: a*) Sign *is a category of signatures, b*) $\mathbf{Sen} : \mathsf{Sign} \to \mathsf{Set}$ *is a functor (let* $\Sigma \in |\mathsf{Sign}|$*, then* $\mathbf{Sen}(\Sigma)$ *corresponds to the set of* Σ*-formulae), c*) $\mathbf{Mod} : \mathsf{Sign}^{\mathsf{op}} \to \mathsf{Cat}$ *is a functor (let* $\Sigma \in |\mathsf{Sign}|$*, then* $\mathbf{Mod}(\Sigma)$ *corresponds to the category of* Σ*-models), d*) $\{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}$*, where* $\models^{\Sigma} \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$*, is a family of binary relations. Such that, for every signature morphism* $\sigma : \Sigma \to \Sigma' \in ||\mathsf{Sign}||$*,* $\phi \in \mathbf{Sen}(\Sigma)$ *and* $\mathcal{M}' \in |\mathbf{Mod}(\Sigma)|$ *the following* $\models$*-invariance condition must hold:* $\mathcal{M}' \models^{\Sigma'} \mathbf{Sen}(\sigma)(\phi)$ *iff* $\mathbf{Mod}(\sigma^{\mathsf{op}})(\mathcal{M}') \models^{\Sigma} \phi$.

Institutions are an abstract formulation of the notion of logical system, more specifically, of its model theory, where the concepts of languages, models and truth are characterised in a category theoretic way. Examples of institutions are: propositional logic, equational logic, first-order logic, first-order logic with equality, dynamic logics and temporal logics (a detailed list is given in [18]).

Let $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|}\rangle$ be an institution, $\Sigma \in |\mathsf{Sign}|$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$ then, we define the functor $\mathbf{Mod}(\Sigma, \Gamma)$ as the full subcategory of $\mathbf{Mod}(\Sigma)$ determined by those models $\mathcal{M} \in |\mathbf{Mod}(\Sigma)|$ such that for all $\gamma \in \Gamma$, $\mathcal{M} \models^{\Sigma} \gamma$. We also overload the symbol $\models^{\Sigma}$, to define a relation between sets of formulae and formulae, as follows: $\Gamma \models^{\Sigma} \alpha$ if and only if $\mathcal{M} \models^{\Sigma} \alpha$ for all $\mathcal{M} \in |\mathbf{Mod}(\Sigma, \Gamma)|$. where $\alpha \in \mathbf{Sen}(\Sigma)$.

Definition 2. *We define the category of theory presentations as the pair* $\langle \mathcal{O}, \mathcal{A}\rangle$ *(usually denoted as* Th*) where:* $\mathcal{O} = \{\, \langle \Sigma, \Gamma\rangle \mid \Sigma \in |\mathsf{Sign}| \text{ and } \Gamma \subseteq \mathbf{Sen}(\Sigma) \,\}$*, and* $\mathcal{A} = \left\{ \sigma : \langle \Sigma, \Gamma\rangle \to \langle \Sigma', \Gamma'\rangle \;\middle|\; \begin{array}{l} \langle \Sigma, \Gamma\rangle, \langle \Sigma', \Gamma'\rangle \in \mathcal{O}, \sigma : \Sigma \to \Sigma' \in ||\mathsf{Sign}|| \\ \text{and } \Gamma' \models^{\Sigma'} \mathbf{Sen}(\sigma)(\Gamma) \end{array} \right\}$.

In addition, if a morphism $\sigma : \langle \Sigma, \Gamma \rangle \to \langle \Sigma', \Gamma' \rangle$ satisfies $\mathbf{Sen}(\sigma)(\Gamma) \subseteq \Gamma'$, it is called *axiom preserving*. By keeping only those morphisms of Th which are axiom preserving we obtain the category Th_0.

As we mentioned before, an institution is a structure $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|} \rangle$ so from now on, whenever we make reference to a given institution I we will be referring to the structure $\mathsf{I} = \langle \mathsf{Sign}^{\mathsf{I}}, \mathbf{Sen}^{\mathsf{I}}, \mathbf{Mod}^{\mathsf{I}}, \{\models^{\mathsf{I}}_{\Sigma}\}_{\Sigma \in |\mathsf{Sign}^{\mathsf{I}}|} \rangle$.

Next, we define the notion of *institution representation* (also find in the literature under the name of co-morphism)[30, Def. 12, Sec. 5].

Definition 3. *Let I and I' be institutions then, the structure $\langle \gamma^{Sign}, \gamma^{Sen}, \gamma^{Mod} \rangle : I \to I'$ is an* institution representation *if and only if: a) $\gamma^{Sign} : \mathbf{Sign} \to \mathbf{Sign}'$ is a functor, b) $\gamma^{Sen} : \mathbf{Sen} \dot{\to} \mathbf{Sen}' \circ \gamma^{Sign}$, is a natural transformation such that for every $\Sigma_1, \Sigma_2 \in |\mathbf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2 \in ||\mathbf{Sign}||$, $\gamma^{Sen}_{\Sigma_2} \circ \mathbf{Sen}(\sigma) = \mathbf{Sen}'(\gamma^{Sign}(\sigma)) \circ \gamma^{Sen}_{\Sigma_1}$, c) $\gamma^{Mod} : \mathbf{Mod}' \circ (\gamma^{Sign})^{\mathsf{op}} \dot{\to} \mathbf{Mod}$, is a natural transformation such that for every $\Sigma_1, \Sigma_2 \in |\mathbf{Sign}|$ and $\sigma : \Sigma_1 \to \Sigma_2 \in ||\mathbf{Sign}||$, $\mathbf{Mod}(\sigma^{\mathsf{op}}) \circ \gamma^{Mod}_{\Sigma_2} = \gamma^{Mod}_{\Sigma_1} \circ \mathbf{Mod}'((\gamma^{Sign})^{\mathsf{op}}(\sigma^{\mathsf{op}}))$, such that, for any $\Sigma \in |\mathbf{Sign}|$, $\gamma^{Sen}_{\Sigma} : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}'(\gamma^{Sign}(\Sigma))$ and $\gamma^{Mod}_{\Sigma} : \mathbf{Mod}'(\gamma^{Sign}(\Sigma)) \to \mathbf{Mod}(\Sigma)$ preserve the following satisfaction condition: for any $\alpha \in \mathbf{Sen}(\Sigma)$ and $\mathcal{M}' \in |\mathbf{Mod}(\gamma^{Sign}(\Sigma))|$, $\mathcal{M}' \models_{\gamma^{Sign}(\Sigma)} \gamma^{Sen}_{\Sigma}(\alpha)$ iff $\gamma^{Mod}_{\Sigma}(\mathcal{M}') \models_{\Sigma} \alpha$.*

An institution representation intuitively corresponds to the relationship between a given institution, and how it is interpreted into another one, in a semantics preserving way. It may be regarded as a realisation for institutions of the established concept of property preserving translation between two theories in two different logics.

Property 1. Let $\mathbf{I}$ and $\mathbf{I}'$ be institutions, and $\rho : \mathbf{I} \to \mathbf{I}'$ an institution representation. For every signature $\Sigma \in |\mathbf{Sign}|$, set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, and Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$, if $\Phi \models_{\Sigma} \varphi$, then $\rho^{Sen}_{\Sigma}(\Phi) \models'_{\rho^{Sign}(\Sigma)} \rho^{Sen}_{\Sigma}(\varphi)$.

We will also use a few other categorical concepts, such as that of a *bicategory* and a *lax functor*; the interested reader is referred to [23].

3 A Characterisation of Component-Based Design

Two main goals of our approach to component-based specification are *generality* and *abstraction*, in the sense that we have designed our characterisation of component-based systems to be specification language independent. We start by providing a category theoretic characterisation of the concepts used in the field of software architecture. There, the building blocks for software design are *components*, *glues*, *ports*, *roles* and *adaptors*.

