A Logic with Temporal Glue for Mixed Specifications

Marc Aiguier, Fabrice Barbier and Pascal Poizat [1]

LaMI, UMR 8042
CNRS - Université d'Évry Val d'Essonne, Genopole
Tour Evry II, 523 place des terrasses de l'Agora, 91000 Évry, France

Abstract

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 (*e.g.* SDL, value-passing process algebras such as extensions of CSP or full-LOTOS, Korrigan) 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 with an expressive temporal gluing mechanism. This logic aims at providing an abstract denotational semantics for mixed specification languages. This logic enables one to reason about mixed specifications at a high level, without targeting a specific model or language. We show how our logic can be seen as an institution, which has the benefits of enabling a common refinement theory for mixed specification languages.

Key words: mixed formal specifications, temporal logic glue, denotational semantics, logic, institution, refinement.

1 Introduction

In the last few years, the need for a separation of concerns with reference to different aspects of systems to be specified (datatypes, behaviours, architectures, communications, real-time constraints, ...) appeared at both the programming and the specification level [1]. Systems are said to be mixed when they have more than one aspect. A lot of languages have been defined to deal with mixed specifications, but may be classified within two groups (see [2] for more references we cannot give here by lack of space). The homogeneous approach

[1] Email:poizat@lami.univ-evry.fr

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

(CASL-LTL, TLA) uses a single formalism (often extending it in a syntactical way) to specify both aspects. The heterogeneous approach (LOTOS, SDL, PSF) uses different formalisms. We think this last approach is well suited 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 followed in previous works on Korrigan [9,10,11], integrating algebraic specifications (for datatypes), Symbolic Transition Systems (STS, for behaviours and interfaces) and temporal logic (to glue components). However, we also think that the homogeneous approach is better suited (*i.e.* simpler) to the verification part of the specification process: no integration problems arise as the verification may be done in a unique framework.

We found out from our experience [3] 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 now to stay symbolic, that is 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 [5]. More recently, the previous works had to be modified and extended to deal with LOTOS specificities [8,7].

Both Architectural Description Languages [13] (ADL) and more recently aspect-oriented approaches [1] expressed the need for the description of system at a high abstraction level and for very expressive means to glue components (or aspects) altogether. An example of such an expressive glue is the temporal logic used in Korrigan to synchronize components within a configuration [10]. Such an explicit temporal logic glue is much more expressive than the implicit ones used in formal ADLs based on process algebras such as Wright for example (see the presentation of our gluing mechanisms below definition 3.6). However, Korrigan only has an operational semantics [9]. If operational semantics are better suited to lower level specifications, a denotational (logic-oriented) semantics is better suited to express such abstract and expressive gluing mechanisms, all the more if data are involved in the exchanges between components. A logical approach is also better suited to enable consistency checks for languages translations (*e.g.* translating Korrigan from/to SDL and LOTOS [11]) or for multi-languages specifications (*e.g.* a part of the system is given in LOTOS, another one in SDL).

In this paper, we present such a logic for mixed specification languages with expressive gluing mechanisms, and give a refinement theory for it. We herein address the theoretical aspects of this logic. More syntactical/expressiveness elements and examples can be found in [10] where we address the definition of components, patterns and architectures using Korrigan, the specification

language that led us in the design of this logic. The paper is structured as follows. Section 2 presents the single-component part of the logic and its denotational semantics. Then, in Section 3, we extend the logic to the specification of components and compositions of components. In Section 4 we propose an institution for our logic and present some fundamental results on refinement this institution enables. To end, Section 5 concludes and gives some perspectives.

2 Basic Mixed Specifications

In this Section, we address the specification of basic components, made up of a static part (datatypes) and a dynamic part (behaviour and communication).

2.1 Data Part

The data part addresses the functional issues of components. It will be described with a many-sorted first order logic. As usual, Σ-*terms*, noted $T_\Sigma(V)$, and Σ-*formulas*, noted $Sen(\Sigma)$, 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 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 $f^{\mathcal{M}} : M_{s_1} \times \cdots \times M_{s_n} \to M_s$ for each function name $f : s_1 \dots s_n \to s \in \mathcal{F}$ and a n-ary relation $r^{\mathcal{M}} : M_{s_1} \times \cdots \times M^{s_n}$ 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 total function $\sigma^\sharp : T_\Sigma(V) \to M$ defined as the canonical extension of any interpretation of variables $\sigma : V \to M$. 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 $\sigma : V \to M$, $\mathcal{M} \models_\sigma \varphi$.

2.2 Behaviour Part

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.

To instantiate the structures we give below, we will use examples written using a toy abstract language that has been defined for this purpose. Such a language can be seen as a generalization of languages such as LOTOS, SDL or Korrigan. We first define exchange profiles and dynamic profiles to be the generalization of the event typing information that is usual in languages such as LOTOS, SDL or their generalization in Korrigan [9]. *Exchange profiles* (over a set δS of dynamic sorts) are a typing of components with which the

component taken into account will communicate. Their form may be $\uparrow \delta s$ for a sending to a δs component, $\downarrow \delta s$ for a reception from a δs component or any combination of these. *Dynamic profiles* (over a set S of sorts and a subset δS of S) then take into account both exchange profiles and the typing of data exchanges. With η being an exchange profile, the form of dynamic profiles may be either $!s\eta$ for a sending of a value of sort s, $?s\eta$ for a reception of a value of sort s, or any combination of these. A more formal definition of dynamic profiles and exchange profiles may be found in [2].

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

Definition 2.1 (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 $((\delta S, \delta s), \mathcal{E}v)$ where:*

- *δS is a subset of S with a distinguished element, δs, called* sort of interest [2] *of $\delta\Sigma$,*
- *$\mathcal{E}v$ is a set of event names, each one equipped with a dynamic profile ρ over $(S, \delta S)$. Moreover, $\mathcal{E}v \cap (\mathcal{F} \cup R) = \emptyset$.*

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

Members of δS are *dynamic sorts*, and values of dynamic sorts are used as identifiers in our logic.

