



LaMI

Laboratoire de Méthodes Informatiques

A Logic for Mixed Specifications

Marc Aiguier, Fabrice Barbier and Pascal Poizat

email(s) : {aiguier,fbarbier,poizat}@lami.univ-evry.fr

<http://www.lami.univ-evry.fr/~{fbarbier,poizat}>

Rapport de Recherche n^o 73-2002

Mai 2002

CNRS – Université d'Evry Val d'Essonne
523, Place des Terrasses
F-91000 Evry France

Résumé

Separation of concerns or aspects is nowadays recognized as an important issue in software engineering, both at the programming and at the design/specification level. The goal of mixed specification languages is to take into account all - or at least several - aspects of systems. We found out from our experience that a lot of mixed specification languages do share most of their features. However, specific theories, such as the symbolic transition systems one, still have to be studied for several families of mixed specification languages. In this paper we propose a logic for mixed specifications that aims at providing a first proposal to give an abstract denotational semantics for mixed specification languages. This logic enables one to reason about a specification at a high level, abstracting away from implementation detail and without targetting a specific (often initial) model. We present the instantiation of this logic on the KORRIGAN formal language, a language devoted for the structured mixed specification of systems and components and briefly discuss its application to LOTOS. We also show how our logic can be seen as an institution, which has the benefits of enabling a refinement theory for mixed specifications.

keywords : Mixed Formal Specifications, Denotational Semantics, Logic, Korrigan

1 Introduction

In the last few years, the need for a separation of concerns with reference to different aspects¹ of systems to be specified appeared at both the programming [24] and the specification level. Some works have tried to define a general framework for the integration or combination of these aspects within a global system or a global development process. However, we think that they very often talk of different aspects of systems in terms of different (yet defined within a single formalism) partial specifications to be integrated. We rather call this point of view integration. It is a very important issue but, nevertheless, we think that aspects are orthogonal (ie very different) properties of systems such as one may find in object-oriented methods : functional, dynamic, distributed, ... We call systems mixed when they have more than one aspect.

Following previous works [18, 4], we focus on the specification of static (ie datatypes), dynamic (ie behaviours, communications, synchronisation, value passing, ...) and compositional (ie specifying a complex component from simpler ones) aspects. Two different approaches may be used to specify mixed systems [29]. The homogeneous approach uses a single formalism (often extending it in a syntactical way) to specify both aspects [8, 5, 25, 9]. The heterogeneous approach uses different formalisms [20, 6, 10, 19, 34, 1]. We think this last approach is well adapted to the mixed specification as it enables one to use the more adapted formalism for each aspect - adapted eventually meaning the formalism one knows well or the formalism of a given component one wishes to reuse. Therefore this is the approach we follow. However, we also think that the

¹Also called views, facets, paradigms or perspectives in the litterature.

homogeneous approach is better suited (ie simpler) to the verification part of the specification process : no integration problems arise as the verification may be done in a unique framework.

In the heterogeneous approach, we found out that the combination of algebraic specification with process algebras (PSF [27] or LOTOS[6]) or very abstract transition systems (SDL [19], proposals over Statecharts [34]) was a promising approach. Therefore, we proposed KORRIGAN [29], a formalism for the structured specification of mixed systems using algebraic datatypes, temporal logic and Symbolic Transition Systems (STS). Our STS originate from Graphical Abstract data Types (GAT) [3] but we recently saw that they could also be related to STG [22]. KORRIGAN is equipped with a methodology (that may be applied to any mixed specification language) that goes from the analysis of the requirement to the generation of concurrent object-oriented code [31]. This language as also been used in a component oriented approach [15].

We proposed an operational semantics for KORRIGAN based on global-states-and-transitions STSs in [14]. However this semantics was restricted to the integration of different aspects specifications - static and dynamic - and did not deal with the composition of specification and the explosion problems that may arise from the combination of a big or non fixed number of components.

We think that this results from the fact that we focused specifically on a particular model of the specification : the initial model described by an STS (or in the case of KORRIGAN, quotiented STS related by abstraction morphisms). If focusing on particular models (such as the initial model) may be of great interest when dealing with animation, bisimulation, model-checking or generating object-oriented code, it is somewhat less efficient, less clear and less concise than abstracting away from operational details, especially when dealing with structuring or big (ie real-size) specifications. Hence, one may want to have a general logic framework for mixed specifications at his disposal to define models in a loose (ie very abstract) way, and therefore to have the whole benefit of such an approach (structuring, abstraction, and refinement theories).

Moreover, we found out from our experience [29, 2] that mixed specification languages do share a lot of features, both in their syntax and their semantics. As far as semantics are concerned, the important issue is to stay symbolic, ie avoid the state explosion problem that arises either when one has value-passing events (the domain of the exchanged values may be infinite) or when components encapsulate some non finite datatype (e.g. a buffer may contain any number of elements). This problem has been studied in the value passing process algebras framework, leading to the definition of Symbolic Transition Graphs, symbolic bisimulation and symbolic modal logics [22, 33, 26]. More recently, the previous works had to be extended to deal with LOTOS specificities (mainly multiway synchronizing) [13, 11, 12].

We think that, rather than building abstract and efficient models in a bottom-up fashion, we could try to build them in a top-down approach. This means that, in parallel with the activity that studies formal languages equipped with an operationnal semantics and developping a theory on symbolic models for such languages (ie mainly bisimulation and model-checking on STS), we advocate

again for the definition of a general logical framework for mixed systems (based on a static part and any dynamic part defined as states and evolutions), and thereafter the exhibition of characteristics under which specific formalisms based on this logic do have a specific model (or several quotiented specific models) and may then be described using STS related within the logic by abstraction morphisms. The study of specific symbolic properties, symbolic bisimulation or symbolic model-checking could then be studied within this more general logical framework. In the meantime, we will have a very general denotational model which may be used to give denotational semantics to the existing mixed specification languages and to define new specific languages.

In this paper we present such a logic and its formal denotational semantics.

The paper is structured as follows. Section 2 presents KORRIGAN which will be used as a running example of instantiation of our logic in the sequel. Section 3 presents the single-component part of the logic and its denotational semantics. Then, in Section 4, we extend the logic to the specification of components and compositions of components. Section 5 shortly presents how mixed specifications languages such as LOTOS may be related to our logic. In Section 6 we propose an institution for our logic and present some fundamental results on refinement this institution enables. To end, Section 7 concludes and gives some perspectives.

2 Korrigan

KORRIGAN is a language devoted to the formal and structured specification of mixed systems. Its aim is to fill the gap between formal specification and software engineering. Hence it is equipped with a methodology and translation mechanisms [31], object-oriented code generation [17] and has been used in component [15] and object oriented [32] approaches. KORRIGAN is therefore inherently focused at a specific operational (initial) model. One may want to be able to think about a specification at a higher (ie more abstract, loose) level.

Here, we present such a mapping for the KORRIGAN language. This enables us both to give a formal denotational semantics for the whole language and to equip our model with a corresponding formal language that has nice features (textual and graphical notations [16], structuring and ability to build component-oriented specifications [15]).

KORRIGAN is based upon the structured specification of communicating components (with identifiers) by means of structures that we call *views* (Fig. 1).

Views are used to describe in a structured and unifying way the different aspects of a component using “internal” and “external” structuring. We define an *Internal Structuring View* abstraction that expresses the fact that, in order to design a component, it is useful to be able to express it under its different aspects (here the static and dynamic aspects, with no exclusion of further aspects that may be identified later on). Another structuring level is achieved through the *External Structuring View* abstraction, expressing that a component may be composed of several subcomponents. Such a composite component may be either

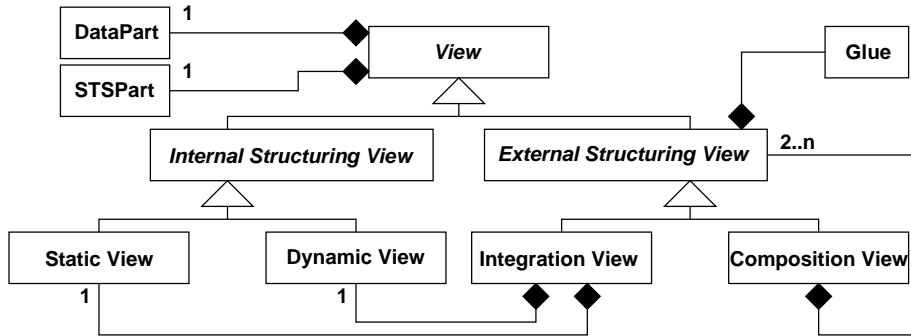


FIG. 1 – Views class hierarchy (UML notation).

an integration component (integrating different internal structuring views in an *Integration View*), or a concurrent component (*Composition View*). Integration views follow an *encapsulation principle*: the static aspect (*Static View*) may only be accessed through the dynamic aspect (*Dynamic View*) and its identifier.

3 Basic Mixed Specifications

3.1 Data part

The data part addresses the functional issues of components (definition of the operation values). It will be described with a many-sorted first order logic. As usual, *terms* and *formulas* are inductively built over a *many-sorted first order signature*, noted $\Sigma = (S, \mathcal{F}, R)$, and a set of *many-sorted variables*, noted $V = (V_s)_{s \in S}$. The signature $\Sigma = (S, \mathcal{F}, R)$ is defined as follows: S is a set of *sorts*, \mathcal{F} is a set of *function names*, each one equipped with a *profile* of form $\omega \rightarrow s$ ($\omega \in S^*$ and $s \in S$), and R is a set of *predicate names*, each one equipped with a *profile* of form $\omega \in S^*$. The set of variables $V = (V_s)_{s \in S}$ is a S -indexed family of sets $V = (V_s)_{s \in S}$ such that for every $s \in S$, $V_s \cap \mathcal{F} = \emptyset$. For every $s \in S$, $T_\Sigma(V)_s$ denotes the standard set of *terms of sort s over Σ* , free with generating set V . We will note $T_\Sigma(V)$ the set of all Σ -terms, $T_\Sigma(V) = (T_\Sigma(V)_s)_{s \in S}$. Given a Σ -term $t \in T_\Sigma(V)$, let us note $\mathcal{V}ar(t)$ the set of variables occurring in t . From Σ -terms, formulas are defined as follows: *atomic formulas* are sentences of form $r(t_1, \dots, t_n)$, where $(t_1, \dots, t_n) \in T_\Sigma(V)_{s_1} \times \dots \times T_\Sigma(V)_{s_n}$ and $r \in R$ is of profile $s_1 \dots s_n$. From the set of atomic formulas, Σ -formulas are built, in accordance with the following conditions: every atomic formula is a Σ -formula and if φ and ψ are Σ -formula, $@ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ and $Q \in \{\forall, \exists\}$, then so are $\varphi @ \psi$, $Qx\varphi$ and $\neg\varphi$.