Components A component describes an independent computational unit of a system. They are formally characterised by theories in a given institution. Note that the approach presented here does not prevent the introduction of another level of heterogeneity where each component is specified in a different formalism,

Component: *Producer*
Attributes: *p-current*: Bit, *p-waiting*: Bool,
ready-in: Bool
Actions: *produce-0*, *produce-1*, *send-0*, *send-1*,
p-init
Axioms:
1. $\Box(\textit{p-init} \to \bigcirc(\textit{p-current} = 0 \wedge \neg\textit{p-waiting}))$
2. $\Box(\textit{produce-0} \vee \textit{produce-1} \to \neg\textit{p-waiting} \wedge$
$\bigcirc\textit{p-waiting})$
3. $\Box(\textit{produce-0} \to \bigcirc(\textit{p-current} = 0))$
4. $\Box(\textit{produce-1} \to \bigcirc(\textit{p-current} = 1))$
5. $\Box((\textit{send-0} \to \textit{p-current} = \textit{0}) \wedge$
$(\textit{send-1} \to \textit{p-current} = \textit{1}))$
6. $\Box(\textit{send-0} \vee \textit{send-1} \to \textit{p-waiting} \wedge$
$\bigcirc\neg\textit{p-waiting})$
7. $\Box(\textit{send-0} \vee \textit{send-1} \to \textit{ready-in} \wedge$
$\textit{p-current} = \bigcirc\textit{p-current})$
8. $\Box(\textit{send-0} \vee \textit{send-1} \vee \textit{produce-0} \vee$
$\textit{produce-1} \vee \textit{p-init} \vee$
$(\textit{p-current} = \bigcirc\textit{p-current} \wedge$
$\textit{p-waiting} = \bigcirc\textit{p-waiting}))$

Component: **Consumer**
Attributes: *c-current*: Bit, *c-waiting*: Bool,
ready-ext: Bool
Actions: *consume*, *extract-0*, *extract-1*, *c-init*
Axioms:
1. $\Box(\textit{c-init} \to \bigcirc(\textit{c-current} = 0 \wedge \textit{c-waiting}))$
2. $\Box(\textit{extract-0} \vee \textit{extract-1} \to \textit{c-waiting} \wedge$
$\bigcirc\neg\textit{c-waiting} \wedge \textit{ready-ext})$
3. $\Box(\textit{extract-0} \to \bigcirc(\textit{c-current} = 0))$
4. $\Box(\textit{extract-1} \to \bigcirc(\textit{c-current} = 1))$
5. $\Box(\textit{consume} \to \neg\textit{c-waiting} \wedge \bigcirc\textit{c-waiting})$
6. $\Box(\textit{consume} \to \textit{c-current} = \bigcirc\textit{c-current})$
7. $\Box(\textit{consume} \vee \textit{extract-0} \vee \textit{extract-1} \vee \textit{c-init} \vee$
$(\textit{c-current} = \bigcirc\textit{c-current} \wedge$
$\textit{c-waiting} = \bigcirc\textit{c-waiting}))$

Fig. 1. A Producer-Consumer specification

in which case the language must deal with this internally, e.g., by instantiating the institution for describing components with a Grothendieck institution formed by all the specification languages needed for the task, as proposed in [5, 27]. Without loss of generality we will assume a single language for component specification formalised as an institution, which will be referred to as $\mathsf{I}^{\mathsf{Comp}}$.

Example 1. (A simple producer and consumer specification) In Fig. 1 we present a formalisation of a producer and a consumer in *propositional temporal logic* [13], a specification language based on *linear temporal logic* [24]; as a consequence, our specifications are state-based. For simplicity, we assume that messages are bits identified by the type *Bit*. The producer's state is defined by a bit-typed attribute *p-current* to store a produced element, a *boolean* attribute *p-waiting* to indicate whether an item is already produced and ready to be sent (so that null values for items are not necessary), and a boolean attribute *ready-in*, so that a producer is informed by the environment when this is ready to receive a product. This specification consists of a set of sorts (*Bit* and *Bool*, in this case), a set of attributes (i.e., flexible variables), some of which are supposed to be controlled by the environment, and a set of action symbols (they are flexible boolean variables indicating the occurrence of an action). The axioms of the specification are linear temporal logic formulae characterising the behaviour of the component, in a rather obvious way. The consumer component can be specified in a similar way.

Notice that these components are coherent with the notion of component in software architecture [16]. The theory associated with a component represents the computational aspects of it, in our case indicating via axioms (and their corresponding consequences) the behaviour of the actions of the component and their effect on its state. It is worth noticing that components described below do not formalise any aspect of the communication between them.

Ports Ports constitute the communication *interfaces* of a component. As in [13], ports can be captured by using *channels*, which consist of theories with no axioms. Given $A \in |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}|$, a port for A is a morphism $\sigma : \mathbf{Th}(\Sigma) \to A \in ||\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}||$ such that $\Sigma \in |\mathsf{Sign}^{\mathsf{I}^{\mathsf{Comp}}}|$.[5]

In software architecture, emphasis is put on the explicit description of communication aspects of a system, separated from the computational, component related, aspects. Below we define the elements relevant in our formalisation, various of which are inspired by the formal approach to interaction characterisation put forward in [12].

Glues In a communication mechanism between two components, a glue captures the way in which these components interact, that is, computational aspects of the interaction, e.g., a protocol. In our setting, glues are also required to be organised in an institution, thus being theories in a given specification language. From now on, the institution used to specify glues will be denoted as $\mathsf{I}^{\mathsf{Glue}}$. Also in this case, similarly to components, the use of more than one specification language can be considered, so different glues can be described by means of different languages.

Roles Roles constitute the interfaces of glues. Thus, given $G \in |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Glue}}}|$, a role for G is a morphism $\pi : \mathbf{Th}(\Sigma) \to G \in ||\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Glue}}}||$ such that $\Sigma \in |\mathsf{Sign}^{\mathsf{I}^{\mathsf{Glue}}}|$.

Connector A connector represents a mechanism for interconnecting two components, and establishes: *a*) the roles of the glue, and *b*) the glue itself. Given a glue $G \in |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Glue}}}|$, and the roles $\sigma_1 : \mathbf{Th}(\Sigma_1) \to G, \sigma_2 : \mathbf{Th}(\Sigma_2) \to G \in ||\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Glue}}}||$ for G, a connector with behaviour G and roles σ_1 and σ_2, is a structure of the form $\langle \sigma_1, G, \sigma_2 \rangle$. If we restrict ourselves to binary connectors, they are required to be organised as a subcategory of the bicategory $\mathbf{co-spans}(\mathsf{Th}_0^{\mathsf{I}^{\mathsf{Glue}}})$, denoted by $\mathbf{Connector}(\mathsf{I}^{\mathsf{Glue}})$. The generalisation to n-ary connectors is straightforward but their category theoretic characterisation is no longer a bicategory but a generalised version of it.

As mentioned above, heterogeneity arises, in architecture description languages, as a reflection of the different nature of components and communication mechanisms. This led us to the use of separate institutions for formalising components (and their ports) and the glues (and their roles). So, we need a way of establishing the relationship between component and connector specifications, or more generally, between their corresponding institutions. There exist various mechanisms for relating institutions, each with a particular meaning when interpreted in the context of software design. The interested reader is referred to [30, 19], where the authors make a thorough study of these mechanisms. We

[5] $\mathbf{Th} : \mathsf{Sign}^{\mathsf{I}^{\mathsf{Comp}}} \to \mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}$ is the right adjoint of the forgetful functor $\mathbf{Sign} : \mathsf{Th}^{\mathsf{I}^{\mathsf{Comp}}} \to \mathsf{Sign}^{\mathsf{I}^{\mathsf{Comp}}}$.

can use Property 1 to draw the relationship between the institutions $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ needed to be able to obtain a complete description of a system as communicating components. Such a relationship is captured as follows. Let $\mathsf{I}^{\mathsf{Sys}}$ be an institution, and let $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ and $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ be institution representations. Thus, $\mathsf{I}^{\mathsf{Sys}}$ serves as a common formal language, in which the components and connectors of a system can be interpreted and put together

Example 2. (Connecting components and connectors directly) A straightforward way of establishing links between components and connectors is by requiring the roles and the ports one wants to connect to be equal when they are translated to the $\mathsf{I}^{\mathsf{Sys}}$ institution. This situation is illustrated in Fig. 2, a) for a component denoted as **A** and a glue denoted as **G**; connections between **G** and a different component **B** are analogous. Observing the upper part of the diagram, component **A** communicates using ports $\pi_A \to \mathbf{A}$, using a medium characterised by the connector formed by the glue **G** and the role $\rho_l \to \mathbf{G}$, to be attached to the port $\pi_A \to \mathbf{A}$. Dashed arrows express the application of the corresponding institution representation to the theories and morphisms appearing in the upper part of the diagram in order to provide a homogeneous description of the whole system in the institution $\mathsf{I}^{\mathsf{Sys}}$. The bottom part of the diagram shows how things are put together in $\mathsf{I}^{\mathsf{Sys}}$, thus obtaining a diagram, in the usual sense of category theory, consisting of the behaviour associated with component **A** and glue **G**.