Example 2.2 We take as a running example a gas station. The station is made up of several tanks with an associated tank pump, and of several pumps equipped with a card reader and a pump manager. Several events may be observed: the user gives his/her card to the card reader, the user identifies himself/herself, the card reader returns a card, the user chooses the gas he/her wants (either unleaded 95, unleaded 98 or diesel), the card reader send the information of the chosen gas to the pump manager, the pump manager activates a tank pump. This will lead to the following definitions, taking $CardReader$ as the (dynamic) sort of interest:

- $(\delta S, \delta s) = (\{CardReader, PumpMgr, TankPump\}, CardReader)$
- $S = \{Card, Gas\} \cup \delta S$,
- $CardReader$ has a $the\_card :\!\!\rightarrow Card$ (static) operation
- $\mathcal{E}v = \{give : ?Card, identif : \varepsilon, return : !Card, choose : ?Gas, on\_duty : !Gas \uparrow PumpMgr\}$

Notation 1 *Let S be a set of sorts. Let A and B be two S-sets. We note B^A the set of partial functions from A to B compatible with S (i.e. for all $\nu \in B^A$, for all $s \in S$, for all $a \in A_s$, if $\nu(a)$ is defined then $\nu(a) \in B_s$). In*

[2] The sort of interest corresponds to the sort of the component (one would use the term class in the object-oriented paradigm) we are defining, other sorts being "used" sorts.

the following, we will note $\nu(a) \searrow$ *(respectively* $\nu(a) \nearrow$*) to mean that* $\nu(a)$ *is defined (respectively undefined).*

A semantics for dynamic signatures may be now given. Events are interpreted by binary relations between states, here interpretation of variables, *i.e.* elements of M^V.

Definition 2.3 (Dynamic model) *Let* $\delta\Sigma = ((\delta S, \delta s), \mathcal{E}v)$ *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* $e \in \mathcal{E}v$ *with a binary relation* $e^{\mathcal{D}}$ *on* M^V*.*

We now address the issue of defining some kind of *dynamic terms* and *dynamic formulas* in the same way static ones are defined from static signatures and variables. They are used to glue the static and dynamic aspects of components altogether, or to define (time) relations between events.

Exchange offers (respectively offers) will correspond to the well-typed "instantiation" of exchange profiles (respectively dynamic profiles). A *η-exchange offer* corresponds to the η exchange profile where sorts in reception have been instantiated by a variable and sorts in emission by a term. A *ρ-offer* corresponds to the ρ dynamic profile where sorts in reception have been instantiated by a variable, sorts in emission by a term and every η exchange profile by an η-exchange offer. A more formal definition of exchange offers and offers may be found in [2].

Example 2.4 With (static) sort Gas being defined as $\{Un95, Un98, Diesel\}$, and variable pm being of (dynamic) sort $PumpMgr$, then $!Un98 \uparrow pm$ is a possible offer for the *on_duty* event whose dynamic profile was $!Gas \uparrow PumpMgr$.

From the set of offers, we can build the set of transition terms. Their interpretation will be a binary relation on states. This comes from the fact that transition terms are used as transition formulas denoting sets of transitions. Such a set-theoretical interpretation is usual in temporal logics as it enables one to use usual set-theoretical operations (union, intersection, ...) which are syntactically denoted by propositional connectors and first-order quantifiers.

Definition 2.5 (Transition terms) *A* transition term *on a dynamic signature* $\delta\Sigma$ *is any well-formed formula built on* $e.\omega$ *with* $e : \rho \in \mathcal{E}v^*$ *and* ω *being a* $\rho-$*offer,* **true***, propositional connectives in* $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ *and first-order quantifiers in* $\{\forall, \exists\}$*. Let us note* $\mathcal{T}_{\delta\Sigma}(V)$ *the whole set of transition terms.*

Notation 2 (Read Variables ($\mathcal{RV}$) and Sent Terms ($\mathcal{ST}$)) *Let* ω *be an offer. We note* $\mathcal{RV}(\omega)$ *(respectively* $\mathcal{ST}(\omega)$*) the whole set of variables* x *(respectively terms* t*) such that* $?x$ *or* $\downarrow x$ *(respectively* $!t$ *or* $\uparrow t$*) occurs in* ω*.*

Definition 2.6 (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* $[\![\tau]\!]_{\mathcal{D}}$*, is the binary*

relation on M^V inductively defined on the structure of τ as follows:

$$(\sigma,\sigma') \in [\![e.\omega]\!]_{\mathcal{D}} \Longleftrightarrow \begin{cases} (\sigma,\sigma') \in e^{\mathcal{D}} \\ \textit{(states are related by the event interpretation)} \\ \wedge(\forall t \in \mathcal{ST}(\omega),\ \sigma^\sharp(t) \searrow) \\ \textit{(sent terms have an interpretation in the source state)} \\ \wedge(\forall x \in \mathcal{RV}(\omega),\ \sigma'(x) \searrow) \\ \textit{(received variables are bound in the target state)} \\ \wedge(\forall x \in V \setminus \mathcal{RV}(\omega),\ \sigma(x) \searrow \Rightarrow \sigma'(x) \searrow \wedge \sigma'(x) = \sigma(x)) \\ \textit{(the value of non received variables has not changed)} \end{cases}$$

$$[\![\mathbf{true}]\!]_{\mathcal{D}} = \bigcup_{e \in \mathcal{E}v} e^{\mathcal{D}}$$

$$[\![\forall x \tau]\!]_{\mathcal{D}} = \begin{cases} [\![\tau]\!]_{\mathcal{D}} & \textit{if } \forall v \in M, \exists \sigma' \in M^V,\ (\sigma,\sigma') \in [\![\tau]\!]_{\mathcal{D}} \wedge \sigma'(x) = v \\ \emptyset & \textit{otherwise} \end{cases}$$

$$[\![\exists x \tau]\!]_{\mathcal{D}} = \begin{cases} [\![\tau]\!]_{\mathcal{D}} & \textit{if } \exists v \in M, \exists \sigma' \in M^V,\ (\sigma,\sigma') \in [\![\tau]\!]_{\mathcal{D}} \wedge \sigma'(x) = v \\ \emptyset & \textit{otherwise} \end{cases}$$

Other operators may be treated in a usual way, that is $[\![\tau_1 \wedge \tau_2]\!]_{\mathcal{D}} = [\![\tau_1]\!]_{\mathcal{D}} \cap [\![\tau_2]\!]_{\mathcal{D}}$, $[\![\tau_1 \vee \tau_2]\!]_{\mathcal{D}} = [\![\tau_1]\!]_{\mathcal{D}} \cup [\![\tau_2]\!]_{\mathcal{D}}$, $[\![\tau_1 \Rightarrow \tau_2]\!]_{\mathcal{D}} = [\![\neg\tau_1]\!]_{\mathcal{D}} \cup [\![\tau_2]\!]_{\mathcal{D}}$, and $[\![\neg\tau]\!]_{\mathcal{D}} = [\![\mathbf{true}]\!]_{\mathcal{D}} \setminus [\![\tau]\!]_{\mathcal{D}}$.

To complete our logic for the behavioural aspects, we now have to define how dynamic properties may be expressed, that is dynamic formulas. These formulas correspond to some form of value-passing extension of HML. Such formulas exists in Korrigan [9] or, as far as LOTOS is concerned, in FULL [7].