The mathematical interpretation of any signature $\Sigma = (S, \mathcal{F}, R)$ is given by a S -set $M = (M_s)_{s \in S}$ provided with a total function $f^{\mathcal{M}} : M_{s_1} \times \dots \times M_{s_n} \rightarrow M_s$ for each function name $f : s_1 \dots s_n \rightarrow s \in \mathcal{F}$ and a n -ary relation $r^{\mathcal{M}} : M_{s_1} \times$

$\dots \times M^{s_n}$ for each predicate name $r : s_1 \dots s_n \in R$. The evaluation of Σ -terms from a Σ -model \mathcal{M} is given by any totale function $\sigma^\# : T_\Sigma(V) \rightarrow M$ defined as the canonical extension of any interpretation of variables $\sigma : V \rightarrow M$. Therefore, we extend any interpretation σ into an unary relation $\mathcal{M} \models_\sigma$ on Σ -formulas as usual. The validation of Σ -formulas from Σ -models is defined by : $\mathcal{M} \models \varphi$ if and only if for any $\sigma : V \rightarrow M$, $\mathcal{M} \models_\sigma \varphi$.

3.2 Behaviour part : syntax

This part addresses the behavioural and communication issues of components (when and under which conditions do some event may take place). As usual with first order logics, the syntax is given by a signature from which we inductively define first syntactical elements which are terms, and then extend them into formulas to specify the expected properties of systems.

Exchange profiles model profiles (ie typing information) of components with who the component taken into account will communicate.

Definition 1 (Exchange profiles) *Let δS be a set of sorts. The set of exchange profile over δS , noted $E_{\delta S}$, is the least set (according to the theoretical set inclusion) inductively defined as follows :*

- $\forall \delta s \in \delta S, \uparrow \delta s \in E_{\delta S}$,
- $\forall \delta s \in \delta S, \downarrow \delta s \in E_{\delta S}$,
- $\forall (\eta_1, \eta_2) \in E_{\delta S} \times E_{\delta S}, \eta_1 \eta_2 \in E_{\delta S}$.

Dynamic profiles then take into account both exchange profiles and the typing of data exchanges (as usual in value passing process algebras such as LOTOS).

Definition 2 (Dynamic profiles) *Let S be a set of sorts and let δS be a non-empty subset of S . The set of dynamic profile over S and δS , noted $P_{S, \delta S}$, is the least set (according to the theoretical set inclusion) inductively defined as follows :*

- $\varepsilon \in P_{S, \delta S}$,
- $\forall s \in S, \forall \eta \in E_{\delta S}, ?s\eta \in P_{S, \delta S}$,
- $\forall s \in S, \forall \eta \in E_{\delta S}, !s\eta \in P_{S, \delta S}$,
- $\forall (\rho_1, \rho_2) \in P_{S, \delta S} \times P_{S, \delta S}, \rho_1 \rho_2 \in P_{S, \delta S}$.

Remark 1 *In the above definition, ‘ η ’ can sometimes denote the empty sequence of exchange profiles.*

Dynamic signatures correspond in the behaviour part to the (more usual) signatures in the data part. They relate event names with their profiles.

Definition 3 (Dynamic signatures) *Let $\Sigma = (S, \mathcal{F}, R)$ be a many-sorted first-order signature. A dynamic signature $\delta\Sigma$ over Σ is a couple $((\delta S, \delta s), \mathfrak{E}\mathfrak{v})$ where :*

- $(\delta S, \delta s)$ is a pointed set such that $\delta S \subseteq S$. δs is called the sort of interest of $\delta \Sigma$,
- $\mathfrak{E}\mathfrak{v}$ is a set of event names, each one equipped with a profile $\rho \in P_{S, \delta s}$.
Moreover, $\mathfrak{E}\mathfrak{v} \cap (\mathcal{F} \cup R) = \emptyset$.

Let us note $\mathfrak{E}\mathfrak{v}^* = \mathfrak{E}\mathfrak{v} \cup \{\varkappa\}$ where \varkappa is a special event, called empty event, equipped with the empty dynamic profile ε and which does not belong to $\mathcal{F} \cup R$.

We now address the issue of defining some kind of *dynamic terms* in the same way static terms may be defined from static signatures.

Exchange offers (resp. offers) will correspond to exchange profiles (resp. dynamic profiles). Such features take their inspiration from KORRIGAN, LOTOS and SDL amongst others.

Definition 4 (Exchange offers) Let $\delta \Sigma = (\delta S, \mathfrak{E}\mathfrak{v})$ be a dynamic signature over a signature Σ , and let V be a set of variables over Σ . For any $\eta \in E_{\delta s}$, let us note $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_\eta$, the least set (according to the theoretical set inclusion) inductively defined on the structure of η as follows :

- $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_{\downarrow \delta s} = \{\downarrow x/x \in V_{\delta s}\}$,
- $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_{\uparrow \delta s} = \{\uparrow t/t \in T_\Sigma(V)_{\delta s}\}$,
- $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_{\eta_1 \cdot \eta_2} = \{\varpi_1 \varpi_2 / \varpi_1 \in \mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_{\eta_1} \wedge \varpi_2 \in \mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_{\eta_2}\}$.

Let us note $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V) = (\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_\eta)_{\eta \in E_{\delta s}}$. The elements in $\mathcal{E}\mathcal{O}_{\delta \Sigma}(V)$ are called exchange offers.

Definition 5 (Offers) With all the notations of Definition 4, for any $\rho \in P_{S, \delta s}$, let us note $\mathcal{O}_{\delta \Sigma}(V)_\rho$, the least set (according to the theoretical set inclusion) inductively defined on the structure of ρ as follows :

- $\mathcal{O}_{\delta \Sigma}(V)_{?s\eta} = \{?x\varpi/x \in V_s \wedge \varpi \in \mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_\eta\}$,
- $\mathcal{O}_{\delta \Sigma}(V)_{!s\eta} = \{!t\varpi/t \in T_\Sigma(V) \wedge \varpi \in \mathcal{E}\mathcal{O}_{\delta \Sigma}(V)_\eta\}$,
- $\mathcal{O}_{\delta \Sigma}(V)_{\rho_1 \cdot \rho_2} = \{\omega_1 \cdot \omega_2 / \omega_1 \in \mathcal{O}_{\delta \Sigma}(V)_{\rho_1} \wedge \omega_2 \in \mathcal{O}_{\delta \Sigma}(V)_{\rho_2}\}$.

Let us note $\mathcal{O}_{\delta \Sigma}(V) = (\mathcal{O}_{\delta \Sigma}(V)_\rho)_{\rho \in P_{S, \delta s}}$. The elements in $\mathcal{O}_{\delta \Sigma}(V)$ are called offers.

From the set $\mathcal{O}_{\delta \Sigma}(V)$ of offers, we can build the set of transition terms that will denote the transitions.

Definition 6 (Transition terms) Given a dynamic signature $\delta \Sigma$ over a signature Σ , let us note $\mathcal{T}_{\delta \Sigma}(V)$ the least set (according to the theoretical set inclusion) inductively defined as follows :

- if $e : \rho \in \mathfrak{E}\mathfrak{v}^*$ and $\omega \in \mathcal{O}_{\delta \Sigma}(V)_\rho$ then $e \cdot \omega \in \mathcal{T}_{\delta \Sigma}(V)$,
- **true** $\in \mathcal{T}_{\delta \Sigma}(V)$,
- if $\tau_1, \tau_2 \in \mathcal{T}_{\delta \Sigma}(V)$ then $\tau_1 @ \tau_2$, with $@ \in \{\Rightarrow, \wedge, \vee\}$,
- if $\tau \in \mathcal{T}_{\delta \Sigma}(V)$ then $\neg \tau \in \mathcal{T}_{\delta \Sigma}(V)$,
- if $x \in V$ and $\tau \in \mathcal{T}_{\delta \Sigma}(V)$ then $Qx\tau \in \mathcal{T}_{\delta \Sigma}(V)$, with $Q \in \{\forall, \exists\}$.

After dynamic signatures and dynamic terms, to complete the syntactical part of our logic for the behavioural aspects, we now have to define how dynamic properties may be expressed, that is dynamic formulas.

VIEW T	
SPECIFICATION	ABSTRACTION
imports A' generic on G variables V hides \bar{A} ops Σ axioms Ax	conditions C limit conditions Cl with Φ initially Φ_0 <hr style="width: 80%; margin: 5px auto;"/> OPERATIONS <hr style="width: 80%; margin: 5px auto;"/> O_i pre: P post: Q

FIG. 2 – Korrigan syntax (Views).

Definition 7 (Dynamic formulas) *With all the notations of Definition 6, a dynamic formula on $\delta\Sigma$ is any well-formed formula built on :*

- Σ -formulas,
- **true**,
- $[\tau]\varphi$, where $\tau \in \mathcal{T}_{\delta\Sigma}(V)$ and φ is a dynamic formula,
- $\langle\tau\rangle\varphi$, where $\tau \in \mathcal{T}_{\delta\Sigma}(V)$ and φ is a dynamic formula,
- propositional connectives in $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ and first-order quantifiers in $\{\forall, \exists\}$.

Let us note $\delta Sen(\delta\Sigma)$ the whole set of dynamic formulas.

Signatures of components and their properties may then be integrated within a dynamic specification : a view.

