Towards heterogeneous formal specifications

Gilles Bernot[1] and Sophie Coudert[1] and Pascale Le Gall[1]

L.a.M.I., Université d'Évry, Cours Monseigneur Roméro, 91025 Evry Cedex, France

{bernot,coudert,legall}@lami.univ-evry.fr

Abstract. We believe that big software systems could be more easily formally specified if several specification approaches were allowed within a single system specification. We propose a notion of heterogeneous framework where the specifier can choose a dedicated specification framework for each specification module. We show how the resulting heterogeneous modular specifications can get semantics, and how modular proofs can still be performed on these specifications. Our contribution is mainly focussed on a sort of interoperability between heterogeneous specification modules and we retrieve, as much as possible, classical notions of "meta-formalisms," modularity for structured specifications, or inference systems, as they are well known in the algebraic specification community. With this respect, our work can be regarded as an attempt to unify frameworks, by accepting and formalizing heterogeneity.

Keywords: heterogeneous specifications, modularity, formal specifications, logical frameworks, inference systems, algebraic specifications, theorem proving.

Introduction

The huge size of software systems requires more and more rigorous approaches for their design, maintenance, reuse, etc. It is in particular especially important to take care of the specification methods and techniques for such systems. *Formal specifications* are more and more often used for critical parts of such systems, and to face the complexity and the variety of the requirements to be specified, several specification languages or dialects will probably be used in the future, each of them being dedicated to specific areas (nuclear plants, public transportation, telecommunications, hardware, etc.) Unfortunately, the wide variety of academic formal specification languages is rather understood as a handicap for formal specifications instead of a flexibility advantage. People need to know how to compare formal specification languages. For this reason, great efforts are devoted to *unifying frameworks*. Some of these efforts are made via "meta-formalisms," following the institution approach [GB84], where specification formalisms can be compared via some kind of institution morphisms. The institution-like approaches have the advantage to keep the flexibility of "tuning" a specification framework according to the project under development [Mes89] [AC92] [SS92] [CM93] [EGR94] [CR94] [Wol95]. Other recent efforts tend to propose a unique common framework, as simple as possible, containing only the indispensable aspects (e.g., within the ESPRIT Basic Research Working Group COMPASS or the IFIP Working Group on

Foundations of Systems Specification). A common framework has the advantage to provide a pragmatic way to mix several existing specification languages together, or at least to identify their intersection.

Heterogeneity *inside* a specification is not addressed by these unifying frameworks. The explosive growth of wide systems induces a cooperation of several subsystems which have been *a priori* made to fulfill very heterogeneous needs. They have been specified and developed according to heterogeneous specification and programming languages. Thus, it becomes very convenient to choose a dedicated specification framework for *each component* of a system, in a similar way than programming languages are chosen at the present time according to the component under development (and the skills of the manpower). In this article, we define heterogeneous specifications where each specified module can have its own underlying specification framework. Owing to this granularity the flexibility is maximal, since the specification language has not to be the same for all components of a huge system.

To define a *heterogeneous framework* for heterogeneous specifications, we have carefully avoided the approach which consists in embedding the whole specification into a big logic that would be a sort of "union" of all the logics appearing in the specification modules. It would have produced a framework far too rich, complex, and possibly inconsistent. Our choice has been to privilege the top level module of a structured specification. The semantics of a specification are made of models in the sense of the framework of the top level module, and the expressible sentences about the specification belong to this framework as well. Nevertheless it remains possible to prove sentences by using lemmas issued from the imported frameworks, and proved within them. Such an approach reduces to one the number of specification frameworks to understand in order to reuse a software part: it is sufficient to understand the top level module specification framework. For reusability purposes, such an approach could as well simplify the management of a library of heterogeneously specified softwares; it suffices to identify the top level framework.

We have also tried, as much as possible, to follow classical ideas from the algebraic specification community for usual (homogeneous) modular specifications. Our contribution is mainly focussed on a sort of *interoperability* between formal specifications, at the proof level via a heterogeneous inference bridge, and at the semantic level via extraction functors.