This simple way of connecting the components, though correct, has some limitations. The differences between $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ might not be merely syntactical, but sometimes their semantics also need to be "*harmonized*". Assume, for instance, that we use Propositional Dynamic Logic (PDL) [20] in order to describe the components of a system, whereas the glues are formalised in Linear Temporal Logic (LTL). The models of these two logics have different structures, since LTL models are interpret formulas along traces, while PDL models have state-based semantics. Even when a more expressive logic might be capable of interpreting both PDL and LTL theories, the coordination of the semantic objects cannot always be obtained merely by a syntactic identification in the more expressive logic. Following the principles of software architecture, we deal with this problem using so called *adaptors.*

Adaptor An adaptor is a connector in $\mathsf{I}^{\mathsf{Sys}}$. The intuition behind the inclusion of adaptors is that roles will interact with ports, not only at a syntactic level as shown in Ex. 2, but mediated by a semantic synchronisation of models, induced by the axioms of the theory characterising the connector. Adaptors in software architecture serve the purpose of solving or alleviating architectural mismatches. In our case, the (potential) mismatch is related to the difference between the logics used for the specification of the components and the connectors.

Example 3. Adaptors help in establishing the links between roles and ports. This situation is illustrated in Fig. 2 b) for a component denoted as **A**, a glue denoted as **G** and an adaptor Γ_{GA}; connections between **G** and a different component **B** are analogous.

$$\begin{tikzcd}
\mathbf{A} \arrow[dd, dashed, "\gamma^{C\cdot}"'] & & & & \mathbf{G} \arrow[dd, dashed, "\gamma^{G\cdot}"] \\
 & \pi_A \arrow[ul] \arrow[ddr, dashed, "\gamma^{C\cdot}"'] & & \rho_l \arrow[ur] \arrow[ddl, dashed, "\gamma^{G\cdot}"] & \\
\gamma^{C\cdot}(\mathbf{A}) & & & & \gamma^{G\cdot}(\mathbf{G}) \\
 & & \begin{array}{c}\gamma^{C\cdot}(\pi_A)\\ =\\ \gamma^{G\cdot}(\rho_l)\end{array} \arrow[ull] \arrow[urr] & &
\end{tikzcd}$$

a) Conn. of comp. and glues by sharing ports and roles.

$$\begin{tikzcd}
\mathbf{A} \arrow[dd, dashed, "\gamma^{C\cdot}"'] & & & \Gamma_{GA} \arrow[ddd, dashed, "id^{\mathsf{I}^{\mathsf{Sys}}}"] & & & \mathbf{G} \arrow[dd, dashed, "\gamma^{G\cdot}"] \\
 & \pi_A \arrow[ul] \arrow[ddd, dashed, "\gamma^{C\cdot}"'] & \delta_A \arrow[ur, "\delta_{GA_l}"] \arrow[dddl, dashed, "id^{\mathsf{I}^{\mathsf{Sys}}}"] & & \delta_{G_l} \arrow[ul, "\delta_{GA_r}"'] \arrow[dddr, dashed, "id^{\mathsf{I}^{\mathsf{Sys}}}"'] & \rho_l \arrow[ur, "\pi_l"] \arrow[ddd, dashed, "\gamma^{G\cdot}"] & \\
\gamma^{C\cdot}(\mathbf{A}) & & & & & & {} \\
 & & & \Gamma_{GA} & & & \\
 & \begin{array}{c}\gamma^{C\cdot}(\pi_A)\\ =\\ \delta_A\end{array} \arrow[uuul] \arrow[uurr] & & & & \begin{array}{c}\gamma^{G\cdot}(\rho_l)\\ =\\ \delta_{G_l}\end{array} \arrow[uull] \arrow[uuur] &
\end{tikzcd}$$

b) Conn. of comp. and glues by means of adaptors.

Fig. 2. Connections between components, glues, adaptors, roles and ports. **Note:** $\gamma^{C\cdot} \to \gamma^{Comp^{\mathsf{Th}_0}}$, $\gamma^{G\cdot} \to \gamma^{Glue^{\mathsf{Th}_0}}$.

Connection A *connection* is formed by a connector together with a pair of adaptors linking the ports of the components participating in the communication. Let $\mathsf{I}^{\mathsf{Sys}}$, $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ be institutions, $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ and $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ be institution representations. Let $\pi = \langle \pi_l : \rho_l \to G, G, \pi_r : \rho_r \to G \rangle \in |\mathbf{Connector}(\mathsf{I}^{\mathsf{Glue}})|$ and $\delta_{GA} = \langle \delta_{GA\,l} : \delta_A \to \Gamma_{GA}, \Gamma_{GA}, \delta_{GA\,r} : \delta_{G_l} \to \Gamma_{GA} \rangle, \delta_{GB} = \langle \delta_{GB_l} : \delta_{G_r} \to \Gamma_{GB}, \Gamma_{GB}, \delta_{GB_r} : \delta_B \to \Gamma_{GB} \rangle \in |\mathbf{Connector}(\mathsf{I}^{\mathsf{Sys}})|$. Then a *connection* is a structure of the form $\langle \delta_{GA}, \pi, \delta_{GB} \rangle$ such that $\delta_{G_l} = \gamma^{Glue^{\mathsf{Th}_0}}(\rho_l)$ and $\delta_{G_r} = \gamma^{Glue^{\mathsf{Th}_0}}(\rho_r)$. Given the institutions $\mathsf{I}^{\mathsf{Sys}}$ and $\mathsf{I}^{\mathsf{Glue}}$ such that $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ is an institution representation, the connections definable over these two institutions will be the complete subcategory of $\mathbf{Connector}(\mathsf{I}^{\mathsf{Sys}}) \times \mathbf{Connector}(\mathsf{I}^{\mathsf{Glue}}) \times \mathbf{Connector}(\mathsf{I}^{\mathsf{Sys}})$ whose objects are those triples satisfying the conditions stated above and will be denoted as $\mathbf{Connection}(\mathsf{I}^{\mathsf{Glue}}, \mathsf{I}^{\mathsf{Sys}})$.

The previous definitions allow us to formalise in a categorical setting the main notions involved in component-based designs as a labeled graph. The next definition formalises graph labelings.

Definition 4. *Let $\mathsf{I}^{\mathsf{Sys}}$, $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ be institutions, and $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ and $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ be institution representations. Let $G = \langle V, E \rangle$ be a graph; then a* labeling *ι for G is a structure of the form $\langle f : V \to |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}|, p : V \to 2^{||\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}||}, g : E \to |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}| \times \mathbf{Connection}(\mathsf{I}^{\mathsf{Glue}}, \mathsf{I}^{\mathsf{Sys}}) \times |\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Comp}}}| \rangle$ such that:*

- *$p(v) \subseteq \left\{ \mathbf{Th}(\sigma) \;\middle|\; \sigma : \Sigma \to \mathbf{Sign}(f(v)) \in ||\mathsf{Sign}^{\mathsf{I}^{\mathsf{Comp}}}|| \right\}$, for all $v \in V$,*
- *let π_1, π_2 and π_3 are the first, second and third projections of a tuple, respectively, and* `dom` *retrieves the domain of a morphism, then for all $e \in E$,* $\mathtt{dom}(\pi_1(\pi_1(\pi_2(g(e))))) = \mathtt{dom}(\pi_1(g(e)))$ *and*

$\mathtt{dom}(\pi_3(\pi_3(\pi_2(g(e))))) = \mathtt{dom}(\pi_1(g(e)))$, *and*
– $\pi_1(g(e)) \in p(src(e))$ *and* $\pi_3(g(e)) \in p(trg(e))$, *for all* $e \in E$.

The intuition behind Def. 4 is that configurations are captured by graphs whose nodes are interpreted as components, and edges as tuples formed by ports and connections capable of enabling the communication.

Definition 5. *Let* $\mathsf{I}^{\mathsf{Sys}}$, $\mathsf{I}^{\mathsf{Comp}}$ *and* $\mathsf{I}^{\mathsf{Glue}}$ *be institutions, and* $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ *and* $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ *be institution representations. Let* $G = \langle V, E\rangle$ *be a graph, and* ι *a labelling for* G *according to Def. 4, then a* system design *is a structure of the form* $\langle G, \iota\rangle$.