Definition 8 (View) *Given a many-sorted first order signature $\Sigma = (S, \mathcal{F}, R)$, a Σ -view \mathcal{V} is a 4-tuple $(\delta\Sigma, \mathbf{Ax}, \mathbf{State}, \mathbf{Init})$, where :*

- $\delta\Sigma$ is a dynamic signature over Σ ,
- $\mathbf{Ax} \subseteq Sen(\Sigma)$,
- $\mathbf{State} \cup \mathbf{Init} \subseteq \delta Sen(\delta\Sigma)$.

3.3 Example : Korrigan Internal Views

These KORRIGAN views model the different sequential aspects of components in a unifying way (Figure 2).

A set of conditions (C, Cl) defines an abstract point of view for components. Pre and postconditions of operations are then defined using these conditions. An STS, *i.e. Symbolic Transition System* may be built in a unique way from this description [30].

The denotation of a view in KORRIGAN (with the notations of Fig. 2) is a Σ -view $(\delta\Sigma, \mathbf{Ax}, \mathbf{State}, \mathbf{Init})$, where :

- \mathbf{Ax} corresponds to Ax ,

- **State** corresponds to Φ ,
- **Init** corresponds to Φ_0 ,
- $\delta\Sigma$ is such that :
 - $\delta s = \{T\}$ (type of interest),
 - $A' \subset S$
 - the set of variables contains *self* (of sort T) and one variable for each member of C and Cl ,
 - the static signature Σ corresponds to the SPECIFICATION part,
 - there is a correspondance between profiles of operations in Σ and $\mathfrak{E}\mathfrak{v}$ [14]
 - genericity is treated as usual in algebraic specifications [28],
 - the relation on $\mathfrak{E}\mathfrak{v}$ satisfies the relations on values of conditions (C , Cl) defined by the pre and postconditions of the event and the possible values of *self* in σ' corresponds to the application of the operation corresponding to the event to the value of *self* in σ and any possible receptions,
 - the preorder is restricted to a transition relation.

Whenever the view specification is given in its STS form (directly or by transformation), the denotation corresponds trivially to the denotation given above with exception that the set of variables contains *self* and one variable V_s per states. **Init** and **State** are then defined respectively as $\bigwedge_i V_i^0 \bigwedge_j \neg V_j^1$ and by the exclusive disjunction (xor) on every V_s (with V^0 being the subset of V related with initial states, and V^1 the subset of V related to the other states). The relation on $\mathfrak{E}\mathfrak{v}$ is directly obtained from the transition relation.

KORRIGAN builds in fact an abstraction over our denotational model using the conditions (C , Cl) and the pre and postconditions. This abstraction corresponds to an STS where states are quotiented over *self* and transitions over the received variables.

3.4 Denotational Semantics

Definition 9 (Read Variables (\mathcal{RV}) and Sent Terms (ST)) *Let $\delta\Sigma$ be a dynamic signature over a signature Σ . Let V be a set of variables on Σ . Let $\omega \in \mathcal{O}_{\delta\Sigma}(V)$. Let us note $\mathcal{RV}(\omega)$ (resp. $ST(\omega)$) the whole set of variables x (resp. terms t) such that $?x$ or $\downarrow x$ (resp. $!t$ or $\uparrow t$) occurs in ω .*

Notation 1 *Let S be a set of sorts. Let A and B be two S -sets. Let us note B^A the set of partial functions from A to B compatible with S (i.e. $\forall \nu \in B^A, \forall s \in S, \forall a \in A_s, \nu(a)$ is defined $\implies \nu(a) \in B_s$). In the following, we will note $\nu(a) \searrow$ (resp. $\nu(a) \nearrow$) to mean that $\nu(a)$ is defined (resp. undefined).*

Definition 10 (Dynamic model) *Let $\delta\Sigma = (\delta S, \mathfrak{E}\mathfrak{v})$ be a dynamic signature over a many-sorted first order signature Σ . Let V be a set of many-sorted variables. A dynamic model \mathcal{D} for $\delta\Sigma$, also called a $\delta\Sigma$ -model, is a Σ -model \mathcal{M} equipped for every $e \in \mathfrak{E}\mathfrak{v}$ with a binary relation $e^{\mathcal{D}}$ on M^V .*

Definition 11 (Evaluation of transition terms) Let $\delta\Sigma$ be a dynamic signature over a signature Σ . Let V be a set of variables on Σ . Let \mathcal{D} be a $\delta\Sigma$ -model. Let $\tau \in \mathcal{T}_{\delta\Sigma}V$. The evaluation of τ on \mathcal{D} , noted $\llbracket \tau \rrbracket_{\mathcal{D}}$, is the binary relation on M^V inductively defined on the structure of τ as follows :

$$(\sigma, \sigma') \in \llbracket e.\omega \rrbracket_{\mathcal{D}} \iff \begin{cases} (\sigma, \sigma') \in e^{\mathcal{D}} \\ \wedge (\forall t \in \mathcal{ST}(\omega), \sigma^\#(t) \searrow) \\ \wedge (\forall x \in \mathcal{RV}(\omega), \sigma'(x) \searrow) \\ \wedge (\forall x \in V \setminus \mathcal{RV}(\omega), \sigma(x) \searrow \Rightarrow \sigma'(x) \searrow \wedge \sigma'(x) = \sigma(x)) \end{cases}$$

$$\llbracket \mathbf{true} \rrbracket_{\mathcal{D}} = \bigcup_{e \in \mathcal{E}\mathfrak{b}} e^{\mathcal{D}}$$

$$\llbracket \tau_1 \wedge \tau_2 \rrbracket_{\mathcal{D}} = \llbracket \tau_1 \rrbracket_{\mathcal{D}} \cap \llbracket \tau_2 \rrbracket_{\mathcal{D}}$$

$$\llbracket \tau_1 \vee \tau_2 \rrbracket_{\mathcal{D}} = \llbracket \tau_1 \rrbracket_{\mathcal{D}} \cup \llbracket \tau_2 \rrbracket_{\mathcal{D}}$$

$$\llbracket \tau_1 \Rightarrow \tau_2 \rrbracket_{\mathcal{D}} = \llbracket \neg \tau_1 \rrbracket_{\mathcal{D}} \cup \llbracket \tau_2 \rrbracket_{\mathcal{D}}$$

$$\llbracket \neg \tau \rrbracket_{\mathcal{D}} = \llbracket \mathbf{true} \rrbracket_{\mathcal{D}} \setminus \llbracket \tau \rrbracket_{\mathcal{D}}$$

$$\llbracket \forall x \tau \rrbracket_{\mathcal{D}} = \begin{cases} \llbracket \tau \rrbracket_{\mathcal{D}} & \text{if } \forall v \in M, \exists \sigma' \in M^V, (\sigma, \sigma') \in \llbracket \tau \rrbracket_{\mathcal{D}} \wedge \sigma'(x) = v \\ \emptyset & \text{otherwise} \end{cases}$$

$$\llbracket \exists x \tau \rrbracket_{\mathcal{D}} = \begin{cases} \llbracket \tau \rrbracket_{\mathcal{D}} & \text{if } \exists v \in M, \exists \sigma' \in M^V, (\sigma, \sigma') \in \llbracket \tau \rrbracket_{\mathcal{D}} \wedge \sigma'(x) = v \\ \emptyset & \text{otherwise} \end{cases}$$

Definition 12 (Satisfaction of dynamic formulas) Let $\delta\Sigma$ be a dynamic signature over a signature Σ . Let V be a set of variables over Σ . Let $\varphi \in \delta\text{Sen}(\delta\Sigma)$. Let \mathcal{D} be a $\delta\Sigma$ -model. \mathcal{D} satisfies φ on a state $\sigma \in M^V$, noted $\mathcal{D} \models_{\sigma} \varphi$, is inductively defined on the structure of φ as follows :

- if φ is a Σ -formula, then $\mathcal{D} \models_{\sigma} \varphi$ if and only if, if σ is defined on all free variables of φ then $\mathcal{M} \models_{\sigma} \varphi$,
- $\forall \sigma \in M^V$, $\mathcal{D} \models_{\sigma} \mathbf{true}$,
- if φ is of the form $[\tau]\psi$ then $\mathcal{D} \models_{\sigma} [\tau]\psi$ if and only if for all $\sigma' \in M^V$ such that $(\sigma, \sigma') \in \llbracket \tau \rrbracket_{\mathcal{D}}$, $\mathcal{D} \models_{\sigma'} \psi$,
- if φ is of the form $\langle \tau \rangle \psi$ then $\mathcal{D} \models_{\sigma} \langle \tau \rangle \psi$ if and only if there exists $\sigma' \in M^V$ such that $(\sigma, \sigma') \in \llbracket \tau \rrbracket_{\mathcal{D}}$, $\mathcal{D} \models_{\sigma'} \psi$,
- propositional connectives and first-order quantifiers are handled as usual.

As usual, $\mathcal{D} \models \varphi$ is defined as $\forall \sigma \in M^V$, $\mathcal{D} \models_{\sigma} \varphi$.

Definition 13 (View model) Let $\mathcal{V} = (\delta\Sigma, \mathbf{Ax}, \mathbf{State}, \mathbf{Init})$ be a view. A view model for \mathcal{V} is a pair $(\mathcal{D}, <^{\mathcal{D}})$ where \mathcal{D} is a $\delta\Sigma$ -model and $<^{\mathcal{D}}$ is a pre-order on M^V such that the following conditions hold :

- $\forall(\sigma, \sigma') \in e^{\mathcal{D}}, \sigma <^{\mathcal{D}} \sigma'$,
- $\mathcal{M} \models \mathbf{Ax}$,
- $\mathcal{D} \models \mathbf{State}$,
- $\forall\sigma \in M^V, \mathcal{D} \models_{\sigma} \mathbf{Init} \implies \forall\sigma' \in M^V, \sigma <^{\mathcal{D}} \sigma'$.