In the first section, we define the notion of *homogeneous framework*, which is slightly weakened with respect to general logics [Mes89] in order to directly accept inference relations containing structural induction principles. In the second section, we define a notion of *heterogeneous framework* which allows to properly define *heterogeneous specifications*, their *semantics*, and an associated *heterogeneous inference system*. Then, we show how some previous works on institution-like frameworks can be reused in order to build heterogeneous frameworks for free; lastly we provide a toy *example* of heterogeneous specification, and develop a typical example of *heterogeneous proof* which delegates the proof of a lemma to a heterogeneously imported module.

1 Homogeneous frameworks

Let *Set* and *Cat* respectively denote the category of sets[1] and the category of categories [ML71] [BW90]. We call "homogeneous framework" a slightly weakened notion of general logic [Mes89].

Definition 1 *A* homogeneous framework *is a quintuple* $(Sig, sen, mod, \vdash, \models)$ *where:*

- Sig *is a category whose objects are called* signatures
- sen *is a (covariant) functor from* Sig *to* Set *associating to each signature a set of* sentences
- mod *is a contravariant functor from* Sig *to* Cat *associating to each signature a category of* models
- $\vdash$*, called* inference system*, is a* $|Sig|$*-indexed family such that for each signature* Σ*,* $\vdash_\Sigma$ *is a binary relation included in* $\mathcal{P}(sen(\Sigma)) \times sen(\Sigma)$
- $\models$*, called* satisfaction relation*, is a* $|Sig|$*-indexed family such that for each signature* Σ*,* $\models_\Sigma$ *is a binary relation included in* $mod(\Sigma) \times sen(\Sigma)$

and the following properties are satisfied:

- $\vdash$ *is reflexive, monotonic and transitive, i.e., respectively*
  - $\forall \Sigma \in Sig, \forall \varphi \in sen(\Sigma),\ \{\varphi\} \vdash_\Sigma \varphi$
  - $\forall \Sigma \in Sig, \forall \Gamma \subseteq sen(\Sigma), \forall \Gamma' \subseteq sen(\Sigma), \forall \varphi \in sen(\Sigma),\quad \Gamma \vdash_\Sigma \varphi$ *and* $\Gamma \subseteq \Gamma' \implies \Gamma' \vdash_\Sigma \varphi$
  - $\forall \Sigma \in Sig, \forall \Gamma \subseteq sen(\Sigma), \forall \Gamma' \subseteq sen(\Sigma), \forall \varphi \in sen(\Sigma),\quad \Gamma \vdash_\Sigma \Gamma'$ *and*[2] $\Gamma \cup \Gamma' \vdash_\Sigma \varphi \implies \Gamma \vdash_\Sigma \varphi$
- (weak) $\vdash$-translation*: for any isomorphism between two signatures* $\sigma : \Sigma \to \Sigma'$*, any* $\Gamma \subseteq sen(\Sigma)$ *and any* $\varphi \in sen(\Sigma)$*,* $\Gamma \vdash_\Sigma \varphi$ *if and only if* $sen(\sigma)(\Gamma) \vdash_{\Sigma'} sen(\sigma)(\varphi)$
- (weak) satisfaction condition*: for any isomorphism between two signatures* $\sigma : \Sigma \to \Sigma'$*, any* $M' \in mod(\Sigma')$ *and any* $\varphi \in sen(\Sigma)$*,* $M' \models_{\Sigma'} sen(\sigma)(\varphi)$ *if and only if* $mod(\sigma)(M') \models_\Sigma \varphi$
- soundness*: for any* $\Gamma \subseteq sen(\Sigma)$ *and any* $\varphi \in sen(\Sigma)$*, if* $\Gamma \vdash_\Sigma \varphi$ *then*[3]

$$\forall M \in mod(\Sigma), \quad (M \models_\Sigma \Gamma) \implies (M \models_\Sigma \varphi)$$

In the remainder of this article, by notation abuse, $sen(\sigma)$ will still be denoted σ, and $mod(\sigma)$ will often be denoted U_σ by analogy with the usual notation for "the forgetful functor."