Definition 2.7 (Dynamic formulas) *A* dynamic formula *on a dynamic signature $\delta\Sigma$ is any well-formed formula built on (static) Σ-formulas,* **true***, temporal connectives $[\tau]\varphi$ and $\langle\tau\rangle\varphi$ with τ being a transition term and φ being a dynamic formula, propositional connectives in $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ and first-order quantifiers in $\{\forall, \exists\}$. Let us note $\delta\mathcal{S}en(\delta\Sigma)$ the whole set of dynamic formulas.*

Definition 2.8 (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 $\varphi \in \delta\mathcal{S}en(\delta\Sigma)$. 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 $\mathcal{M} \models_\sigma \varphi$,*

- *for all* $\sigma \in M^V$, $\mathcal{D} \models_\sigma \mathbf{true}$,
- *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* $(\sigma, \sigma') \in [\![\tau]\!]_\mathcal{D}$, $\mathcal{D} \models_{\sigma'} \psi$,
- *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* $(\sigma, \sigma') \in [\![\tau]\!]_\mathcal{D}$, $\mathcal{D} \models_{\sigma'} \psi$,
- *propositional connectives and first-order quantifiers are handled as usual.*

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

Example 2.9 We will take events in the card reader as examples. Dynamic formulas may be used both to relate dynamic events with static operations

- after the user gave a card c, the *the_card* operation yields c:
  $\forall c : Card\ [give?c]\ the\_card = c$

or dynamic events with dynamic events

- after the user identifies himself/herself, the card reader either (nondeterminism) returns the card or asks for the gas choice:
  $[identif](\langle return!the\_card\rangle \mathbf{true} \ \vee\ \forall gas : Gas \langle choose?gas\rangle \mathbf{true})$

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

Definition 2.10 (View) *Given a many-sorted first order signature* $\Sigma = (S, \mathcal{F}, R)$*, a* Σ-view $\mathcal{V}$ *is a 4-tuple* $(\delta\Sigma, \mathcal{A}\mathbf{x}, \mathcal{S}\mathbf{tate}, \mathcal{I}\mathbf{nit})$*, where* $\delta\Sigma$ *is a dynamic signature over* Σ*,* $\mathcal{A}\mathbf{x}$ *is a set of* Σ*-formulas, and* $\mathcal{S}\mathbf{tate}$ *and* $\mathcal{I}\mathbf{nit}$ *are sets of dynamic formulas.*

Definition 2.11 (View model) *Let* $\mathcal{V} = (\delta\Sigma, \mathcal{A}\mathbf{x}, \mathcal{S}\mathbf{tate}, \mathcal{I}\mathbf{nit})$ *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:*

- *for all* $(\sigma, \sigma') \in e^\mathcal{D}$, $\sigma <^\mathcal{D} \sigma'$,
- $M \models \mathcal{A}\mathbf{x}$,
- $\mathcal{D} \models \mathcal{S}\mathbf{tate}$,
- *for all* $\sigma \in M^V$*, if* $\mathcal{D} \models_\sigma \mathcal{I}\mathbf{nit}$ *then for all* $\sigma' \in M^V$, $\sigma <^\mathcal{D} \sigma'$.

The pair $(\mathcal{D}, <^\mathcal{D})$ 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. Therefore, the semantics of view specifications can be seen as a possible worlds semantics. The satisfaction of formulas depends on states (*i.e.* partial functions of M^V) in which they are evaluated (they are contingent).

3 Composed Mixed Specifications

This part addresses the composition, architectural, synchronizing and inter-component communication aspects of systems. As for the basic mixed spec-

ifications, we here define first composed signatures, then composed formulas and finally composed specifications. Here we have no need for any composed equivalent of terms. Throughout this Section, we will use the $\widetilde{.}$ symbol to denote concepts related to composed specifications (*e.g.* Σ was a signature in the previous Section, $\widetilde{\Sigma}$ will be a composed signature in this Section).

Notation 3 *Given a view* $\mathcal{V} = ((\delta\Sigma, \delta s), \mathcal{A}\mathbf{x}, \mathcal{S}\mathbf{tate}, \mathcal{I}\mathbf{nit})$ *with* $\delta\Sigma = (\delta S, \mathcal{E}v)$ *and* $\Sigma = (S, \mathcal{F}, R)$*, we will use the following notations:*

- $\delta\mathcal{S}ig[\mathcal{V}] = \delta\Sigma$
- $\delta\mathcal{S}ort[\mathcal{V}] = \delta S$
- $\delta int[\mathcal{V}] = \delta s$
- $\mathcal{S}ig[\mathcal{V}] = \Sigma$
- $\mathcal{R}el[\mathcal{S}ig[\mathcal{V}]] = R$
- $\mathcal{F}un[\mathcal{S}ig[\mathcal{V}]] = \mathcal{F}$

Definition 3.1 (Composed mixed signature) *A* composed mixed signature $\widetilde{\Sigma}$ *is a finite set* $\{\mathcal{V}_1, \ldots, \mathcal{V}_n / n \in \mathbb{N}\}$ *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,* $\widetilde{\Sigma}$ *satisfies for any couple of views* $(\mathcal{V}_i, \mathcal{V}_j)$ *of* $\widetilde{\Sigma}$ *the following conditions:*

- $\delta int[\mathcal{V}_i] = \delta int[\mathcal{V}_j] \Rightarrow \mathcal{V}_i = \mathcal{V}_j$ *(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:*
  - $\forall \alpha \in S^*_{i,j}, \forall s \in S_{i,j}, \forall f : \alpha \to s,\ f \in \mathcal{F}_i \Leftrightarrow f \in \mathcal{F}_j$,
  - $\forall \alpha \in S^*_{i,j}, \forall r : \alpha,\ r \in R_i \Leftrightarrow r \in R_j$.

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

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

We first define the models for composed mixed signatures, and then global states and environments for these models (to deal with the gluing between the individual models).

Definition 3.2 (Composed mixed model) *Let* $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ *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 ...),*
- $\forall 1 \leq i, j \leq n$:
  - $\forall f \in F_i \cap F_j,\ f^{\mathcal{M}_i} = f^{\mathcal{M}_j}$,
  - $\forall r \in R_i \cap R_j,\ r^{\mathcal{M}_i} = r^{\mathcal{M}_j}$,
  - $\forall e \in \mathcal{E}v_i \cap \mathcal{E}v_j,\ e^{\mathcal{D}_i} = e^{\mathcal{D}_j}$.

  (... and same interpretations for their functions, predicates and events[3] *)*

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

Composed mixed states denote functions that, taking the identification of a component, yield the substitutions for this component's states (*i.e.* within

[3] Overloading can easily be achieved through renaming.

this component's model).