The interest of using a pre-order is to be able to define later on a refinement relation on dynamic behaviours. Moreover, $(\mathcal{D}, <^{\mathcal{D}})$ can be seen as a Kripke structure where states are valuation of variables (M^V) and where the reachability relation is naturally denoted by the $<^{\mathcal{D}}$ pre-order which contains the execution of events.

4 Composed Mixed Specifications

This part addresses the composition, architectural, synchronizing and inter-component communication aspects of systems.

4.1 Composition : syntax

As for the basic mixed specifications, we here define first composed signatures, then composed formulas and finally composed specifications. Here we have no need for any composed equivalent of terms.

Notation 2 Given a view $\mathcal{V} = ((\delta\Sigma, \delta s), \mathbf{Ax}, \mathbf{State}, \mathbf{Init})$ with $\delta\Sigma = (\delta S, \mathfrak{Cv})$ and $\Sigma = (S, \mathcal{F}, R)$, we will use the following notations :

- $\delta\text{Sig}[\mathcal{V}] = \delta\Sigma$ - $\text{Rel}[\text{Sig}[\mathcal{V}]] = R$
- $\delta\text{Sort}[\mathcal{V}] = \delta S$ - $\text{Fun}[\text{Sig}[\mathcal{V}]] = \mathcal{F}$
- $\text{Sig}[\mathcal{V}] = \Sigma$ - $\delta\text{int}[\mathcal{V}] = \delta s$

Definition 14 (Composed mixed signature) A composed mixed signature $\tilde{\Sigma}$ is a finite set $\{\mathcal{V}_1, \dots, \mathcal{V}_n / n \in \mathbb{N}\}$ such that for each $1 \leq i \leq n$, \mathcal{V}_i is a Σ_i -view with $\Sigma_i = (S_i, \mathcal{F}_i, R_i)$. Moreover, $\tilde{\Sigma}$ satisfies for any couple of views $(\mathcal{V}_i, \mathcal{V}_j)$ of $\tilde{\Sigma}$ the following conditions :

- $\delta\text{int}[\mathcal{V}_i] = \delta\text{int}[\mathcal{V}_j] \implies \mathcal{V}_i = \mathcal{V}_j$ (unicity of view types definitions),
- if we note $S_{i,j} = S_i \cap S_j$ and $\delta S_{i,j} = \delta S_i \cap \delta S_j$, then :
 - $\forall\alpha \in S_{i,j}^*, \forall s \in S_{i,j}, \forall f : \alpha \rightarrow s, f \in \mathcal{F}_i \Leftrightarrow f \in \mathcal{F}_j$,
 - $\forall\alpha \in S_{i,j}^*, \forall r : \alpha, r \in R_i \Leftrightarrow r \in R_j$.

(same sorts share the same function names and predicate names)

Let us note $\tilde{\delta s} = \{\delta s_i\}$ (δs_i is the sort of interest of \mathcal{V}_i), and $\tilde{S} = \bigcup_{1 \leq i \leq n} S_i$.

Definition 15 (Projected state formula) Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be a composed mixed signature. Let \tilde{V} be a \tilde{S} -indexed family of sets. A projected state formula on $\tilde{\Sigma}$ is any well-formed formula built on :

EXTERNAL STRUCTURING VIEW T	
SPECIFICATION	COMPOSITION δ
imports A' generic on G variables V hides \bar{A}	is $id_i : Obj_i < I_i >$ axioms Ax_Θ with Φ, Ψ initially Φ_0

FIG. 3 – Korrigan syntax (Compositions).

- $x.\varphi$ where $x \in \tilde{V}_{\delta s}$ and $\varphi \in \delta Sen(\delta Sig[\mathcal{V}])$,
- propositional connectives in $\{\Rightarrow, \Leftrightarrow, \wedge, \vee, \neg\}$ and first-order quantifiers in $\{\forall, \exists\}$,
- $Qx.\varphi$ where $x \in \tilde{V}_{\delta s}$, φ is a projected state formula and $Q \in \{\forall, \exists\}$.

Let us note $\mathcal{P}\delta Sen(\tilde{\Sigma})$ the set of projected state formulas, and $\mathcal{P}Sen(\tilde{\Sigma})$ (set of projected formulas), the subset of $\mathcal{P}\delta Sen(\tilde{\Sigma})$ where φ in $x.\varphi$ is taken in the $Sen(Sig[\mathcal{V}])$ subset of $\delta Sen(\delta Sig[\mathcal{V}])$, and φ in $Qx.\varphi$ is a projected formula.

Definition 16 (Composed mixed specification) A composed mixed specification \tilde{C} is a 4-tuple $(\tilde{\Sigma}, \tilde{Ax}, \tilde{State}, \tilde{Init})$, where :

- $\tilde{\Sigma}$ is a composed mixed signature,
- $\tilde{Ax} \subseteq \mathcal{P}Sen(\tilde{\Sigma})$,
- $\tilde{State} \cup \tilde{Init} \subseteq \mathcal{P}\delta Sen(\tilde{\Sigma})$.

4.2 Example : Korrigan External Views

These views model the different compositions (ie aspect integration and parallel composition) of components.

Components are “glued” altogether in external structuring views (Figure 3) using both axioms and temporal logic formulas. This glue expresses a generalized form of synchronous product (for STS) and may be used to denote different concurrency modes and communication semantics. The δ component may be either LOOSE, ALONE or KEEP and is used in the operational semantics to express different concurrency modes (synchronous or asynchronous modes) and communication schemes. The **axioms** clause is used to link abstract guards that may exist in some components with operations defined in other components. The Φ and Φ_0 elements are state formulas expressing correct combinations of the components conditions (Φ) and initial ones (Φ_0). The Ψ element is a set of couples of transition formulas expressing which transitions have to be triggered at the same time (this expresses synchronization and communication). The COMPOSITION clauses may use a syntactic sugar : the *range* operator ($i : [1..N]$ or $i : [e_1, \dots, e_n]$), a bounded universal quantifier.

The denotation of a composition in KORRIGAN (with the notations of Fig. 3) is a composed mixed specification $(\tilde{\Sigma}, \tilde{Ax}, \tilde{State}, \tilde{Init})$ where :

- $\widetilde{\Sigma}$ is the composed mixed signature build on the views declared in the **is** clause;
- $\widetilde{\mathcal{A}\mathbf{x}}$ corresponds to the $\mathcal{A}\mathbf{x}_\theta$ axioms;
- each one of the free variables in the KORRIGAN glue formulas is implicitly quantified within the denotational semantics by \forall ;
- $\widetilde{\mathbf{State}}$ corresponds, with $(x.\psi_{i_x}, y.\psi_{i_y})$ being couples in Ψ , $x?\langle\psi\rangle =_{def} x.\langle\psi\rangle true$, $x!\langle\psi\rangle =_{def} x?\langle\psi\rangle \wedge \neg x?\langle\neg\psi\rangle$, $\bowtie_i^! =_{def} x!\psi_{i_x} \Leftrightarrow y!\psi_{i_y}$ and $\bowtie_i^? =_{def} x?\psi_{i_x} \Leftrightarrow y?\psi_{i_y}$:
 - for the LOOSE semantics to $\widetilde{\mathbf{State}}_{LOOSE} =_{def} \bigwedge_i \bowtie_i^!$ (nothing excepted synchronizings),
 - for the ALONE semantics to $\widetilde{\mathbf{State}}_{ALONE} =_{def} \widetilde{\mathbf{State}}_{LOOSE} \vee \bigwedge_i \forall j \in [x, y] (\neg j?\psi_{i_j} \Rightarrow \forall k \neq j \ k!\varkappa)$ (synchronizings or glue with \varkappa),
 - for the KEEP semantics to $\widetilde{\mathbf{State}}_{KEEP} =_{def} \widetilde{\mathbf{State}}_{LOOSE} \vee \bigwedge_i \forall j \in [x, y] (\neg j?\psi_{i_j} \Rightarrow \forall k \neq j \ \forall \psi \neq \psi_{i_k} \notin \Psi \ k?\psi \neq)$ (synchronizings or glue with anything that has not to be synchronized).
- $\widetilde{\mathcal{I}\mathbf{nit}}$ corresponds to Φ_0 ;
- the SPECIFICATION part is incorporated within any of the signatures of the views (ie the components).

The complexity of the $\widetilde{\mathbf{State}}$ formulas come from the fact that our logic expresses possibilities and KORRIGAN obligations.

KORRIGAN deals with a tree-structured specification. To obtain a denotational semantics for this, the structure is first flattened and then the above definitions yield (this is usual in algebraic specification structuring for example). Hence, the syntax is hierarchical, the semantics is flat.

In a previous work [14] we gave an operationnal semantics for the KORRIGAN formal specification language. However, this semantics we restricted to the integration of static and dynamic aspects and did not deal with the composition of several integrations. Using the mapping we presented here, one can obtain a denotational semantics for the full KORRIGAN language. One may then derive a full operationnal semantics from this denotational semantics in a similar way to what has been presented for Verilog in [23].

4.3 Denotational Semantics

We first define the models for composed mixed specifications, and then global states and environments for these models (to deal with the gluing between the individual models). To end, we define the validation of projected formulas (ie the constraints of the composition level) over composed models.

Definition 17 (Composed mixed model) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be a composed mixed signature. A composed mixed model over $\widetilde{\Sigma}$, or for short a $\widetilde{\Sigma}$ -model, $\widetilde{\mathcal{D}}$ consists on one view model $(\mathcal{D}_i, <^{\mathcal{D}_i})$ for each \mathcal{V}_i ($1 \leq i \leq n$), such that :*

- $\forall 1 \leq i, j \leq n, \forall s \in S_i \cap S_j, (M_i)_s = (M_j)_s$ (same sorts share the same interpretation ...),
- $\forall 1 \leq i, j \leq n$:
 - $\forall f \in F_i \cap F_j, f^{\mathcal{M}_i} = f^{\mathcal{M}_j}$,
 - $\forall r \in R_i \cap R_j, r^{\mathcal{M}_i} = r^{\mathcal{M}_j}$,
 - $\forall e \in \mathfrak{E}v_i \cap \mathfrak{E}v_j, e^{\mathcal{D}_i} = e^{\mathcal{D}_j}$.