As usual in the field of institutions, a specification of a system will be a diagram in the category of theories of a given institution, and the composition of the theories in the system specification will be the co-limit of such a diagram. This requires the category of theories to be finitely co-complete which, by [18, Thm. 11], follows directly when the category of signatures is finitely co-complete. In our case, the diagram is obtained by using the fact that the graph is expressed in terms of two institutions, $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$, for which there exists an institution $\mathsf{I}^{\mathsf{Sys}}$ and institution representations $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ and $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$, guaranteeing that a common interpretation is feasible.

The following theorem will be an important tool. Intuitively, this theorem tells us that whenever a connector (a co-span in $\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Glue}}}$) is translated from $\mathsf{I}^{\mathsf{Glue}}$ to $\mathsf{I}^{\mathsf{Sys}}$, using an institution representation, it yields a co-span in $\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Sys}}}$, thus complying with the restrictions associated with composition in the bicategory $\mathbf{co\text{-}span}(\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Sys}}})$.

Theorem 1. *Let* I *and* I' *be institutions such that* $\mathsf{Sign}^{\mathsf{I}}$ *and* $\mathsf{Sign}^{\mathsf{I}'}$ *are co-complete and have pushouts, and let* $\gamma : \mathsf{I} \to \mathsf{I}'$ *be an institution representation. Then, the pointwise extension of* $\gamma^{\mathsf{Th_0}^{\mathsf{I}}} : \mathsf{Th_0}^{\mathsf{I}} \to \mathsf{Th_0}^{\mathsf{I}'}$, $\widehat{\gamma^{\mathsf{Th_0}^{\mathsf{I}}}} : \boldsymbol{co\text{-}span}(\mathsf{Th_0}^{\mathsf{I}}) \to \boldsymbol{co\text{-}span}(\mathsf{Th_0}^{\mathsf{I}'})$, *is a lax functor.*

The following definition is based on the previous result, and enables us to integrate the computational parts of the glue and the adaptors in a communication mechanism. As the reader will notice, the connections, which are triples involving a connector and two adaptors, are translated into a single connector in the richer institution used to integrate components and connectors.

Definition 6. *Let* $\mathsf{I}^{\mathsf{Sys}}$, $\mathsf{I}^{\mathsf{Comp}}$ *and* $\mathsf{I}^{\mathsf{Glue}}$ *be institutions, and* $\gamma^{Comp} : \mathsf{I}^{\mathsf{Comp}} \to \mathsf{I}^{\mathsf{Sys}}$ *and* $\gamma^{Glue} : \mathsf{I}^{\mathsf{Glue}} \to \mathsf{I}^{\mathsf{Sys}}$ *be institution representations. Let* $G = \langle V, E\rangle$ *be a graph and* $\iota = \langle f, p, g\rangle$ *an interpretation for* G. *We define* $F(\langle G, \iota\rangle) = \langle \delta_0, \delta_1\rangle : G_\iota \to \mathbf{graph}(\mathsf{Th_0}^{\mathsf{I}^{\mathsf{Sys}}})$ *as follows:*

$$\begin{aligned}
G_\iota = \langle V \cup \textstyle\bigcup_{e\in E}\{r_e^1, g_e, r_e^2\}, \bigcup_{e\in E}\{e_1, e_1', e_2', e_2\}\rangle \text{ such that:}\\
src(e_1) = r_e^1 \text{ and } trg(e_1) = src(e),\\
src(e_1') = r_e^1 \text{ and } trg(e_1') = g_e,\\
src(e_2') = r_e^2 \text{ and } trg(e_2') = g_e, \text{ and}\\
src(e_2) = r_e^2 \text{ and } trg(e_2) = trg(e),
\end{aligned}$$

$$\delta_0(v) = \begin{cases} \gamma^{Comp^{\mathsf{Th}_0}}(f(v)) & , \textit{ if } v \in V. \\ \mathtt{dom}(\pi_1(\pi_1(\pi_2(g(e))))) & , \textit{ if } v = r_e^1. \\ \pi_2(\pi_1(\pi_2(g(e)))\ ;\widehat{\gamma^{Glue}(\pi_2(g(e)))};\ \pi_3(\pi_2(g(e)))) & , \textit{ if } v = g_e. \\ \mathtt{dom}(\pi_3(\pi_3(\pi_2(g(e))))) & , \textit{ if } v = r_e^2. \end{cases}$$

$$\delta_1(e) = \begin{cases} \gamma^{Comp^{\mathsf{Th}_0}}(\pi_1(g(e))) & , \textit{ if } src(e) = r_e^1 \textit{ and } trg(e) = a. \\ \pi_1(\pi_1(g(e))\ ;\widehat{\gamma^{Glue}(\pi_2(g(e)))};\ \pi_3(g(e))) & , \textit{ if } src(e) = r_e^1 \textit{ and } trg(e) = g_e. \\ \pi_3(\pi_1(g(e))\ ;\widehat{\gamma^{Glue}(\pi_2(g(e)))};\ \pi_3(g(e))) & , \textit{ if } src(e) = r_e^2 \textit{ and } trg(e) = g_e. \\ \gamma^{Comp^{\mathsf{Th}_0}}(\pi_3(g(e))) & , \textit{ if } src(e) = r_e^2 \textit{ and } trg(e) = b. \end{cases}$$

In order to make the previous construction clearer, we illustrate how this applies to our previously introduced example of the producer and consumer, when we interconnect the parts and form a system design.

Example 4. (Putting the producer and the consumer together in a synchronous way) Putting the Producer and Consumer component specifications together in a synchronous way requires just *coordinating* them. As put forward in [9], this can be achieved by indicating how attributes are "connected" or identified with attributes of other components, and by synchronising actions.

This is a straightforward way of connecting two components, which simply expresses a correlation between the symbols of the components. In our example, we may want to make the components interact by synchronising the `send-i` and `extract-i` actions, of the producer and consumer, respectively, and by identifying `ready-in` and `p-waiting`, in the producer, with `c-waiting` and `ready-ext` in the consumer, respectively. This situation requires the system design to be over a single institution, so components and glues are expressed in a common language, as theories in $\mathsf{I}^{\mathsf{LTL}}$. To make it clearer, $\Sigma =$ [**Attributes** : $\mathbf{x}, \mathbf{y}$: **Bool**; **Actions** : $\mathbf{a}, \mathbf{b}$], and $\gamma^{Comp^{\mathsf{Th}_0}} = \gamma^{Glue^{\mathsf{Th}_0}} = id_{\mathsf{Th}_0{}^{\mathsf{I}^{\mathsf{LTL}}}}$.

Putting together **Producer** and **Consumer** in a synchronous way can be done in a homogeneous setting. Of course, the machinery we have defined will actually demonstrate its potential when dealing with heterogeneous specifications. Example 5 generalises the previous one, in which the components need to be connected asynchronously, and the communication mechanism is specified in a formalism different from that used for components.

Example 5. (Putting the producer and the consumer together in an asynchronous way) Consider a more complex communicating scenario for the producer and the consumer, in which these components need to interact via an asynchronous communication channel. The idea is to maintain the specifications for producer and consumer, which have already been appropriately characterised, and model the asynchronous nature of the channel within the communication specification, i.e., in the connector. This cannot be captured simply by identification of symbols in the interconnected parts. We will assume the state of the glue is characterised just by a queue whose functional behaviour is described in equational logic (Fig. 3). Now, we put these specifications together, so that the producer

Component: **BitQueue**
Sorts: Queue
Ops:
empty : Queue,
isEmpty? : Queue $\to$ Bool,
enqueue : Queue $\times$ Bit $\to$ Queue,
dequeue : Queue $\to$ Queue,
front : $\{$ q : Queue $\mid \neg$*isEmpty?*(q) $\}$ $\to$ Bit.

Axioms: **vars** : q : Queue, b, b' : Bit
1. $isEmpty?(empty) = true$
2. $isEmpty?(enqueue(b, q)) = false$
3. $front(enqueue(b, empty)) = b$
4. $front(enqueue(b', enqueue(b, q))) = front(enqueue(b, q))$
5. $dequeue(enqueue(b, empty)) = empty$
6. $dequeue(enqueue(b, enqueue(b', q))) = enqueue(b, dequeue(enqueue(b', q)))$

Vars: q: Queue

Fig. 3. Producer and Consumer with Asynchronous communication

and the consumer communicate via a buffer of bit messages specified by the above queue. As opposed to the previous example, now we have a different formalism for the communication specification, i.e., I^{Glue} is **Eq** (equational logic), and the problem of putting together the three components cannot be syntactically solved.