Definition 3.3 (Composed mixed state) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be a composed mixed signature. Let $\widetilde{V}$ be a $\widetilde{S}$-indexed family of sets of variables. Let $\widetilde{\mathcal{D}}$ be a $\widetilde{\Sigma}$-model. For every $1 \leq i \leq n$, let us note $\widetilde{V}_{S_i}$ the restriction of $\widetilde{V}$ to S_i. A* composed mixed state *over $\widetilde{\mathcal{D}}$ is any $\delta\widetilde{S}$-indexed family of functions $\widetilde{\gamma}_{\delta s_i} : \widetilde{M}_{\delta S_i} \rightarrow M_i^{\widetilde{V}_{S_i}}$ such that $\forall 1 \leq i, j \leq n, \forall n \in \widetilde{M}_{\delta S_i} \cap \widetilde{M}_{\delta S_j}, \widetilde{\gamma}_{\delta S_i}(n) = \widetilde{\gamma}_{\delta S_j}(n)$. Let us note $\mathcal{S}t[\widetilde{\mathcal{D}}]$ the set of composed mixed states over $\widetilde{\mathcal{D}}$.*

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

Definition 3.4 (Environment) *With all the notations of Definition 3.3, an* environment *over $\widetilde{\mathcal{D}}$ is a pair $\mathcal{E} = (\widetilde{\sigma}, \widetilde{\gamma})$ where $\widetilde{\sigma} : \widetilde{V}_{\widetilde{\delta S}} \rightarrow \widetilde{M}_{\widetilde{\delta S}}$ is a function compatible with sorts (i.e. $\widetilde{\sigma}(\widetilde{V}_{\delta s}) \subseteq \widetilde{M}_{\delta s}$) and $\widetilde{\gamma} \in \mathcal{S}t[\widetilde{\mathcal{D}}]$ such that:*

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

We now define the way to express properties on our composed specifications. Projected formulas and projected state formulas enable one to express properties of the systems in terms of properties on the elements that compose it. They are used to restrict the set of models of composed mixed signatures.

Definition 3.5 (Projected state formula) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be a composed mixed signature. Let $\widetilde{V}$ be a $\widetilde{S}$-indexed family of sets. A* projected state formula *on $\widetilde{\Sigma}$ is any well-formed formula built on:*

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

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

Definition 3.6 (Satisfaction of projected state formulas) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be a composed mixed signature. Let $\widetilde{\varphi} \in \mathcal{P}\delta\mathcal{S}en(\Sigma)$ be a projected state*

formula. Let $\widetilde{\mathcal{D}}$ be a $\widetilde{\Sigma}$-model. Let $\mathcal{E} = (\widetilde{\sigma}, \widetilde{\gamma})$ be an environment. Let us note $\widetilde{\delta V}$ the restriction of $\widetilde{V}$ to $\widetilde{S}$. $\widetilde{\mathcal{D}}$ satisfies $\widetilde{\varphi}$ for $\mathcal{E}$, noted $\widetilde{\mathcal{D}} \models_{\mathcal{E}} \widetilde{\varphi}$, if and only if:

- *if $\widetilde{\varphi}$ is of the form $x.\psi$ with $x \in \widetilde{V_{\delta s_i}}$ then $\widetilde{\mathcal{D}} \models_{\mathcal{E}} \widetilde{\varphi}$ if and only if $\mathcal{D}_i \models_{\widetilde{\gamma}(\widetilde{\sigma}(x))} \psi$,*
- *if $\widetilde{\varphi}$ is of the form $\forall x.\widetilde{\psi}$ then $\widetilde{\mathcal{D}} \models_{\mathcal{E}} \widetilde{\varphi}$ if and only if for any environment $\mathcal{E}'$ such that for all $y \neq x \in \widetilde{\delta V}, \widetilde{\sigma}(y) = \widetilde{\sigma'}(y)$, $\widetilde{\mathcal{D}} \models_{\mathcal{E}'} \widetilde{\psi}$.*
- *if $\widetilde{\varphi}$ is of the form $\exists x.\widetilde{\psi}$ then $\widetilde{\mathcal{D}} \models_{\mathcal{E}} \widetilde{\varphi}$ if and only if there exists an environment $\mathcal{E}'$ such that for all $y \neq x \in \widetilde{\delta V}, \widetilde{\sigma}(y) = \widetilde{\sigma'}(y)$, $\widetilde{\mathcal{D}} \models_{\mathcal{E}'} \widetilde{\psi}$.*

Propositional connectives and first-order quantifiers are handled as usual.

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

These formulas are a very expressive (temporal) gluing mechanism and generalize more implicit synchronizing glues of languages such as CCS, CSP, Wright or LOTOS. Let us take some examples (here CCS and LOTOS).

In CCS events are synchronized pairwise (event a with event $\overline{a}$) in a 1-to-1 fashion. This yields a non observable event, τ, which may not be synchronized anymore. LOTOS is more expressive, taking data exchange and synchronizing with value agreement into account. In LOTOS one also explicitly states which events are synchronized and which ones are not. For example, if one has two processes $P = a; stop$ and $Q = a; stop$ (both do a and stop), then (s)he may either state that they synchronize on a ($P|[a]|Q$), or not ($P|||Q$). Finally, unless explicitly hidden, events in synchronizings may be synchronized again (in $(P|[a]|Q)|[a]|Q$, the three processes synchronize on a)

Our formulas generalize and extend these gluing mechanisms. In languages which may be based on our logic, we can express things such as [4]:

- (explicit) synchronizing: $c_1.[a]\mathbf{true} \Leftrightarrow c_2.[a]\mathbf{true}$ (c_1 and c_2 must synchronize on a)
- connecting event ports with different names: $c_1.[a_1]\mathbf{true} \Leftrightarrow c_2.[a_2]\mathbf{true}$
- value passing: $\forall x : Nat(c_1.[a?x]\mathbf{true} \Leftrightarrow c_2.[a!x]\mathbf{true})$
- broadcasting (1-to-N, all in the same time): $server.[send]\mathbf{true} \Rightarrow \mathbf{ALL}(client.i : [1..N]).[receive]\mathbf{true}$
- exclusive peer-to-peer (1-with-N, but one at a time): $server.[send]\mathbf{true} \Rightarrow \mathbf{ONE}(client.i : [1..N]).[receive]\mathbf{true}$
- exclusive states: $\neg(cooler_1.(active = true) \wedge cooler_2.(active = true))$, both coolers are not active at the same time

Korrigan, which is an example of such a language, has a graphical notation for components and compositions taking these formulas into account [10].

[4] With $\mathbf{ALL}(S).\widetilde{\psi}$ being defined as $\bigwedge_{x \in S} x.\widetilde{\psi}$ (set/range quantification) and $\mathbf{ONE}(S).\widetilde{\psi}$ as $\bigvee_{x \in S}(x.\widetilde{\psi} \bigwedge_{y \in S, x \neq y} \neg y.\widetilde{\psi})$ (one and only one in a set/range).