(... and same interpretations for their functions, predicates and events²)

Let us note \tilde{M} the \tilde{S} -set obtained by gluing all S_i -sets M_i ($1 \leq i \leq n$) together. Finally, let us note $\tilde{M}_{\tilde{\delta S}}$ the restriction of \tilde{M} to sorts in $\tilde{\delta S}$.

Composed mixed states denote functions that, taking the identification of a component, yield the substitutions for this component's states (ie within this component's model).

Definition 18 (Composed mixed state) Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be a composed mixed signature. Let \tilde{V} be a \tilde{S} -indexed family of sets of variables. Let $\tilde{\mathcal{D}}$ be a $\tilde{\Sigma}$ -model. For every $1 \leq i \leq n$, let us note \tilde{V}_{S_i} the restriction of \tilde{V} to S_i . A composed mixed state over $\tilde{\mathcal{D}}$ is any $\tilde{\delta S}$ -indexed family of functions $\tilde{\gamma}_{\delta S_i} : \tilde{M}_{\delta S_i} \rightarrow M_i^{\tilde{V}_{S_i}}$ such that $\forall 1 \leq i, j \leq n, \forall n \in \tilde{M}_{\delta S_i} \cap \tilde{M}_{\delta S_j}, \tilde{\gamma}_{\delta S_i}(n) = \tilde{\gamma}_{\delta S_j}(n)$. Let us note $St[\tilde{\mathcal{D}}]$ the set of composed mixed states over $\tilde{\mathcal{D}}$.

Environments are used to take the gluing of different components into account within our global models. The components concerned are dealt with by the $\tilde{\sigma}$ part of the environments. Valuations are dealt with by the state ($\tilde{\gamma}$) part of the environments.

Definition 19 (Environment) With all the notations of Definition 18, an environment over $\tilde{\mathcal{D}}$ is a pair $\mathcal{E} = (\tilde{\sigma}, \tilde{\gamma})$ where $\tilde{\sigma} : \tilde{V}_{\tilde{\delta S}} \rightarrow \tilde{M}_{\tilde{\delta S}}$ is a function compatible with sorts (i.e. $\tilde{\sigma}(\tilde{V}_{\delta S}) \subseteq \tilde{M}_{\delta S}$) and $\tilde{\gamma} \in St[\tilde{\mathcal{D}}]$ such that :

- $\forall 1 \leq i \leq n, \forall m \in \tilde{M}_{\delta S_i}, \forall y \in V_i \cap \tilde{V}_{\delta S}, \tilde{\gamma}(m)(y) = \tilde{\sigma}(y)$
(identifiers denote one component only : the identification value of a component - ie the value of the variable denoting it - and the one known in other components - ie the value of the same variable in other components - are equal)
- $\forall 1 \leq i, j \leq n, \forall m \in \tilde{M}_{\delta S_i}, \forall m' \in \tilde{M}_{\delta S_j}, \forall x \in V_i \cap V_j, \tilde{\gamma}_{\delta S_i}(m)(x) = \tilde{\gamma}_{\delta S_j}(m')(x)$
(the same variable used in two states of two components have the same value - value exchange)

Definition 20 (Satisfaction of projected state formulas) Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be a composed mixed signature. Let $\tilde{\varphi} \in \mathcal{P}\delta Sen(\Sigma)$ be a projected state formula. Let $\tilde{\mathcal{D}}$ be a $\tilde{\Sigma}$ -model. Let $\mathcal{E} = (\tilde{\sigma}, \tilde{\gamma})$ be an environment. Let us note $\tilde{\delta V}$ the restriction of \tilde{V} to \tilde{S} . $\tilde{\mathcal{D}}$ satisfies $\tilde{\varphi}$ for \mathcal{E} , noted $\tilde{\mathcal{D}} \models_{\mathcal{E}} \tilde{\varphi}$, if and only if :

²Overloading can easily be achieved through renaming.

- if $\tilde{\varphi}$ is of the form $x.\psi$ with $x \in \widetilde{V}_{\delta s_i}$ then $\tilde{\mathcal{D}} \models_{\mathcal{E}} \tilde{\varphi}$ if and only if $\mathcal{D}_i \models_{\tilde{\gamma}(\tilde{\sigma}(x))} \psi$,
- if $\tilde{\varphi}$ is of the form $\forall x.\tilde{\psi}$ then $\tilde{\mathcal{D}} \models_{\mathcal{E}} \tilde{\varphi}$ if and only if for any environment \mathcal{E}' such that for all $y \neq x \in \delta\tilde{V}$, $\tilde{\sigma}(y) = \tilde{\sigma}'(y)$, $\tilde{\mathcal{D}} \models_{\mathcal{E}'} \tilde{\psi}$.
- if $\tilde{\varphi}$ is of the form $\exists x.\tilde{\psi}$ then $\tilde{\mathcal{D}} \models_{\mathcal{E}} \tilde{\varphi}$ if and only if there exists an environment \mathcal{E}' such that for all $y \neq x \in \delta\tilde{V}$, $\tilde{\sigma}(y) = \tilde{\sigma}'(y)$, $\tilde{\mathcal{D}} \models_{\mathcal{E}'} \tilde{\psi}$.
- propositional connectives and first-order quantifiers are handled as usual.

Satisfaction of projected formulas is handled in the same way. As usual, $\tilde{\mathcal{D}} \models \tilde{\varphi}$ is defined as $\forall \mathcal{E}, \tilde{\mathcal{D}} \models_{\mathcal{E}} \tilde{\varphi}$.

Notation 3 Let $\tilde{\mathcal{D}}$ be a $\tilde{\Sigma}$ -model. Let us define the pre-order on $St[\tilde{\mathcal{D}}]$ as follows :

$$\tilde{\gamma} \prec \tilde{\gamma}' \Leftrightarrow (\forall 1 \leq i \leq n, \forall m \in \widetilde{M}_{\delta s_i}, \tilde{\gamma}(m) <^{\mathcal{D}_i} \tilde{\gamma}'(m))$$

Definition 21 (Models of specification) Let $\tilde{C} = (\tilde{\Sigma}, \tilde{\mathcal{A}x}, \tilde{\mathcal{S}tate}, \tilde{\mathcal{I}nit})$ be a composed mixed specification. A \tilde{C} -model is a $\tilde{\Sigma}$ -model $\tilde{\mathcal{D}}$ such that :

- $\tilde{\mathcal{D}} \models \tilde{\mathcal{A}x} \cup \tilde{\mathcal{S}tate}$,
- $\forall \tilde{\gamma} \in St[\tilde{\mathcal{D}}], (\forall \tilde{\sigma}, \tilde{\mathcal{D}} \models_{(\tilde{\sigma}, \tilde{\gamma})} \tilde{\mathcal{I}nit}) \Rightarrow (\forall \tilde{\gamma}' \in St[\tilde{\mathcal{D}}], \tilde{\gamma} \prec \tilde{\gamma}')$.

5 Relating the Logic with Specification Languages

In this paper, we proposed a general denotational model which may be used to give denotational semantics to the existing mixed specification languages and to define new specific languages. We also presented such a mapping for the KORRIGAN language. In this Section, we explain briefly how it may be done also for the LOTOS language simply by relating LOTOS and KORRIGAN and stating differences between these two languages.

As presented in [31] where we translate a subpart of KORRIGAN into LOTOS to be able to reuse LOTOS specific tools, KORRIGAN is more expressive than LOTOS, hence any process³ in LOTOS can be expressed in a restricted form of KORRIGAN. One can easily see that the symbolic semantics of a sequential LOTOS behaviour (as presented in [13]) corresponds to an STS. Such an STS is the basic way of specifying a process using KORRIGAN. As far as the glue (ie LOTOS parallel composition) is concerned, the translation means restricting KORRIGAN to the ALONE (anything not to be glued happens alone) kind of parallel composition and using only gluing formulas of the form $(x_i.po_i, x_j.po_j)$ with x_i and x_j being subprocesses identifiers, p being any gate present in the $[[[]]]$ LOTOS operator (δ has to be expressed explicitly), and o_i and o_j being the LOTOS offers associated with the p gate in the behaviours. Actually, only the sequence ($;$), choice ($[[[]]]$), recursive call, and parallel composition ($[[[]]]$, $[[[]]]$ and $[[[]]]$) operators of LOTOS are directly taken into account, but one can easily

³Indeed, its semantics.

achieve some extensions of KORRIGAN to deal with them. However, a mapping of LOTOS into our logical framework can be done in a similar way than we did for KORRIGAN. We plan to do this to prove translations of LOTOS into KORRIGAN are semantically correct (both mappings into our framework correspond to behaviourally equivalent processes).

6 Fundamental Results : an institution for our logic

In the field of axiomatic specifications (i.e. based on logical frameworks), a lot of different formalisms have been defined, each one devoted to some aspects of software engineering (typing, dynamical aspects, temporality, real-time, theorem-proving issues, modularity issues, refinement, etc.). Most of the time, beside the original idea underlying a new formalism, the authors must develop a lot of inevitable formal results which generalise some well-known classical results. Since J. Goguen and R. Burstall's works with their institutions [21], it has been established that such results can be generalised at a meta-level. Therefore, showing that a formalism is an institution allows the authors to take advantage from this knowledge in order to directly use general results of institutions such as existence of quotients, free models, amalgamation properties underlying to modularity issues, and refinement issues.

Institutions formally axiomatise the notion of logical system from a model theoretical point of view. An institution is a quadruple (Sig, Sen, Mod, \models) where Sig is a category of *signatures*, $Sen : Sig \rightarrow Set$ is a functor which maps every signature to its set of sentences, $Mod : Sig \rightarrow Cat$ is a contravariant functor which maps every signature to its category of models, and $\models = (\models_{\Sigma})_{\Sigma \in |Sig|}$ is a Sig -indexed family of relations $\models_{\Sigma} \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$. Given a signature morphism $\sigma : \Sigma \rightarrow \Sigma'$, $Mod(\sigma) : Mod(\Sigma') \rightarrow Mod(\Sigma)$ is called the *reduct functor*. Moreover, this quadruple satisfies the following property, so-called, *satisfaction condition* : $\forall (\sigma : \Sigma \rightarrow \Sigma') \in Hom_{Sig}, \forall \mathcal{M} \in |Mod(\Sigma')|, \forall \varphi \in Sen(\Sigma), \mathcal{M} \models_{\Sigma'} Sen(\sigma)(\varphi) \Leftrightarrow Mod(\sigma)(\mathcal{M}) \models_{\pm} \varphi$.