We need to find an appropriate institution $\mathsf{I}^{\mathsf{Sys}}$, expressive enough to interpret, in a semantics preserving way, both linear temporal logic and equational logic. We will use first-order linear temporal logic [24]. The institution representation γ^{Comp} is the standard embedding of propositional temporal logic into first-order temporal logic. The institution representation γ^{Glue} is the embedding of equational logic into first-order logic with equality.

The reader should notice that, since the components and the glue are specified in different logics, we need suitable adaptors to put them together, which have to be specified in the richer institution $\mathsf{I}^{\mathsf{Sys}}$. Figs. 4 a) and 4 b) correspond to the adaptors in first-order LTL. Note that in the axioms q is a flexible variable, and q' is a rigid or logical (specification) variable. The reader should notice that even when the adaptors presented in Figs. 4 a) and 4 b) look complex in relation to the components being connected, they would remain the same, independently of the complexity of the components; this means that one could consider a more complex specification of the producer and the consumer, including the formalisation of the internal processes by which the information is produced and consumed, which could be highly complex. Objects originating ports and roles are the axiomless theories with signatures: 1. $\{send\text{-}i_{i=1,2}, ready\text{-}in, p\text{-}init\}$ for π_A, 2. $\{\text{Bool}, q, isEmpty? : \text{Queue} \to \text{Bool}, enqueue : \text{Queue} \times \text{Bit} \to \text{Queue}\}$ for ρ_l, 3. $\{extract\text{-}i_{i=1,2}, ready\text{-}ext, c\text{-}init\}$ for π_B, and 4. $\{\text{Bool}, q, isEmpty? : \text{Queue} \to \text{Bool}, dequeue : \text{Queue} \times \text{Bit} \to \text{Queue}, front : \text{Queue} \to \text{Bit}\}$ for ρ_r. The morphisms relating π_A, ρ_l, π_B and ρ_r with the corresponding theories associated with components, adaptors and glues, are inclusions in the corresponding category of signatures.

The reader should notice that the way in which the definitions and methodology we provided above interpret the elements of an architecture, allowed us to go from a model of a producer and a consumer connected in a synchronous way (see Ex. 4) to a model of a producer and a consumer connected in an asynchronous way

Component: **Adapt**
Sorts: Queue
Attributes: q: Queue
Actions: *p-init*, *send-0*, *send-1*, *ready-in*
Functions:
  enqueue : Queue $\times$ Bit $\rightarrow$ Queue,
  isEmpty? : Queue $\rightarrow$ Bool
Axioms:
1. $(\forall q' : \text{Queue}) q = q' \wedge \textit{send-0} \rightarrow \bigcirc(q = \textit{enqueue}(0, q'))$
2. $(\forall q' : \text{Queue}) q = q' \wedge \textit{send-1} \rightarrow \bigcirc(q = \textit{enqueue}(1, q'))$
3. $(\forall q' : \text{Queue}) q = q' \wedge \neg \textit{send-1} \rightarrow \neg \textit{send-0} \rightarrow \bigcirc(q = q')$
4. $\textit{p-init} \rightarrow \textit{isEmpty?}(q)$
5. $\textit{ready-in} \leftrightarrow \textit{true}$

a) A first-order LTL specification of **Adapt**.

Component: **Adapt'**
Sorts: Queue
Attributes: q: Queue
Actions: *c-init*, *extract-0*, *extract-1*, *ready-ext*
Functions:
  isEmpty? : Queue $\rightarrow$ Bool,
  dequeue : Queue $\rightarrow$ Queue,
  front : $\{ q : \text{Queue} \mid \neg \textit{isEmpty?}(q) \} \rightarrow$ Bit
Axioms:
1. $\textit{isEmpty?}(q) \rightarrow \neg \textit{extract-0} \wedge \neg \textit{extract-1}$
2. $(\forall q' : \textit{Queue})(q = q' \wedge \textit{extract-0}) \rightarrow (\textit{front}(q) = 0 \wedge \bigcirc(q' = \textit{dequeue}(q)))$
3. $(\forall q' : \textit{Queue})(q = q' \wedge \textit{extract-1}) \rightarrow (\textit{front}(q) = 1 \wedge \bigcirc(q' = \textit{dequeue}(q)))$
4. $(\forall q' : \textit{Queue})(q = q' \wedge \neg \textit{extract-1} \wedge \neg \textit{extract-0}) \rightarrow \bigcirc(q = q')$
5. $\textit{c-init} \rightarrow \textit{isEmpty?}(q)$
6. $\textit{ready-ext} \leftrightarrow \neg \textit{isEmpty?}(q)$

b) A first-order LTL specification of **Adapt**$'$.

Fig. 4. Specification of an adaptor.

(see Ex. 5), just by replacing the connection without modifying the components involved in the architectural design.

4 On the Institutions for Systems

In this section we introduce some results that allow us to obtain an institution of systems in a systematic way based on suitable specification languages for describing components and communications. A property that we want for such an institution is that both components and communications interpreted in the system institution can be mapped back to their original languages. This requirement emerges from the fact that it is often useful to be able to move back and forth from the (perhaps less expressive) specification languages used for components and communications to the formalism used to build the complete descriptions of the system. Moving from the component (resp. communication) specification language to the system specification language enables one to promote properties; moving from the system back to the components (resp. communications) allows us, for instance, to identify problems in the specifications of our "building blocks" when a counterexample of a property of the (whole) system is found.

Glueing two institutions together in a general way We provide a simple and general way of glueing two institutions into a new one. The motivation for doing so is, as we mentioned before, to help the specifier in the development of a suitable logic in which to express the system description, when one does not have in hand such a formalism.

Once $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ are fixed, it is possible to characterise an institution $\mathsf{I}^{\#}$ in which $\mathsf{I}^{\mathsf{Comp}}$ and $\mathsf{I}^{\mathsf{Glue}}$ can be put together. Furthermore, $\mathsf{I}^{\mathsf{Sys}}$ can be obtained by extending $\mathsf{I}^{\#}$ with additional logical structure depending on the properties required to be expressed. This must be done in such a way that there exists an institution representation $\iota_C : \mathsf{I}^{\mathsf{Comp}} \rightarrow \mathsf{I}^{\#}$, $\iota_G : \mathsf{I}^{\mathsf{Glue}} \rightarrow \mathsf{I}^{\#}$ and $\epsilon : \mathsf{I}^{\#} \rightarrow \mathsf{I}^{\mathsf{Sys}}$.

Let I^C and I^G be institutions. The following definition provides an institution constructed out of I^C and I^G. It is inspired by the construction presented by Sannella and Tarlecki in [29, Sec. 4.1.2], but it is slightly different. In [29, Ex. 4.1.44] and [29, Ex. 4.1.45], Sannella and Tarlecki provide the definitions of co-product and product of institutions, respectively. In the first case, as explained by the authors, the construction corresponds to putting two institutions together with no interaction; in the second case, the construction provides a way of putting them together but synchronising formulae by means of pairs. In our case, we need formulae to remain independent (requiring a co-product), but composite models to be pairs, each model coming from the corresponding institution (requiring a product). This need will become clear in Ex. 6 where we will extend the institution of the next definition with boolean operators combining formulae coming from any of the two logical systems, thus requiring models to give semantics to them.

Definition 7. $\mathsf{I}^{\#}(\mathsf{I}^{Comp}, \mathsf{I}^{Glue})$ *is defined as follows:*

– $\mathsf{Sign}^{\#} = \mathsf{Sign}^C \times \mathsf{Sign}^G$,
– $\mathbf{Sen}^{\#} = \left(\mathbf{Sen}^C \circ \pi_{left}\right) + \left(\mathbf{Sen}^G \circ \pi_{right}\right)$,
– $\mathbf{Mod}^{\#} = \left(\mathbf{Mod}^C \circ \pi_{left}\right) \times \left(\mathbf{Mod}^G \circ \pi_{right}\right)$,
– *Let* $\alpha \in \mathbf{Sen}^{\#}(\langle \Sigma^C, \Sigma^G \rangle)$ *and* $\langle \mathcal{M}^C, \mathcal{M}^G \rangle \in |\mathbf{Mod}^{\#}(\langle \Sigma^C, \Sigma^G \rangle)|$, *then we say that* $\langle \mathcal{M}^C, \mathcal{M}^G \rangle \models^{\#}_{\langle \Sigma^C, \Sigma^G \rangle} \alpha$ *if and only if: 1.* $\exists \alpha^C \in \mathbf{Sen}^C(\Sigma^C) | \alpha = in_{left}(\alpha^C) \wedge \mathcal{M}^C \models^{\mathsf{Comp}}_{\Sigma^C} \alpha^C$, *or 2.* $\exists \alpha^G \in \mathbf{Sen}^G(\Sigma^G) | \alpha = in_{right}(\alpha^G) \wedge \mathcal{M}^G \models^{\mathsf{Glue}}_{\Sigma^G} \alpha^G$.