Definition 3.7 (Composed mixed specification) *A* composed mixed specification $\widetilde{\mathcal{C}}$ *is a 4-tuple* $(\widetilde{\Sigma}, \widetilde{\mathcal{A}\mathbf{x}}, \widetilde{\mathcal{S}\mathbf{tate}}, \widetilde{\mathcal{I}\mathbf{nit}})$*, where* $\widetilde{\Sigma}$ *is a composed mixed signature,* $\widetilde{\mathcal{A}\mathbf{x}}$ *is a set of projected state formulas, and* $\widetilde{\mathcal{S}\mathbf{tate}}$ *and* $\widetilde{\mathcal{I}\mathbf{nit}}$ *are sets of projected formulas.*

Example 3.8 We suppose here that within the composed mixed specification of the system, the card reader is identified with *cr*, the pump manager with *pm* and the tank pumps with $un95\_tp$, $un98\_tp$ and $diesel\_tp$. Using projected term formulas, we may now express that when the card reader sends the gas choice to the pump manager, the pump manager activates the correct pump: $cr.[on\_duty!Un98 \uparrow pm]\mathbf{true} \Rightarrow pm.[activate \uparrow un98\_tp]\mathbf{true}$

Notation 4 *Let* $\widetilde{\mathcal{D}}$ *be a* $\widetilde{\Sigma}$*-model. Let us define the pre-order on* $\mathcal{S}t[\widetilde{\mathcal{D}}]$ *as follows:* $\widetilde{\gamma} \widetilde{<} \widetilde{\gamma}' \Leftrightarrow (\forall 1 \leq i \leq n, \forall m \in \widetilde{M_{\delta S_i}}, \widetilde{\gamma}(m) <^{\mathcal{D}_i} \widetilde{\gamma}'(m))$

Definition 3.9 (Models of specification) *Let* $\widetilde{C} = (\widetilde{\Sigma}, \widetilde{\mathcal{A}\mathbf{x}}, \widetilde{\mathcal{S}\mathbf{tate}}, \widetilde{\mathcal{I}\mathbf{nit}})$ *be a composed mixed specification. A* $\widetilde{C}$-model *is a* $\widetilde{\Sigma}$*-model* $\widetilde{\mathcal{D}}$ *such that:*

- $\widetilde{\mathcal{D}} \models \widetilde{\mathcal{A}\mathbf{x}} \cup \widetilde{\mathcal{S}\mathbf{tate}}$,
- $\forall \widetilde{\gamma} \in \mathcal{S}t[\widetilde{\mathcal{D}}]$, $(\forall \widetilde{\sigma}, \widetilde{\mathcal{D}} \models_{(\widetilde{\sigma}, \widetilde{\gamma})} \widetilde{\mathcal{I}\mathbf{nit}}) \Rightarrow (\forall \widetilde{\gamma}' \in \mathcal{S}t[\widetilde{\mathcal{D}}], \widetilde{\gamma} \widetilde{<} \widetilde{\gamma}')$.

In this Section we have defined a logic which enables one to reason on concrete mixed specification languages (*e.g.* Korrigan, LOTOS, SDL) in an abstract way (defining a denotational semantics for them). This logic could also be used to deal with multi-languages issues as long as the languages are defined using it. In the next Section we will see that another benefit of this logic is to be granted with a generic refinement theory for such languages.

4 Institution and Refinement

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 generalize some well-known classical results. Since J. Goguen and R. Burstall's works on institutions [12], it has been established that such results can be generalized at a meta-level. Showing that a formalism is an institution allows its authors to directly use general results such as existence of quotients, free models, amalgamation properties underlying to modularity issues, and refinement issues.

Here, we are going to show that our mixed specification logic is an institution in order to define a refinement theory for views. We will show that results on horizontal and vertical refinements for institutions hold for composed mixed specifications. For this purpose, we will use notations and results established

in [6] and [14]. By lack of place, we do not give proofs nor definitions of category theory (used in the definition of institutions) in this paper. The reader interested in proofs can find them in [2]. A simple presentation of category theory can be found in [15].

4.1 An Institution for the Mixed Specification Logic

Institutions formally axiomatize the notion of logical system from a theoretical point of view. An institution is a quadruple $(Sig, Sen, Mod, \models)$ where Sig is a category (a class of objects and morphisms between these objects) of *signatures*, $Sen : Sig \to Set$ is functor (a couple of mappings, one between classes and one between morphisms) which maps every signature to its set of sentences, $Mod : Sig \to Cat$ 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 $\sigma : \Sigma \to \Sigma'$, $Mod(\sigma) : Mod(\Sigma') \to Mod(\Sigma)$ is called *reduct functor*. Moreover, this quadruple satisfies the following property, the so-called *satisfaction condition*: $\forall(\sigma : \Sigma \to \Sigma') \in Hom_{Sig}$, $\forall \mathcal{M} \in |Mod(\Sigma')|$, $\forall \varphi \in |Sen(\Sigma)|$, $\mathcal{M} \models_{\Sigma'} Sen(\sigma)(\varphi) \Leftrightarrow Mod(\sigma)(\mathcal{M}) \models_\Sigma \varphi$. In such a generic framework, a specification is any couple $SP = (\Sigma, Ax)$ where $\Sigma \in |Sig|$ and $Ax \subseteq Sen(\Sigma)$. The notations $\mathcal{M} \models_\Sigma \varphi$ is extended in the usual way to categories of models and sets of formulas. Given a specification $SP = (\Sigma, Ax)$, the category of Σ-models $\mathcal{M}$ such that $\mathcal{M} \models_\Sigma Ax$ is noted $Mod(SP)$. A Σ-sentence φ is a *semantic consequence* of a specification SP if and only if for every $\mathcal{M} \in |Mod(SP)|$, $\mathcal{M} \models \varphi$.

4.1.1 The category of composed mixed models.

First, we must define an appropriate morphism notion between view models.

Notation 5 *For any interpretation of variables $\sigma : V \to M$, we note $\mu(\sigma) : V \to M'$ the application defined by $x \mapsto \mu(\sigma(x))$. Finally, with R being a binary relation on M^V, we note $\mu(R)$ the set $\{(\mu(\sigma), \mu(\sigma')) \mid (\sigma, \sigma') \in R\}$.*

Definition 4.1 (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 $\mu : \mathcal{M} \to \mathcal{M}'$ such that:*

- *$\mu(e^{\mathcal{D}}) \subseteq e^{\mathcal{D}'}$ (event compatibility)*
- *$\mu(<^{\mathcal{D}}) \subseteq <^{\mathcal{D}'}$ (pre-order compatibility)*

We then extend it to composed mixed models.