6.1 The category of composed mixed models

In first, we must define an appropriate morphism notion between view model.

Definition 22 (View model morphisms) *Given a view \mathcal{V} , a \mathcal{V} -morphism between two view models for \mathcal{V} $(\mathcal{D}, <^{\mathcal{D}})$ and $(\mathcal{D}', <^{\mathcal{D}'})$ is a Σ -morphism $\mu : \mathcal{M} \rightarrow \mathcal{M}'$ such that : let us note for any $\sigma : V \rightarrow M, \mu(\sigma) : V \rightarrow M'$ the application defined by $x \mapsto \mu(\sigma(x))$. Finally, let R be a binary relation on M^V , then let us note $\mu(R) = \{(\mu(\sigma), \mu(\sigma')) \mid (\sigma, \sigma') \in R\}$.*

- Event compatibility : $\mu(e^{\mathcal{D}}) \subseteq e^{\mathcal{D}'}$
- preorder compatibility : $\mu(<^{\mathcal{D}}) \subseteq <^{\mathcal{D}'}$

Therefore, we extend it to composed mixed models.

Definition 23 (Composed mixed model morphisms) Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be a composed mixed signature. A $\tilde{\Sigma}$ -morphism between two $\tilde{\Sigma}$ -models $\tilde{\mathcal{D}}$ and $\tilde{\mathcal{D}}'$ is defined for each $1 \leq i \leq n$ by a \mathcal{V}_i -morphism $\mu_i : \mathcal{D}_i \rightarrow \mathcal{D}'_i$. Moreover, for every $1 \leq i, j \leq n$ and every $s \in S_i \cap S_j$, we have $(\mu_i)_s = (\mu_j)_s$.

Clearly, $\tilde{\Sigma}$ -models and $\tilde{\Sigma}$ -morphisms form a category. Let us note it $Mod(\tilde{\Sigma})$.

6.2 Reduct functor and satisfaction condition

An essential ingredient which is missing is an appropriate morphism notion for composed mixed signatures.

Definition 24 (Dynamic signature morphism) Let $\delta\Sigma = ((\delta S, \delta s), \mathfrak{E}\mathfrak{v})$ and $\delta\Sigma' = ((\delta S', \delta s'), \mathfrak{E}\mathfrak{v}')$ be two dynamic signatures. A dynamic signature morphism $\nu : \delta\Sigma \rightarrow \delta\Sigma'$ is a signature morphism $\nu : \Sigma \rightarrow \Sigma'$ such that :

- $\nu(\delta S) \subseteq \delta S'$ and $\nu(\delta s) = \delta s'$,
- for every event $e : \rho$ in $\delta\Sigma$, $\nu(e) : \nu(\rho)$ belongs to $\delta\Sigma'$ where $\nu(\rho)$ is the natural extension of ρ to dynamic profiles of $P_{S, \delta S}$.

Definition 25 (View morphism) Let $\mathcal{V} = (\delta\Sigma, \mathcal{A}x, State, \mathcal{I}nit)$ and $\mathcal{V}' = (\delta\Sigma', \mathcal{A}x', State', \mathcal{I}nit')$ be two views. A view morphism $\nu : \mathcal{V} \rightarrow \mathcal{V}'$ is a dynamic signature morphism $\nu : \delta\Sigma \rightarrow \delta\Sigma'$ such that $\nu(\mathcal{A}x) \subseteq \mathcal{A}x'$, $\nu(State) \subseteq State'$ and $\nu(\mathcal{I}nit) \subseteq \mathcal{I}nit$.

Given a view morphism ν , let us note $\bar{\nu}$ its canonical extension to dynamic formulas.

Definition 26 (Composed mixed signature morphisms) Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ and $\tilde{\Sigma}' = \{\mathcal{V}'_1, \dots, \mathcal{V}'_m\}$ be two composed mixed signatures. A Composed mixed signature morphism $\tilde{\nu}$ is a set $\{\nu_1, \dots, \nu_n\}$ such that :

- for every $1 \leq i \leq n$, $\nu_i : \mathcal{V}_i \rightarrow \mathcal{V}'_i$ is a view morphism;
- for every $1 \leq i, j \leq n$:
 - $\forall s \in S_i \cap S_j, \nu_i(s) = \nu_j(s)$;
 - $\forall f \in F_i \cap F_j, \nu_i(f) = \nu_j(f)$;
 - $\forall r \in R_i \cap R_j, \nu_i(r) = \nu_j(r)$;
 - $\forall e \in \mathfrak{E}\mathfrak{v}_i \cap \mathfrak{E}\mathfrak{v}_j, \nu_i(e) = \nu_j(e)$;

Given a composed mixed signature morphism $\tilde{\nu}$, let us note $\bar{\tilde{\nu}}$ its canonical extension to both projected formulas and projected state formulas.

Proposition 1 Let $\nu : \mathcal{V} \rightarrow \mathcal{V}'$ be a view morphism. Let $(\mathcal{D}', <^{\mathcal{D}'})$ be a \mathcal{V} -model. Let us note $\mathcal{D}'_{\upharpoonright \nu}$ the dynamic-model \mathcal{D} for $\delta\Sigma$ defined as follows :

- the Σ -model $\mathcal{M} = \mathcal{M}'_{\upharpoonright \nu}$, and
- for every $e \in \mathfrak{E}\mathfrak{v}$, $e^{\mathcal{D}} = \{(\sigma_{\upharpoonright \nu}, \sigma'_{\upharpoonright \nu}) \mid (\sigma, \sigma') \in e^{\mathcal{D}'}\}$

where $\sigma_{\upharpoonright \nu}$ is the restriction of σ to sorts of S .

Then, for any $\varphi \in \delta Sen(\delta\Sigma)$, we have $:\mathcal{D}' \models \bar{\nu}(\varphi) \iff \mathcal{D}'_{\upharpoonright \nu} \models \varphi$.

Proof 1 For every $s \in S$, we do not prune any element of $M'_{\nu(s)}$. Moreover, semantics of functions, relations and events of \mathcal{V} are preserved between \mathcal{D}' and $\mathcal{D}'_{\uparrow\nu}$. Therefore, for every atom α , and every interpretation $\sigma \in V^{M'}$, we have : $\mathcal{D}' \models_{\sigma} \bar{v}(\alpha) \iff \mathcal{D}'_{\uparrow\nu} \models_{\sigma_{\uparrow\nu}} \alpha$.

Corollary 1 With all the notations of Proposition 1, Let us note $<^{\mathcal{D}'_{\uparrow\nu}}$ the preorder on M^V defined by : $<^{\mathcal{D}'_{\uparrow\nu}} = \{(\sigma_{\uparrow\nu}, \sigma'_{\uparrow\nu}) \mid (\sigma, \sigma') \in <^{\mathcal{D}'}\}$. Then, the couple $(\mathcal{D}'_{\uparrow\nu}, <^{\mathcal{D}'_{\uparrow\nu}})$ is a \mathcal{V}' -model.

Proposition 1 is the satisfaction condition for the view specification logic.

Definition 27 (Reduct functor) Let $\tilde{v} : \tilde{\Sigma} \rightarrow \tilde{\Sigma}'$ be a composed mixed signature morphism. Let $\tilde{\mathcal{D}}'$ be a $\tilde{\Sigma}'$ -model. The reduct functor $_{}_{\uparrow\tilde{v}} : Mod(\tilde{\Sigma}') \rightarrow Mod(\tilde{\Sigma})$ is defined as follows :

- for each $\tilde{\Sigma}'$ -model $\tilde{\mathcal{D}}'$, $\tilde{\mathcal{D}}'_{\uparrow\tilde{v}}$ is the $\tilde{\Sigma}$ -model $\tilde{\mathcal{D}}$ where for every $1 \leq i \leq n$, $\mathcal{D}_i = (\mathcal{D}'_i)_{\uparrow\nu_i}$ and $<^{\mathcal{D}_i} = (<^{\mathcal{D}'_i})_{\uparrow\nu_i}$ (by Corollary 1, the couple $(\mathcal{D}_i, <^{\mathcal{D}_i})$ is a \mathcal{V}_i -model).
- for each $\tilde{\Sigma}'$ -morphism $\tilde{\mu} : \tilde{\mathcal{D}} \rightarrow \tilde{\mathcal{D}}'$, $\tilde{\mu}_{\uparrow\tilde{v}} : \tilde{\mathcal{D}}_{\uparrow\tilde{v}} \rightarrow \tilde{\mathcal{D}}'_{\uparrow\tilde{v}}$ is the $\tilde{\Sigma}$ -morphism defined for all $1 \leq i \leq n$ by the \mathcal{V}_i -morphism $(\mu_i)_{\uparrow\nu_i}$ where μ_i is the corresponding \mathcal{V}'_i -morphism of $\tilde{\mu}$.

Theorem 1 (Satisfaction condition) Let $\tilde{v} : \tilde{\Sigma} \rightarrow \tilde{\Sigma}'$ be a composed mixed signature morphism. Let $\tilde{\mathcal{D}}'$ be a $\tilde{\Sigma}'$ -model. Let $\tilde{\varphi}$ be projected formula or a projected state formula over $\tilde{\Sigma}$. Then, we have : $\tilde{\mathcal{D}}' \models \tilde{v}(\tilde{\varphi}) \iff \tilde{\mathcal{D}}'_{\uparrow\tilde{v}} \models \tilde{\varphi}$.