Theorem 2. *Let* I^C *and* I^G *be institutions. Then,* $\mathsf{I}^{\#}(\mathsf{I}^C, \mathsf{I}^G)$ *is an institution.*

Property 2. Let I^C and I^G be institutions such that Sign^C and Sign^G are finitely co-complete. Then $\mathsf{Sign}^{\#}$ is finitely co-complete.

Definition 8. $\iota_C = \langle \gamma_C^{Sign}, \gamma_C^{Sen}, \gamma_C^{Mod} \rangle : \mathsf{I}^C \to \mathsf{I}^{\#}(\mathsf{I}^C, \mathsf{I}^G)$ *is defined as follows:*

– *for* $\Sigma \in |\mathsf{Sign}^C|$, $\gamma_C^{Sign}(\Sigma) = \langle \Sigma, \emptyset^G \rangle$, *where* $\emptyset^G$ *is the empty signature in* I^{G}, *and if* $\sigma \in ||\mathsf{Sign}^C||$, *then* $\gamma_C^{Sign}(\sigma) = \langle \sigma, id_{\emptyset^G} \rangle$,
– *for* $\Sigma \in |\mathsf{Sign}^C|$, *we define* $\gamma^{Sen}_{Comp\,\Sigma} : \mathbf{Sen}^{Comp}(\Sigma) \to \mathbf{Sen}^{\#} \circ \gamma_C^{Sign}(\Sigma)$, *as* ${\gamma_C^{Sen}}_{\Sigma} = in_{left}$, *and*
– *for* $\Sigma \in |\mathsf{Sign}^C|$, ${\gamma_C^{Mod}}_{\Sigma} : \mathbf{Mod}^{\#} \circ (\gamma_C^{Sign})^{\mathsf{op}}(\Sigma) \to \mathbf{Mod}^C(\Sigma)$, *is defined as* ${\gamma_C^{Mod}}_{\Sigma} = \pi_{left}$.

$\iota_G = \langle \gamma_G^{Sign}, \gamma_G^{Sen}, \gamma_G^{Mod} \rangle : \mathsf{I}^G \to \mathsf{I}^{\#}(\mathsf{I}^C, \mathsf{I}^G)$ *is defined in an analogous way.*

Theorem 3. *Let* I^C *and* I^G *be institutions. Then,* ι_C *and* ι_G *are institution representations.*

So far we have put together components and glues in a single language. However, it is obvious that we have not achieved any interaction between the languages as there is no actual "coordination" of their semantics. To deal with this issue, we can extend $\mathsf{I}^{\#}$ by adding logical behaviour that "coordinates" elements from I^{C} and I^{G}. The idea consists of extending $\mathsf{I}^{\#}(\mathsf{I}^{C}, \mathsf{I}^{G})$ to a new institution I^{Sys} where the additional logical behaviour is incorporated, but satisfying the following conditions: 1. $\mathsf{Sign}^{Sys} = \mathsf{Sign}^{\#}$, 2. for all $\Sigma \in |\mathsf{Sign}^{Sys}|$, $\mathbf{Sen}^{Sys}(\Sigma) \supseteq \mathbf{Sen}^{\#}(\Sigma)$, 3. for all $\Sigma \in |\mathsf{Sign}^{Sys}|$, $\mathbf{Mod}^{Sys}(\Sigma) = \mathbf{Mod}^{\#}(\Sigma)$, and 4. for all $\Sigma \in |\mathsf{Sign}^{Sys}|$, $\alpha \in \mathbf{Sen}^{\#}$ and $\mathcal{M} \in \mathbf{Mod}^{Sys}(\Sigma)$: $\mathcal{M} \models^{Sys}_{\Sigma} \alpha$ iff $\mathcal{M} \models^{\#}_{\Sigma} \alpha$.

Then, if $\mathsf{I}^{Sys} = \langle \mathsf{Sign}^{Sys}, \mathbf{Sen}^{Sys}, \mathbf{Mod}^{Sys}, \{\models^{Sys}_{\Sigma}\}_{\Sigma \in |\mathsf{Sign}^{Sys}|} \rangle$ is an institution, we have the guarantee that an institution representation ϵ exists simply by taking it to be the trivial inclusion institution representation, which of course satisfies the satisfaction invariance condition.

The following example shows how to extend an institution with boolean operators. Our construction, although very similar to the one presented by Sannella and Tarlecki in [29, Ex. 4.1.41], requires a slightly different treatment of formulae because their satisfaction, as we demonstrated before, must be evaluated in the corresponding model of the pair. Notice that as we are building composite formulae out of formulae coming from different institutions, the only way to assert their satisfaction by a model is by having a notion of model capable of interpreting every piece, thus justifying the need for a definition of institution whose formulae is the co-product of the sets of formulae of the two institutions and whose class of models is the product of the corresponding classes of models. Extending $\mathsf{I}^{\#}(\mathsf{I}^{C}, \mathsf{I}^{G})$ with boolean operators provides the most basic coordination of behavior by synchronising models through formulae they must be satisfy. More complex extensions can be made by choosing other logics to build on top of $\mathsf{I}^{\#}(\mathsf{I}^{C}, \mathsf{I}^{G})$; some of them also require a more complex class of models.

Example 6. Let I^{C} and I^{G} be institutions. Then, I^{Sys} is defined as the structure $\langle \mathsf{Sign}^{Sys}, \mathbf{Sen}^{Sys}, \mathbf{Mod}^{Sys}, \{\models^{Sys\,\Sigma}\}_{\Sigma \in |\mathsf{Sign}^{Sys}|} \rangle$ where:

- $\mathsf{Sign}^{Sys} = \mathsf{Sign}^{\#}$,
- for all $\Sigma^{C} \in |\mathsf{Sign}^{Comp}|$, $\Sigma^{G} \in |\mathsf{Sign}^{Glue}|$:
  - $in_{left}(\alpha) \in \mathbf{Sen}^{Sys}(\langle \Sigma^{C}, \Sigma^{G} \rangle)$, for all $\alpha \in \mathbf{Sen}^{C}(\Sigma^{C})$,
  - $in_{right}(\alpha) \in \mathbf{Sen}^{Sys}(\langle \Sigma^{C}, \Sigma^{G} \rangle)$, for all $\alpha \in \mathbf{Sen}^{G}(\Sigma^{G})$,
  - if $\alpha, \beta \in \mathbf{Sen}^{Sys}(\langle \Sigma^{C}, \Sigma^{G} \rangle)$, then $\{\neg\alpha, \alpha \vee \beta\} \in \mathbf{Sen}^{\mathsf{Sys}}(\langle \Sigma^{C}, \Sigma^{G} \rangle)$.
- $\mathbf{Mod}^{Sys} = \mathbf{Mod}^{\#}$, and
- for all $\langle \Sigma^{C}, \Sigma^{G} \rangle \in |\mathsf{Sign}^{Sys}|$, $\langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \in |\mathbf{Mod}^{\#}(\langle \Sigma^{C}, \Sigma^{G} \rangle)|$:

$$
\begin{aligned}
&\langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} in_{left}(\alpha^{C}) \text{ iff } \mathcal{M}^{C} \models^{C}_{\Sigma^{C}} \alpha^{C} \\
&\langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} in_{right}(\alpha^{G}) \text{ iff } \mathcal{M}^{G} \models^{C}_{\Sigma^{G}} \alpha^{G} \\
&\langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} \neg\alpha \text{ iff not } \langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} \alpha \\
&\langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} \alpha \vee \beta \text{ iff} \\
&\qquad \langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} \alpha \text{ or} \langle \mathcal{M}^{C}, \mathcal{M}^{G} \rangle \models^{\#}_{\langle \Sigma^{C}, \Sigma^{G} \rangle} \beta
\end{aligned}
$$

Proving that I^{Sys} is an institution is simple because $\mathsf{I}^{\#}$ is an institution and the boolean addition constitutes no problem in the proof. Equally simple is the proof that there exists an institution representation $\epsilon : \mathsf{I}^{\#} \to \mathsf{I}^{Sys}$.