Definition 4.2 (Composed mixed model morphisms) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be a composed mixed signature. A $\widetilde{\Sigma}$-*morphism *between two $\widetilde{\Sigma}$-models $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{D}'}$ is defined for each $1 \leq i \leq n$ by a $\mathcal{V}_i$-morphism $\mu_i : \mathcal{D}_i \to \mathcal{D}'_i$. 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, $\widetilde{\Sigma}$-models and $\widetilde{\Sigma}$-morphisms form a category. Let us note it $Mod(\widetilde{\Sigma})$.

4.1.2 Reduct functor and satisfaction condition.
An essential ingredient which is missing is an appropriate morphism notion for composed mixed signatures. We first have to define morphisms between dynamic signatures and morphisms between views.

Definition 4.3 (Dynamic signature morphism) *Let $\delta\Sigma = ((\delta S, \delta s), \mathcal{E}v)$ and $\delta\Sigma' = ((\delta S', \delta s'), \mathcal{E}v')$ be two dynamic signatures. A* dynamic signature morphism *$\nu : \delta\Sigma \to \delta\Sigma'$ is a signature morphism $\nu : \Sigma \to \Sigma'$ 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 4.4 (View morphism) *Let $\mathcal{V} = (\delta\Sigma, \mathcal{A}x, \mathcal{S}tate, \mathcal{I}nit)$ and $\mathcal{V}' = (\delta\Sigma', \mathcal{A}x', \mathcal{S}tate', \mathcal{I}nit')$ be two views. A* view morphism *$\nu : \mathcal{V} \to \mathcal{V}'$ is a dynamic signature morphism $\nu : \delta\Sigma \to \delta\Sigma'$ such that $\nu(\mathcal{A}x) \subseteq \mathcal{A}x'$, $\nu(\mathcal{S}tate) \subseteq \mathcal{S}tate'$ and $\nu(\mathcal{I}nit) \subseteq \mathcal{I}nit$.*

Given a a view morphism ν, let us note $\overline{\nu}$ its canonical extension to dynamic formulas. We may now define morphisms between composed mixed signatures.

Definition 4.5 (Composed mixed signature morphisms) *Let $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ and $\widetilde{\Sigma}' = \{\mathcal{V}'_1, \ldots, \mathcal{V}'_m\}$ be two composed mixed signatures. A* Composed mixed signature morphism *$\widetilde{\nu}$ is a set $\{\nu_1, \ldots, \nu_n\}$ such that:*

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

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

Proposition 4.6 *Let $\nu : \mathcal{V} \to \mathcal{V}'$ be a view morphism. Let $(\mathcal{D}', <^{\mathcal{D}'})$ be a $\mathcal{V}$-model. Let us note ${\mathcal{D}'}_{\restriction\nu}$ the dynamic-model $\mathcal{D}$ for $\delta\Sigma$ defined as follows:*

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

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

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

Corollary 4.7 *With all the notations of Proposition 4.6, Let us note $<^{{\mathcal{D}'}_{\restriction\nu}}$ the pre-order on M^V defined by: $<^{{\mathcal{D}'}_{\restriction\nu}} = \{(\sigma_{\restriction\nu}, \sigma'_{\restriction\nu}) \mid (\sigma, \sigma') \in <^{\mathcal{D}'}\}$. Then, the couple $({\mathcal{D}'}_{\restriction\nu}, <^{{\mathcal{D}'}_{\restriction\nu}})$ is a $\mathcal{V}'$-model.*

Proposition 4.6 is the satisfaction condition for the view specification logic.

Definition 4.8 (Reduct functor) *Let $\widetilde{\nu} : \widetilde{\Sigma} \to \widetilde{\Sigma}'$ be a composed mixed sig-*

nature morphism. Let $\widetilde{\mathcal{D}'}$ *be a* $\widetilde{\Sigma}'$*-model. The* reduct functor $_{-|\widetilde{\nu}} : Mod(\widetilde{\Sigma}') \to Mod(\widetilde{\Sigma})$ *is defined as follows:*

- *for each* $\widetilde{\Sigma}'$*-model* $\widetilde{\mathcal{D}'}$, $\widetilde{\mathcal{D}'}_{|\widetilde{\nu}}$ *is the* $\widetilde{\Sigma}$*-model* $\widetilde{\mathcal{D}}$ *where for every* $1 \leq i \leq n$, $\mathcal{D}_i = (\mathcal{D}'_i)_{|\nu_i}$ *and* $<^{\mathcal{D}_i} = (<^{\mathcal{D}'_i})_{|\nu_i}$ *(by Corollary 4.7, the couple* $(\mathcal{D}_i, <^{\mathcal{D}_i})$ *is a* $\mathcal{V}_i$*-model).*
- *for each* $\widetilde{\Sigma}'$*-morphism* $\widetilde{\mu} : \widetilde{\mathcal{D}} \to \widetilde{\mathcal{D}'}$, $\widetilde{\mu}_{|\widetilde{\nu}} : \widetilde{\mathcal{D}}_{|\widetilde{\nu}} \to \widetilde{\mathcal{D}'}_{|\widetilde{\nu}}$ *is the* $\widetilde{\Sigma}$*-morphism defined for all* $1 \leq i \leq n$ *by the* $\mathcal{V}_i$*-morphism* $(\mu_i)_{|\nu_i}$ *where* μ_i *is the corresponding* $\mathcal{V}'_i$*-morphism of* $\widetilde{\mu}$*.*

Theorem 4.9 (Satisfaction condition) *Let* $\widetilde{\nu} : \widetilde{\Sigma} \to \widetilde{\Sigma}'$ *be a composed mixed signature morphism. Let* $\widetilde{\mathcal{D}'}$ *be a* $\widetilde{\Sigma}'$*-model. Let* $\widetilde{\varphi}$ *be projected formula or a projected state formula over* $\widetilde{\Sigma}$*. Then, we have:* $\widetilde{\mathcal{D}'} \models \overline{\widetilde{\nu}}(\widetilde{\varphi}) \Longleftrightarrow \widetilde{\mathcal{D}'}_{|\widetilde{\nu}} \models \widetilde{\varphi}$.