Proof 2 This directly results from the fact that every syntactical part of $\tilde{\Sigma}$ preserves its semantics from $\tilde{\mathcal{D}}'$ to $\tilde{\mathcal{D}}'_{\uparrow\tilde{v}}$.

Corollary 2 Let $\tilde{C} = (\tilde{\Sigma}, \tilde{Ax}, \tilde{State}, \tilde{Init})$ and $\tilde{C}' = (\tilde{\Sigma}', \tilde{Ax}', \tilde{State}', \tilde{Init}')$ be two composed mixed specifications such that there is a signature morphism $\tilde{v} : \tilde{\Sigma} \rightarrow \tilde{\Sigma}'$, $\tilde{v}(\tilde{Ax}) \subseteq \tilde{Ax}'$, $\tilde{v}(\tilde{State}) \subseteq \tilde{State}'$, and $\tilde{v}(\tilde{Init}) \subseteq \tilde{Init}'$. Then, the reduct functor $_{}_{\uparrow\tilde{v}}$ can be co-restricted to $_{}_{\uparrow\tilde{v}} : Mod(\tilde{C}') \rightarrow Mod(\tilde{C})$.

In order to fit composed mixed specifications within the institution framework, we must mention for every formula the signature on which it is defined.

Definition 28 Let $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$ be composed mixed signature. A well-formed $\tilde{\Sigma}$ -formula is :

- for every $1 \leq i \leq n$, any pair (\mathcal{V}_i, φ) where $\varphi \in \delta Sen(\delta \Sigma_i)$;
- and any pair $(\tilde{\Sigma}, \tilde{\varphi})$ where $\tilde{\varphi}$ is any projected formula or a projected state formula over $\tilde{\Sigma}$.

Theorem 2 (Institution of composed mixed specifications)

The quadruple $INS_{CMS} = (Sig_{CMS}, Mod_{CMS}, Sen_{CMS}, \models_{CMS})$ is an institution whereby :

- Sig_{CMS} is the category of composed mixed signatures and composed mixed signature morphisms.
- The functor $Sen_{CMS} : Sig_{CMS} \rightarrow Set$ maps :
 - each composed mixed signature $\tilde{\Sigma}$ to the set of well-formed $\tilde{\Sigma}$ -formulas (cf. Definition 28) and
 - each composed mixed signature morphism $\tilde{\nu}$ to $\tilde{\bar{\nu}}$.
- The contravariant functor $Mod_{CMS} : Sig_{CMS} \rightarrow Cat$ maps :
 - each composed mixed signature $\tilde{\Sigma}$ to the category $Mod(\tilde{\Sigma})$ and
 - each composed mixed signature morphism $\tilde{\sigma}$ to the reduct functor $_{|\tilde{\sigma}}$.
- $\models_{CMS} = (\models_{\tilde{\Sigma}})_{\tilde{\Sigma} \in |Sig_{CMS}|}$ where for each composed system signature $\tilde{\Sigma} = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$, $\models_{\tilde{\Sigma}}$ is the satisfaction relation of a well-formed $\tilde{\Sigma}$ -formula Γ by a $\tilde{\Sigma}$ -model $\tilde{\mathcal{D}}$ defined as follows :
 - if Γ is of the form $(\tilde{\Sigma}, \tilde{\varphi})$ then $\tilde{\mathcal{D}} \models_{\tilde{\Sigma}} \Gamma$ iff $\tilde{\mathcal{D}} \models \tilde{\varphi}$ (cf. Definition 20) ;
 - if Γ is of the form (\mathcal{V}_i, φ) then $\tilde{\mathcal{D}} \models_{\tilde{\Sigma}} \Gamma$ iff $\mathcal{D}_i \models \varphi$ (cf. Definition 12).

By Proposition 1, we can easily define, by following the same arguments than in Theorem 2, the institution for view specifications. Let us note it Ins_{view} .

6.3 Refinement of views

Here, we are going to take advantage from showing that the view specification logic is an institution to define a refinement theory for views. Moreover, we will show that usual both horizontal and vertical refinements throughout composed mixed specification structuration hold. To achieve this purpose, we will use notations and results established both in [7] and [36].

6.3.1 Syntax of refinement.

Specification refinement consists on removing axioms of specifications and replacing them by more concrete ones. In our framework, this will be simply defined as follows :

Definition 29 (View refinement) *A view \mathcal{V}_{impl} is a refinement of a view \mathcal{V} if and only if $Sig[\mathcal{V}_{impl}] = Sig[\mathcal{V}]$. Let us note $\mathcal{V}_{impl} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}$ such a refinement.*

6.3.2 Semantics of refinement.

A refinement will be correct provided that the behaviour of the implementation is indistinguishable from the behaviour of the higher level specification under consideration. Semantically, this is expressed as follows :

Definition 30 (Semantic refinement) *Let $\mathcal{V} \rightsquigarrow_{\Sigma} \mathcal{V}'$ be a refinement. \mathcal{V} is a semantic refinement of \mathcal{V}' , written $\mathcal{V} \Vdash_{\Sigma} \mathcal{V}'$, if and only if $Mod(\mathcal{V}) \subseteq Mod(\mathcal{V}')$.*

Usually, specification refinements allow us to extend signatures. This can be abstractly obtained from the following basic set of specification building operations :

- union : for any two views \mathcal{V}_1 and \mathcal{V}_2 with $Sig[\mathcal{V}] = Sig[\mathcal{V}']$,
 $\mathcal{V}_1 \cup \mathcal{V}_2$ is a view with semantics
- $$Mod(\mathcal{V}_1 \cup \mathcal{V}_2) = Mod(\mathcal{V}_1) \cap Mod(\mathcal{V}_2)$$
- translate : for any view \mathcal{V} and any signature morphism $\nu : Sig[\mathcal{V}] \rightarrow \Sigma'$,
translate \mathcal{V} by ν is a view with semantics
- $$Mod(\mathbf{translate\ } \mathcal{V} \mathbf{ by\ } \nu) = \{(\mathcal{D}', <\mathcal{D}'\} \mid (\mathcal{D}'|_{\nu}, (<\mathcal{D}')|_{\nu}) \in Mod(\mathcal{V})\}$$
- derive : for any view \mathcal{V}' and any signature morphism $\nu : \Sigma \rightarrow Sig[\mathcal{V}']$,
derive from \mathcal{V}' by ν is a view with semantics
- $$Mod(\mathbf{derive\ from\ } \mathcal{V}' \mathbf{ by\ } \nu) = Mod(\mathcal{V})|_{\nu}$$

Definition 31 (Conservative extension along ν) *With all the notations of Definition 30, \mathcal{V} is a conservative extension of \mathcal{V}' along ν if and only if $Mod(\mathcal{V}_{impl})|_{\nu} = Mod(\mathcal{V})$.*

From there, we can instantiate the institution independent proof system of [7] and obtain the following rules :

Definition 32 (Rules) *Let $\mathcal{V} = (\delta Sig[\mathcal{V}], Ax, State, Init)$. Let iso be a signature isomorphism. Let ν be a signature morphism. The family of refinement relations $(\rightsquigarrow_{\Sigma})_{\Sigma \in |Sig|}$ is defined by the following set of rules :*

- (Basic) $\frac{\mathcal{V} \models Ax' \cup State'}{(Sig[\mathcal{V}], Ax', State', Init) \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}} \mathcal{V} = (\delta Sig[\mathcal{V}], Ax, State, Init)$
- (Sum) $\frac{\mathcal{V}_1 \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V} \quad \mathcal{V}_2 \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}}{\mathcal{V}_1 \cup \mathcal{V}_2 \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}} \quad (Trans_1) \frac{\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}]} \mathbf{translate\ } \mathcal{V}' \mathbf{ by\ } iso^{-1}}{\mathbf{translate\ } \mathcal{V} \mathbf{ by\ } iso \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}'}$
- (Trans₂) $\frac{\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}]} \mathbf{derive\ from\ } \mathcal{V}' \mathbf{ by\ } \nu}{\mathbf{translate\ } \mathcal{V} \mathbf{ by\ } \nu \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}'}$
- (Derive) $\frac{\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}''}{\mathbf{derive\ from\ } \mathcal{V}' \mathbf{ by\ } \nu \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}''} \quad \mathcal{V}'' \text{ is a conservative extension of } \mathcal{V}' \text{ along } \nu$
- (Trans - equiv) $\frac{\mathbf{translate\ (translate\ } \mathcal{V} \mathbf{ by\ } iso) \mathbf{ by\ } \nu \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}''}{\mathbf{translate\ } \mathcal{V} \mathbf{ by\ } \nu \circ iso \rightsquigarrow_{Sig[\mathcal{V}']} \mathcal{V}''}$

Theorem 3 (Soundness and completeness) *For any views \mathcal{V} and \mathcal{V}' such that $Sig[\mathcal{V}] = Sig[\mathcal{V}']$, we have : $\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}' \iff \mathcal{V} \Vdash_{Sig[\mathcal{V}]} \mathcal{V}'$.*

(see the proof of Theorem 4.5 in [7])

6.3.3 Composition.

Of course, it is not reasonable to implement a specification as a whole in a single step. Large softwares usually require many refinement steps before obtaining efficient programs. This leads to the notion of sequential composition of implementations steps. Usually, composition of enrichment is divided into two concepts mainly : horizontal composition and vertical composition. Horizontal composition deals with refinement of subparts of composed mixed specifications

when they are structured into specification “blocks”. In our framework, blocks are views. On the contrary, vertical composition deals with many refinement steps.