Glueing two institutions in a known logic When one has in hand a logical system I formalised as an institution $\langle \mathsf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \{\models^{\Sigma}\}_{\Sigma \in |\mathsf{Sign}|} \rangle$ such that, given I^C and I^G as defined in the previous example, there exists institution representations $\gamma_C : \mathsf{I}^C \to \mathsf{I}$ and $\gamma_G : \mathsf{I}^G \to \mathsf{I}$, part of the problem is already solved. We however need a way of getting, from a whole system's specification, the parts that composed it in their original formalisms. The main technical difficulty at this point arises from the fact that symbols coming from components and glues may be identified as a single symbol in the system's language.

Definition 9. *Let* Sign^C *and* Sign^G *have pushouts of arbitrary co-spans and have initial objects* $\emptyset^C$ *and* $\emptyset^G$ *respectively; and suppose* $\gamma_C^{Sign}(\emptyset^C) = \gamma_G^{Sign}(\emptyset^G)$*. Then, we define* $\mathsf{I}^{Sys}(\mathsf{I})$ *in the following way:*

- $\mathsf{Sign}^{Sys} = \langle \mathcal{O}, \mathcal{A} \rangle$ *such that:*
  - $\mathcal{O} = \left\{ \langle \sigma_c : \Sigma^C \to \Sigma', \sigma_g : \Sigma^G \to \Sigma' \rangle \text{ pushout in } \mathsf{Sign} \right\}$,
  - $\mathcal{A} = \left\{ \left\langle \sigma_l : \Sigma^C \to \Sigma^{C'}, \sigma_s : \Sigma^S \to \Sigma^{S'}, \sigma_r : \Sigma^G \to \Sigma^{G'} \right\rangle \mid \langle \sigma_c, \sigma_g \rangle, \langle \sigma_c', \sigma_g' \rangle \in \mathcal{O} \text{ and } \sigma_c' \circ \sigma_l = \sigma_s \circ \sigma_c, \sigma_s \circ \sigma_g = \sigma_g' \circ \sigma_r \right\}$

  Identities and composition are defined component-wise.
- *for all* $\langle \sigma_c : \Sigma^C \to \Sigma', \sigma_g : \Sigma^G \to \Sigma' \rangle \in |\mathsf{Sign}^{Sys}|$ *and* $\langle \sigma_l : \Sigma^C \to \Sigma^{C'}, \sigma_s : \Sigma^S \to \Sigma^{S'}, \sigma_r : \Sigma^G \to \Sigma^{G'} \rangle \in ||\mathsf{Sign}^{Sys}||$*:* $\mathbf{Sen}^{Sys}(\langle \sigma_c, \sigma_g \rangle) = \mathbf{Sen}(\Sigma')$*,* $\mathbf{Sen}^{Sys}(\langle \sigma_l, \sigma_s, \sigma_r \rangle) = \mathbf{Sen}(\sigma_s)$*,*
- *for all* $\langle \sigma_c : \Sigma^C \to \Sigma', \sigma_g : \Sigma^G \to \Sigma' \rangle \in |\mathsf{Sign}^{Sys}|$ *and* $\langle \sigma_l : \Sigma^C \to \Sigma^{C'}, \sigma_s : \Sigma^S \to \Sigma^{S'}, \sigma_r : \Sigma^G \to \Sigma^{G'} \rangle \in ||\mathsf{Sign}^{Sys}||$*:* $\mathbf{Mod}^{Sys}(\langle \sigma_c, \sigma_g \rangle) = \mathbf{Mod}(\Sigma')$ *and* $\mathbf{Mod}^{Sys}(\langle \sigma_l, \sigma_s, \sigma_r \rangle) = \mathbf{Mod}(\sigma_s)$*,*
- *for all* $\langle \sigma_c : \Sigma^C \to \Sigma', \sigma_g : \Sigma^G \to \Sigma' \rangle \in |\mathsf{Sign}^{Sys}|$*,* $\alpha \in \mathbf{Sen}^{Sys}(\langle \sigma_c, \sigma_g \rangle)$ *and* $\mathcal{M} \in \mathbf{Mod}^{Sys}(\langle \sigma_c, \sigma_g \rangle)$*,* $\mathcal{M} \models^{Sys}_{\langle \sigma_c, \sigma_g \rangle} \alpha$ *iff* $\mathcal{M} \models_{\Sigma'} \alpha$*.*

Notice that the definition of $\mathsf{I}^{Sys}(\mathsf{I})$ only differs from I in the category of signatures. This is because having pushouts as signatures opens up the possibility of tracing back the source of the objects we are dealing with. In the case of sentences and models, we only consider the signature that is in the target of the morphisms constituting the pushout. This construction is particularly useful in the cases there is a need to identify both the common part of the partial descriptions of the system (the fraction of the description on which the synchronisation of the languages takes place), and the elements that correspond to only one of the descriptions[6].

Theorem 4. *Let* I *be an institution. Then,* $\mathsf{I}^{Sys}(\mathsf{I})$ *is an institution.*

Theorem 5. *Let* I *be an institution such that* Sign *has an initial object* $\emptyset^I$ *and pushouts for arbitrary co-spans. Then,* Sign^{Sys} *is finitely co-complete.*

[6] In [3] we use institution representations to give semantics to schema promotion in Z notation. There, whenever a manager for the whole system is constructed, the only way to prove the commutativity of the diagrams (see [3, Sec. 4.1]) is by preserving the information revealing the language from which each of the elements originates.

Definition 10. *Let* Sign^C *and* Sign^G *have pushouts of arbitrary co-spans and have initial objects* $\emptyset^C$ *and* $\emptyset^G$ *respectively, and* $\gamma_C^{Sign}(\emptyset^C) = \gamma_G^{Sign}(\emptyset^G)$*. Then, we define* $\iota_C = \langle \iota_C^{Sign}, \iota_C^{Sen}, \iota_C^{Mod} \rangle : \mathsf{I}^C \to \mathsf{I}^{Sys}(\mathsf{I})$ *as follows:*

- *if* $\Sigma \in |\mathsf{Sign}^C|$*, then* $\iota_C^{Sign}(\Sigma) = \langle \sigma^C, \sigma^G \rangle$ *such that* $\langle \sigma^C, \sigma^G \rangle$ *is the pushout of* $\langle \gamma_C^{Sign}(\emptyset^C \to \Sigma), \gamma_G^{Sign}(id_{\emptyset^G}) \rangle$*; if* $\sigma \in ||\mathsf{Sign}^C||$*, then* $\iota_C^{Sign}(\sigma) = \langle \gamma_C^{Sign}(\sigma), \gamma_C^{Sign}(\sigma), \gamma_G^{Sign}(id_{\emptyset^G}) \rangle$*,*
- *if* $\Sigma \in |\mathsf{Sign}^C|$*, then we define* ${\iota_C^{Sen}}_\Sigma : \mathbf{Sen}^C(\Sigma) \to \mathbf{Sen}^{Sys} \circ \iota_C^{Sign}(\Sigma)$*, as* ${\iota_C^{Sen}}_\Sigma = {\gamma_C^{Sen}}_\Sigma$*,*
- *we define* ${\iota_C^{Mod}}_\Sigma : \mathbf{Mod}^{Sys} \circ (\iota_C^{Sign})^{\mathsf{op}}(\Sigma) \to \mathbf{Mod}^C(\Sigma)$*, as* ${\iota_C^{Mod}}_\Sigma = {\gamma_C^{Mod}}_\Sigma$*.*

$\iota_G = \langle \gamma_G^{Sign}, \gamma_G^{Sen}, \gamma_G^{Mod} \rangle : \mathsf{I}^G \to \mathsf{I}^{Sys}(\mathsf{I})$ *is defined analogously.*

Theorem 6. *Let* Sign^C *and* Sign^C *have pushouts of arbitrary co-spans and have initial objects* $\emptyset^C$ *and* $\emptyset^G$ *respectively, and* $\gamma_C^{Sign}(\emptyset^C) = \gamma_G^{Sign}(\emptyset^G)$*. Then,* ι_C *and* ι_G *are institution representations.*

5 Conclusions and related work

We have presented an abstract and heterogeneous categorical characterisation of component-based systems. Our characterisation is logic/language independent, based on the categorical notion of *institution*. The heterogeneity of the approach is aimed at favouring a more genuine separation of concerns in the specification of components, and how these communicate. Our characterisation is based on the view that the different elements of a software architecture, such as components, connectors, roles, ports and adaptors, may be more faithfully specified in different formalisms, which have then to be put together into a setting in which one can reason about these parts and the whole system in a coordinated way. While institutions are used to abstractly capture specification formalisms, we employ *institution representations* to relate the different formalisms. In particular, we show how to build a *system institution*, in which the various parts of the specification can be represented as identifiable pieces of the overall specification, and we can reason about system properties by performing relevant formal analyses over it. Our contribution involves then a the formal characterisation of the conditions to combine formalisms in a heterogeneous setting, heavily relying on the notions of institution and institution representation.