Corollary 4.10 *Let* $\widetilde{C} = (\widetilde{\Sigma}, \widetilde{\mathcal{A}x}, \widetilde{\mathcal{S}tate}, \widetilde{\mathcal{I}nit})$ *and* $\widetilde{C'} = (\widetilde{\Sigma}', \widetilde{\mathcal{A}x'}, \widetilde{\mathcal{S}tate'}, \widetilde{\mathcal{I}nit'})$ *be two composed mixed specifications such that there is a signature morphism* $\widetilde{\nu} : \widetilde{\Sigma} \to \widetilde{\Sigma}'$, $\overline{\widetilde{\nu}}(\widetilde{\mathcal{A}x}) \subseteq \widetilde{\mathcal{A}x'}$, $\overline{\widetilde{\nu}}(\widetilde{\mathcal{S}tate}) \subseteq \widetilde{\mathcal{S}tate'}$, *and* $\overline{\widetilde{\nu}}(\widetilde{\mathcal{I}nit}) \subseteq \widetilde{\mathcal{I}nit'}$*. Then, the reduct functor* $_{-|\widetilde{\nu}}$ *can be co-restricted to* $_{-|\widetilde{\nu}} : Mod(\widetilde{C'}) \to Mod(\widetilde{C})$*.*

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 4.11 *Let* $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ *be composed mixed signature. A* well-formed $\widetilde{\Sigma}$-formula *is:*

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

At last, all this enables us to define an institution for our composed mixed specifications.

Theorem 4.12 (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* $Sen_{CMS} : Sig_{CMS} \to Set$ *maps:*
  - *each composed mixed signature* $\widetilde{\Sigma}$ *to the set of well-formed* $\widetilde{\Sigma}$*-formulas (see Definition 4.11) and*
  - *each composed mixed signature morphism* $\widetilde{\nu}$ *to* $\overline{\widetilde{\nu}}$*.*
- *The contravariant functor* $Mod_{CMS} : Sig_{CMS} \to Cat$ *maps:*
  - *each composed mixed signature* $\widetilde{\Sigma}$ *to the category* $Mod(\widetilde{\Sigma})$ *and*
  - *each composed mixed signature morphism* $\widetilde{\sigma}$ *to the reduct functor* $_{-|\widetilde{\sigma}}$*.*
- $\models_{CMS} = (\models_{\widetilde{\Sigma}})_{\widetilde{\Sigma} \in |Sig_{CMS}|}$ *where for each composed system signature* $\widetilde{\Sigma} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$, $\models_{\widetilde{\Sigma}}$ *is the satisfaction relation of a well-formed* $\widetilde{\Sigma}$*-formula*

Γ *by a* $\widetilde{\Sigma}$*-model* $\widetilde{\mathcal{D}}$ *defined as follows:*
- *if* Γ *is of the form* $(\widetilde{\Sigma}, \widetilde{\varphi})$ *then* $\widetilde{\mathcal{D}} \models_{\widetilde{\Sigma}} \Gamma$ *iff* $\widetilde{\mathcal{D}} \models \widetilde{\varphi}$ *(see Definition 3.6);*
- *if* Γ *is of the form* $(\mathcal{V}_i, \varphi)$ *then* $\widetilde{\mathcal{D}} \models_{\widetilde{\Sigma}} \Gamma$ *iff* $\mathcal{D}_i \models \varphi$ *(see Definition 2.8).*

By Proposition 4.6, we can easily define, using the same arguments than for Theorem 4.12, the institution for view specifications. Let us note it Ins_{view}.

4.2 Views Refinement

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

Definition 4.13 (View refinement) *A view* $\mathcal{V}_{impl}$ *is a* refinement *of a view* $\mathcal{V}$ *if and only if* $\mathcal{S}ig[\mathcal{V}_{impl}] = \mathcal{S}ig[\mathcal{V}]$. *Let us note* $\mathcal{V} \rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]} \mathcal{V}_{impl}$ *such a refinement.*

A refinement is correct provided that the behaviour of the implementation is indistinguishable from the behaviour of the higher level specification under consideration. When dealing with loose semantics, since design choices can be made through refinement steps, the refinement semantics consists on cutting down the class of specification models. This is expressed as follows:

Definition 4.14 (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 $\mathcal{S}ig[\mathcal{V}_1] = \mathcal{S}ig[\mathcal{V}_2]$,
	$\mathcal{V}_1 \cup \mathcal{V}_2$ 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 $\nu : \mathcal{S}ig[\mathcal{V}] \to \Sigma'$,
	translate $\mathcal{V}$ **by** ν is a view with semantics
	$Mod(\textbf{translate } \mathcal{V} \textbf{ by } \nu) = \{(\mathcal{D}', <^{\mathcal{D}'}) \mid (\mathcal{D}'_{\restriction \nu}, (<^{\mathcal{D}'})_{\restriction \nu}) \in Mod(\mathcal{V})\}$
derive:	for any view $\mathcal{V}'$ and any signature morphism $\nu : \Sigma \to \mathcal{S}ig[\mathcal{V}']$,
	derive from $\mathcal{V}'$ **by** ν is a view with semantics
	$Mod(\textbf{derive from } \mathcal{V}' \textbf{ by } \nu) = Mod(\mathcal{V}')_{\restriction \nu}$

Definition 4.15 (Conservative extension along ν) *With all the notations of Definition 4.14,* $\mathcal{V}'$ *is a* conservative extension *of* $\mathcal{V}$ along ν *if and only if* $Mod(\mathcal{V}')_{\restriction \nu} = Mod(\mathcal{V})$.

From there, we can instantiate the institution independent proof system of [6] and obtain the following rules:

Definition 4.16 (Rules) *Let $\mathcal{V} = (\delta\mathcal{S}ig[\mathcal{V}], \mathcal{A}x, \mathcal{S}tate, \mathcal{I}nit)$. Let iso be a signature isomorphism. Let ν be a signature morphism. The family of refinement relations $(\rightsquigarrow_\Sigma)_{\Sigma\in|\mathcal{S}ig|}$ is defined by the following set of rules:*

$$(Basic)\frac{\mathcal{V}\models\mathcal{A}x'\cup\mathcal{S}tate'}{(\mathcal{S}ig[\mathcal{V}],\mathcal{A}x',\mathcal{S}tate',\mathcal{I}nit)\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}}\ \mathcal{V}=(\delta\mathcal{S}ig[\mathcal{V}],\mathcal{A}x,\mathcal{S}tate,\mathcal{I}nit)$$

$$(Sum)\frac{\mathcal{V}_1\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}\quad\mathcal{V}_2\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}}{\mathcal{V}_1\cup\mathcal{V}_2\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}}\qquad(Trans_1)\frac{\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\textbf{translate } \mathcal{V}' \textbf{ by } iso^{-1}}{\textbf{translate } \mathcal{V} \textbf{ by } iso\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}']}\mathcal{V}'}$$

$$(Trans_2)\frac{\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\textbf{derive from } \mathcal{V}' \textbf{ by } \nu}{\textbf{translate } \mathcal{V} \textbf{ by } \nu\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}']}\mathcal{V}'}$$

$$(Derive)\frac{\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}'']}\mathcal{V}''}{\textbf{derive from } \mathcal{V}' \textbf{ by } \nu\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}']}\mathcal{V}'}\ \begin{array}{l}\mathcal{V}''\text{ is a conservative extension}\\ \text{of }\mathcal{V}'\text{ along }\nu\end{array}$$

$$(Trans-equiv)\frac{\textbf{translate } (\textbf{translate } \mathcal{V} \textbf{ by } iso) \textbf{ by } \nu\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}'']}\mathcal{V}''}{\textbf{translate } \mathcal{V} \textbf{ by } \nu\circ iso\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}'']}\mathcal{V}''}$$

Theorem 4.17 (Soundness and completeness) *For any views $\mathcal{V}$ and $\mathcal{V}'$, we have: $\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}' \iff \mathcal{V}\Vdash_{\mathcal{S}ig[\mathcal{V}]}\mathcal{V}'$.*

4.2.1 Composition.

Of course, it is not reasonable to implement a specification as a whole in a single step. Complex systems usually require many refinement steps before obtaining efficient programs. This leads to the notion of sequential composition of implementation steps. Usually, the composition of enrichments is divided into two separate concepts: 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 and denotes the transitive closure of refinement relations.