Notation 4 *Let \tilde{C} be a composed mixed specification. Let \mathcal{V} be a view belonging to \tilde{C} and let $\mathcal{V}_{impl} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}$. Let us note $\tilde{C}[\mathcal{V}/\mathcal{V}_{impl}]$ the composed mixed specification obtained from \tilde{C} by substituting \mathcal{V} by \mathcal{V}_{impl} .*

Theorem 4 (Horizontal refinement) *With all the previous notations, if $\mathcal{V}_{impl} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}$, then : $\forall \tilde{\varphi} \in Sen(\tilde{\Sigma}), \tilde{C} \models \tilde{\varphi} \implies \tilde{C}[\mathcal{V}/\mathcal{V}_{impl}] \models \tilde{\varphi}$.*

Theorem 5 (Vertical composition) *The following rule is sound :*

$$\frac{\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}' \quad \mathcal{V}' \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}''}{\mathcal{V} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}''}$$

Proof 3 *Uniquely from the hypothesis that $\mathcal{V}_{impl} \rightsquigarrow_{Sig[\mathcal{V}]} \mathcal{V}$, both reduct functors from $Mod(\mathcal{V}_{impl})$ to $Mod(\mathcal{V})$ and from $Mod(\tilde{C}[\mathcal{V}/\mathcal{V}_{impl}])$ to $Mod(\mathcal{V})$ are [36]’s constructors where the specification-building operation is the usual translate operation taken from [35]. Consequently, this can be directly obtained from the [36]’s vertical composition theorem.*

7 Conclusion

In this paper we presented a logic and denotational model for mixed specifications. We first presented a logic for simple mixed systems (views) and then extended it to take into account the specification of components and composition of components. This logic has been applied to give a denotational semantics to KORRIGAN. We also have explained how LOTOS could be related to our logic (studying relations between LOTOS and KORRIGAN) and shown that our logic can be seen as an institution, which has the benefits of enabling a refinement theory for mixed specifications mapped into our logic.

Our works with KORRIGAN lead to the definition of abstractions over such models of mixed specifications. A track of research is actually devoted to the study of symbolic models (not in the sense of BDD but rather of symbolic transitions systems). Our next work will be to study symbolic bisimulation and symbolic temporal logics (with actions) in the context of our logic, and to see if current works done specifically for CSP with value passing or for LOTOS may be related to this work. Further on, we are beginning a work on the categorization of notions of aspects and aspect combination operators. We already noticed common points between the definition of static and dynamic aspects within our logic and we think this can be investigated.

Références

- [1] Unified Modeling Language, Version 1.3. Technical report, Rational Software Corp., <http://www.rational.com/uml>, June 1999.
- [2] Michel Allemand, Christian Attiogbe, Pascal Poizat, Jean-Claude Royer, and Gwen Salaün. SHE'S Project : a Report of Joint Works on the Integration of Formal Specification Techniques. In *INT'2002, Workshop on Integration of Specification Techniques with Applications in Engineering*, 2002. to appear.
- [3] Pascal André. *Méthodes formelles et à objets pour le développement du logiciel : Etudes et propositions*. PhD Thesis, Université de Rennes I (Institut de Recherche en Informatique de Nantes), Juillet 1995.
- [4] Egidio Astesiano, Maura Cerioli, and Gianna Reggio. Plugin Data Constructs into Paradigm-Specific Languages : Towards an Application to UML. In T. Rus, editor, *International Conference on Algebraic Methodology And Software Technology (AMAST'2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 273–292. Springer-Verlag, 2000.
- [5] Egidio Astesiano and Gianna Reggio. Labelled Transition Logic : An Outline. Technical Report TR-96-20, DISI, 1996.
- [6] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1) :25–29, January 1988.
- [7] Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 2002. to appear.
- [8] Manfred Broy and Martin Wirsing. Algebraic State Machines. In T. Rus, editor, *International Conference on Algebraic Methodology And Software Technology (AMAST'2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 89–118. Springer-Verlag, 2000.
- [9] Glenn Bruns. *Distributed Systems Analysis with CCS*. International Series in Computer Science. Prentice-Hall, 1997.
- [10] Robert Büssow, Robert Geisler, Wolfgang Grieskamp, and Marcus Klar. Integrating Z with Dynamic Modeling Techniques for the Specification of Reactive Systems. 1998.
- [11] Muffy Calder, Savi Maharaj, and Carron Shankland. An Adequate Logic for Full LOTOS. In Pamela Zave J. N. Oliveira, editor, *Formal Methods Europe (FME)*, volume 2021 of *Lecture Notes in Computer Science*, pages 384–395. Springer-Verlag, 2001.
- [12] Muffy Calder, Savi Maharaj, and Carron Shankland. A Modal Logic for Full LOTOS Based on Symbolic Transition Systems. *The Computer Journal*, 45(1) :55–61, 2002.
- [13] Muffy Calder and Carron Shankland. A Symbolic Semantics and Bisimulation for Full LOTOS. In Myungchul Kim, Byoungmoon Chin, Sungwon Kang, and Danhyung Lee, editors, *Formal Techniques for Networked and*

- Distributed Systems (FORTE)*, volume 197 of *IFIP Conference Proceedings*, pages 185–200. Kluwer, 2001.
- [14] Christine Choppy, Pascal Poizat, and Jean-Claude Royer. A Global Semantics for Views. In T. Rus, editor, *International Conference on Algebraic Methodology And Software Technology (AMAST'2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 165–180. Springer-Verlag, 2000.
 - [15] Christine Choppy, Pascal Poizat, and Jean-Claude Royer. Formal Specification of Mixed Components with Korrigan. In *Asia-Pacific Software Engineering Conference (APSEC'2001)*, pages 169–176. IEEE Computer Society, 2001.
 - [16] Christine Choppy, Pascal Poizat, and Jean-Claude Royer. Specification of Mixed Systems in KORRIGAN with the Support of an UML-Inspired Graphical Notation. In H. Hussmann, editor, *Fundamental Approaches to Software Engineering (FASE'2001)*, volume 2029 of *Lecture Notes in Computer Science*, pages 124–139. Springer-Verlag, 2001.
 - [17] Christine Choppy, Pascal Poizat, and Jean-Claude Royer. The KORRIGAN Environment. *Journal of Universal Computer Science*, 7(1) :19–36, 2001. Special issue on Tools for System Design and Verification.
 - [18] Hartmut Ehrig and Fernando Orejas. Integration Paradigm for Data Type and Process Specification Techniques”. *Bulletin of EATCS - Formal Specification Column*, (65), 1998.
 - [19] Jan Ellsberger, Dieter Hogrefe, and Amardeo Sarma. *SDL : Formal Object-oriented Language for Communicating Systems*. Prentice-Hall, 1997.
 - [20] Clemens Fischer. How to Combine Z with a Process Algebra. In J. P. Bowen, A. Fett, and M. G. Hinchey, editors, *ZUM'98 : the Z Formal Notation*, volume 1493 of *Lecture Notes in Computer Science*, pages 5–23. Springer-Verlag, 1998.
 - [21] Joseph A. Goguen and Rod M. Burstall. Institutions : abstract model theory for specifications and programming. *Journal of the ACM*, 39(1) :95–146, 1992.
 - [22] Matthew Hennessy and Huimin Lin. Symbolic Bisimulations. *Theoretical Computer Science*, 138(2) :353–389, 1995.
 - [23] Zhu Huibiao, Jonathan P. Bowen, and He Jifeng. Deriving Operational Semantics from Denotational Semantics for Verilog. In *Asia-Pacific Software Engineering Conference (APSEC'2001)*, pages 177–184. IEEE Computer Society, 2001.
 - [24] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In M. Aksit and Matsuo S., editors, *ECOOP'97*, volume 1241 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
 - [25] Leslie Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3) :872–923, 1994.

- [26] Huimin Lin. Model Checking Value-passing Processes. In *Asia-Pacific Software Engineering Conference (APSEC'2001)*, pages 3–10. IEEE Computer Society, 2001.
- [27] S. Mauw and G. J. Veltink. An introduction to psf_d . In J. Diaz and F. Orejas, editors, *TAPSOFT'89*, volume 352 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.
- [28] Fernando Orejas. *Algebraic Foundations of System Specification*, chapter Structuring and Modularity. Springer-Verlag, 1999.
- [29] Pascal Poizat. *KORRIGAN : a Formalism and a Method for the Structured Formal Specification of Mixed Systems*. PhD thesis, Institut de Recherche en Informatique de Nantes, Université de Nantes, December 2000. available at URL `ftp://ftp.lami.univ-evry.fr/pub/specif/poizat/documents/these.ps.gz`, in French.
- [30] Pascal Poizat, Christine Choppy, and Jean-Claude Royer. Concurrency and Data Types : A Specification Method. An Example with LOTOS. In J. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, Selected Papers of the 13th International Workshop on Algebraic Development Techniques (WADT'98)*, volume 1589 of *Lecture Notes in Computer Science*, pages 276–291, Lisbon, Portugal, 1999. Springer-Verlag.
- [31] Pascal Poizat, Christine Choppy, and Jean-Claude Royer. From Informal Requirements to COOP : a Concurrent Automata Approach. In J.M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems*, volume 1709 of *Lecture Notes in Computer Science*, pages 939–962, Toulouse, France, 1999. Springer-Verlag.
- [32] Pascal Poizat and Jean-Claude Royer. Une proposition de composants formels. In *Langages et Modèles à Objets (LMO)*, volume 8(1–2) of *Revue L'Objet*, pages 231–245, 2002.
- [33] Julian Rathke and Matthew Hennessy. Local Model Checking for Value-Passing Processes (Extended Abstract). In Martín Abadi and Takayasu Ito, editors, *Third International Symposium on Theoretical Aspects of Computer Software TACS'97*, volume 1281 of *Lecture Notes in Computer Science*, pages 250–266. Springer-Verlag, 1997.
- [34] Gianna Reggio and Lorenzo Repetto. CASL-Chart : A Combination of Statecharts and of the Algebraic Specification Language CASL. In T. Rus, editor, *International Conference on Algebraic Methodology And Software Technology (AMAST'2000)*, volume 1816 of *Lecture Notes in Computer Science*, pages 243–257. Springer-Verlag, 2000.
- [35] Donald Sannella and Andrzej Tarlecki. Building specifications in an arbitrary institution. In *Intl. Symposium on Semantics of Data Types*, volume 173 of *Lect. Notes Comp. Sci.*, pages 337–356. Springer-Verlag, 1984.

- [36] Donald Sannella and Andrzej Tarlecki. Toward formal development of programs from algebraic specifications : implementations revisited. *Acta Informatica*, 25 :233–281, 1988.