Our work is related to various formalisms for the specification of component-based systems, in particular those seeking heterogeneity and abstraction. A main source of inspiration is the categorical approach put forward by Fiadeiro et al [9, 15, 11] in relation to the architecture description language *CommUnity* [15, 13]. *CommUnity* comprises a specific component-based design language, in which components and connectors are specified in a particular way. In our approach, components and connectors might be defined in any formalism, *CommUnity* being a particular case. *Acme* [17] seeks similar objectives; it is defined as an *interchange architecture description language*, a setting where different formalisms might be combined. However, Acme has no actual formal semantics, and the examples of translations from particular architecture description languages to Acme are defined in an *ad-hoc* manner, generally dealt with only at a syntactic level. Thus, questions such as the coherence of resulting Acme specifications, cannot be answered in Acme's context.

CASL [2] is an algebraic language for formal specification, which uses the notion of institutions to achieve a high degree of abstraction. Architectural specifications in CASL [1] are built by using basic relationships between modules like refinement or extension, i.e., the architectural structure of a system in terms of components and connections are not explicitly captured in CASL.

In [21] an heterogeneous approach for specifying service-oriented systems is presented. The basic idea is to use two different institutions to capture the different levels involved in a service-oriented system. One institution is used to specify the local behaviour of services, while the other institution is used as a global logic to describe the orchestration of services. The two levels are related via a co-morphism (or institution representation). Notice that the global logic is used for the description of the common behaviour of components, but it is not used for describing coordination mechanisms in an abstract way, as is done by glues in the present paper. Also notice that this approach is heterogeneous but not language independent. HETS [26, 27] is a framework for integrating different institutions to support heterogeneous specifications of systems. We share the interest in heterogeneous frameworks of institutions, but our focus is on formalising the notions from software architecture and the corresponding kinds of entities, namely components, connectors, roles, ports, and adaptors, to structure heterogeneous system specifications. Nevertheless, we are committed to the HETS view that whenever the design requires the use of different languages for describing components (reap. glues) Grothendieck institutions provide the most suitable tool to deal with that level of heterogeneity. In [28], Mossakowski and Tarlecki presented a framework meant to be a tool for heterogeneous software specification. This framework exploits the use of morphisms and co-morphisms between institutions in a coordinated way, not only allowing moving a specification to a more expressive language, but also to project a part of the system into a less expressive one. Differences and similarities between their framework and ours are essentially the same as mentioned in the comparison above with HETS.

References

1. M. Bidoit, D. Sannella and A. Tarlecki, *Architectural Specifications in CASL*, in Proc. of AMAST '98, LNCS, 1999.
2. T. Mossakowski, A. Haxthausen, D. Sannella, and A. Tarlecki. *CASL: The common algebraic specification language: Semantics and proof theory*, Computing and Informatics, vol. 22, 2003.
3. P.F. Castro, N. Aguirre, C.G. López Pombo, T.S.E. Maibaum: *A Categorical Approach to Structuring and Promoting Z Specifications*, in Proc. FACS, LNCS, 2012.
4. M. Cengarle, A. Knapp, A. Tarlecki and M. Wirsing, *A Heterogeneous Approach To UML Semantics*, in Proc. of Concurrency, Graphs and Models, LNCS, 2008.

5. R. Diaconescu and K. Futatsugi. *Logical foundations of CafeOBJ.* Theor. Comp. Sc. 285(2), 2002.
6. H. Ehrig and B. Mahr, *Fundamentals of Algebraic Specification 2*, Springer, 1990.
7. H. Ehrig, M. Große-Rhode and U. Wolter, *On the Role of Category Theory in the Area of Algebraic Specification*, in Proc. of COMPASS/ADT, LNCS, 1995.
8. E. Allen Emerson, *Temporal and modal logic*, Handbook of Theoretical Computer Science, Vol. B, Elsevier, 1990.
9. J. Fiadeiro and T. Maibaum, *Temporal Theories as Modularisation Units for Concurrent System Specification.* Formal Asp. of Comp., 4(3), Springer, 1992.
10. J. Fiadeiro and T. Maibaum, *Describing, Structuring and Implementing Objects*, In Proc. of the REX Workshop, LNCS, 1990.
11. J. Fiadeiro and M. Wermelinger, *A graph transformation approach to software architecture reconfiguration*, Sc. of Comp. Prog. 44(2), Elsevier, 2002.
12. J. Fiadeiro and V. Schmitt, *Structured Co-spans: An Algebra of Interaction Protocols*, In Proc. CALCO 2007, LNCS, 2007.
13. J. Fiadeiro, *Categories for Software Engineering*, Springer, 2004.
14. J. Fiadeiro and T. S. E. Maibaum. *A Mathematical Toolbox for the Software Architect*, in Proc. Workshop on Software Specification and Design, IEEE, 1995.
15. J. Fiadeiro and T. S. E. Maibaum. *Categorical Semantics of Parallel Program Design*, Sc. of Comp. Prog. 28, 1997.
16. D. Garlan, *Software Architecture: A Roadmap*, ACM, 2000.
17. D. Garlan, R. Monroe and D. Wile, *Acme: an architecture description interchange language*, in Proc. of CASCON 97, 1997.
18. J. Goguen and R. Burstall, *Institutions: Abstract Model Theory for Specification and Programming.* Journal of the ACM, 39(1), ACM, 1992.
19. J. Goguen and G. Rosu, *Institution Morphisms*, Formal Asp. of Comp., V.13, Springer, 2002.
20. D. Harel, D. Kozen and J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.
21. A. Knapp, G. Marczynski, M. Wirsing and A. Zawlocki, *A Heterogeneous Approach to Service-Oriented Systems Specification*, in Proc. of SAC 2010, ACM, 2010.
22. A. Lopes and J. Fiadeiro, *Superposition: composition vs refinement of non-deterministic, action-based systems*, Formal Asp. of Comp. 16(1), Springer, 2004.
23. S. McLane, *Categories for working mathematicians*, Springer, 1971.
24. Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*, Springer, 1991.
25. J. Meseguer, *General Logics*, in Logic Colloquium '87, North Holland, 1989.
26. Till Mossakowski, *Heterogeneous Theories and the Heterogeneous Tool Set*, in Semantic Interoperability and Integration, Dagstuhl Seminar Proc., 2005.
27. T. Mossakowski, C. Maeder and K. Luttich, *The Heterogeneous Tool Set, Hets*, in Proc. of TACAS 2007, LNCS, 2007.
28. T. Mossakowski and A. Tarlecki. *Heterogeneous Logical Environments for Distributed Specifications.* In Proc. of WADT 2009, LNCS, 2009.
29. D. Sannella and A. Tarlecki. *Foundations of Algebraic Specification and Formal Software Development.* Springer, 2012
30. A. Tarlecki, *Moving Between Logical Systems*, in Proc. of COMPASS, LNCS, 1995.
31. A. Tarlecki, *Toward Specifications for Reconfigurable Component Systems*, In Proc. of ICATPN 2007, LNCS, 2007.
32. A. Tarlecki, *Towards Heterogeneous Specifications*, in Frontiers of Combining Systems, Vol. 2, 2000.
33. M. Wermelinger and J. Fiadeiro, *A graph transformation approach to software architecture reconfiguration*, Sc. of Comp. Prog. 44(2), Elsevier, 2002.