Notation 6 *Let $\widetilde{C}$ be a composed mixed specification. Let $\mathcal{V}$ be a view belonging to $\widetilde{C}$ and let $\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}_{impl}]}\mathcal{V}_{impl}$. Let us note $\widetilde{C}[\mathcal{V}/\mathcal{V}_{impl}]$ the composed mixed specification obtained from $\widetilde{C}$ by substituting $\mathcal{V}$ by $\mathcal{V}_{impl}$.*

Theorem 4.18 (Horizontal refinement) *With all the previous notations, if $\mathcal{V}\rightsquigarrow_{\mathcal{S}ig[\mathcal{V}_{impl}]}\mathcal{V}_{impl}$, then: $\forall\widetilde{\varphi}\in\mathcal{S}en(\widetilde{\Sigma}),\ \widetilde{C}\models\widetilde{\varphi}\Longrightarrow\widetilde{C}[\mathcal{V}/\mathcal{V}_{impl}]\models\widetilde{\varphi}$.*

Corollary 4.19 *$Mod(\widetilde{C}[\mathcal{V}/\mathcal{V}_{impl}])_{|\widetilde{\nu}}\subseteq Mod(\widetilde{C})$.*

Theorem 4.18 means that as soon as composed mixed specifications are structured into view specifications, these view specifications can be refined by any correct realization (according the meaning given here) independently of each other. Thus, we gain an incremental method.

Theorem 4.20 (Vertical composition) *The following rule is sound:*

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

In this Section we have established an institution vision of our logic and we have shown that a refinement theory could be defined for it, in a generic way (hence non language specific). This should help in defining, in the future, generic theories for mixed specification languages such as the STS one which was firstly defined for value passing CSP [5] and had to be modified and extended for LOTOS [8,7].

5 Conclusion

In this paper we presented a logic and denotational model for mixed specifications. We first presented a logic for simple mixed systems and then extended it to take into account the specification of components and composition of components. Our logic aims at describing systems at a high level of abstraction (describe the "what" and not the "how"). For this purpose, it uses implicit states and dynamic observations of events. Such a high-level description of systems is also advocated in Architectural Description Languages (ADL). Indeed, we are investigating the use of Korrigan as a formal ADL. One of its interesting features is its very expressive synchronizing mechanisms (gluing components in architectures using temporal logic). However, Korrigan had no denotational semantics. This issue is now dealt with by our logic which may be used to give an abstract denotational semantics to other languages that do not have one, such as LOTOS. Moreover, we have shown that our logic could be seen as an institution, which has the benefits of enabling a refinement theory for mixed specification languages mapped into our logic.

A track of research is actually devoted to the study of symbolic models (STS). 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 proposals done specifically for CSP with value passing or for LOTOS may be generalized for a class of mixed specification languages such as Korrigan and our generic framework proposal for the integration of formal datatypes within state diagrams [4]. 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.

References

[1] *Aspect-Oriented Software Development*, `http://www.aosd.net/`.

[2] Aiguier, M., F. Barbier and P. Poizat, *A Logic for Mixed Specifications*, Technical Report 73-2002, LaMI (2002),

http://www.lami.univ-evry.fr/~poizat/documents/publications/RR-ABP02.ps.gz.

[3] Allemand, M., C. Attiogbe, P. Poizat, J.-C. Royer and G. 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.

[4] Attiogbé, C., P. Poizat and G. Salaün, *Integration of Formal Datatypes within State Diagrams*, in: M. Pezzè, editor, *Fundamental Approaches to Software Engineering (FASE'2003)*, Lecture Notes in Computer Science **2621** (2003), pp. 341–355.

[5] Bergstra, J., A. Ponse and S. Smolka, editors, "Handbook of Process Algebra," North-Holland, 2001 pp. 427–478.

[6] Borzyszkowski, T., *Logical systems for structured specifications*, Theoretical Computer Science **286** (2002), pp. 197–245.

[7] Calder, M., S. Maharaj and C. Shankland, *A Modal Logic for Full LOTOS Based on Symbolic Transition Systems*, The Computer Journal **45** (2002), pp. 55–61.

[8] Calder, M. and C. Shankland, *A Symbolic Semantics and Bisimulation for Full LOTOS*, in: M. Kim, B. Chin, S. Kang and D. Lee, editors, *Formal Techniques for Networked and Distributed Systems (FORTE)*, IFIP Conference Proceedings **197** (2001), pp. 185–200.

[9] Choppy, C., P. Poizat and J.-C. Royer, *A Global Semantics for Views*, in: T. Rus, editor, *International Conference on Algebraic Methodology And Software Technology (AMAST'2000)*, Lecture Notes in Computer Science **1816** (2000), pp. 165–180.

[10] Choppy, C., P. Poizat and J.-C. Royer, *Formal Specification of Mixed Components with Korrigan*, in: *Asia-Pacific Software Engineering Conference (APSEC'2001)* (2001), pp. 169–176.

[11] Choppy, C., P. Poizat and J.-C. Royer, *The Korrigan Environment*, Journal of Universal Computer Science **7** (2001), pp. 19–36, special issue on Tools for System Design and Verification.

[12] Goguen, J. A. and R. M. Burstall, *Institutions: abstract model theory for specifications and programming*, Journal of the ACM **39** (1992), pp. 95–146.

[13] Medvidovic, N. and R. N. Taylor, *A Classification and Comparison Framework for Software Architecture Description Languages*, IEEE Transactions on Software Engineering **26** (2000), pp. 70–93.

[14] Sannella, D. and A. Tarlecki, *Toward formal development of programs from algebraic specifications: implementations revisited*, Acta Informatica **25** (1988), pp. 233–281.

[15] van Oosten, J., *Basic category theory*, Lecture Series LS-95-1, BRICS (1995), http://www.brics.aau.dk/BRICS/LS/95/1/BRICS-LS-95-1.ps.gz.