Homogeneous frameworks are logics as defined by Meseguer in [Mes89] except that the $\vdash$-translation and the satisfaction condition are restricted to signature isomorphisms only. It means that properties are preserved by renaming of operations and variables (since these notions are hidden behind signatures and sentences). Our motivation for such a weakening is:

[1] with *total functions* as morphisms

[2] $\Gamma \vdash_\Sigma \Gamma'$ means $\forall \psi \in \Gamma', \Gamma \vdash_\Sigma \psi$

[3] $M \models_\Sigma \Gamma$ means $\forall \psi \in \Gamma,\ M \models_\Sigma \psi$

– Without these restrictions to isomorphisms, it would not be possible to consider inference systems with structural induction. If we consider for example a (first order) signature morphism $\Sigma_1 \to \Sigma_2$ which is not surjective, then a proof by structural induction on Σ_1 has no reason to remain valid in Σ_2. Moreover, in such a case, the functor *mod* should concern only finitely generated models and similarly, the satisfaction condition is not always satisfied.
– Of course, to solve such situations, it is often argued that it is possible to tune the notions of signature, sentence or model to fulfill the definition of general logic. We do not encourage such modifications of homogeneous frameworks. Indeed, the more the homogeneous frameworks are tuned in order to fit a unifying presentation of frameworks, the more the relationships between heterogeneous specification modules are difficult to establish and the more their meaning become obscure (see e.g., partial-to-total translation in Section 3.2). It is also often argued that induction and finitely generated models consideration belong to a specification language level and has not to be captured directly in the logic. In computer science, the proof of non-trivial properties requires structural induction. Thus, we want to take into account the fundamental theoretical problems raised by structural induction.

However, to deal with the lack of satisfaction condition, we give the following definition which formalizes usual encapsulation principles of programming languages. It will be used in Section 2.2 to define semantics of heterogeneous specifications (Definition 6).

Definition 2 *A homogeneous framework* $(Sig, sen, mod, \vdash, \models)$ *being given, let* $\delta : \Sigma_{imp} \to \Sigma$ *be a signature morphism and* $\mathcal{I}mport$ *a subcategory of* $mod(\Sigma_{imp})$*. A functor* $F : \mathcal{I}mport \to mod(\Sigma)$ *is said* δ-encapsulated *if and only if*

$$\forall \varphi \in sen(\Sigma_{imp}), \forall M \in \mathcal{I}mport, \quad M \models_{\Sigma_{imp}} \varphi \iff F(M) \models_{\Sigma} \delta(\varphi)\,.$$

Most of the time (e.g., [BB91] [BEPP87] [NOS95]), instead of δ-encapsulated functors, authors consider functors such that $U_\delta \circ F = Id$ and the δ-encapsulation results from the satisfaction condition (see Section 3.1).

Definition 3 *A homogeneous framework* $(Sig, sen, mod, \vdash, \models)$ *being given:*

– *A (flat)* presentation *is a tuple* $P = (\Sigma, \Gamma)$ *such that* Σ *is a signature and* $\Gamma \subseteq sen(\Sigma)$*. The elements of* Γ *are often called the* axioms *of* P*. We define presentation morphisms* $\sigma : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ *as signature* isomorphisms $\sigma : \Sigma \to \Sigma'$ *such that* $\Gamma' \vdash_{\Sigma'} \sigma(\Gamma)$*. Let* $Pres$ *denote the category of presentations, and* $sign : Pres \to Sig$ *denote the forgetful functor.*
– *The semantics of presentations is by definition the contravariant functor* Mod *from* $Pres$ *to* Cat *such that* $Mod(\Sigma, \Gamma)$ *is the full subcategory of* $mod(\Sigma)$ *whose objects satisfy* Γ *with respect to* $\models$*.*[4]

[4] *Mod* is actually a functor owing to the $\vdash$-translation and the satisfaction condition.

– *a* specification module *is a tuple* $\triangle P = (\delta, \triangle\Gamma)$ *such that* $\delta : \Sigma_{imp} \to \Sigma$ *is a signature morphism and* $\triangle\Gamma \subseteq sen(\Sigma)$*. The signature* Σ_{imp} *is called the* imported signature *of the module.*

– *in the homogeneous case,* structured specifications *can be recursively defined as follows: any presentation* $P \in Pres$ *is a structured specification and its* flattened signature *is* $sign(P)$*; if* SP_{imp} *is a structured specification whose flattened signature is* Σ_{imp}*, and* $\triangle P = (\delta : \Sigma_{imp} \to \Sigma\,,\ \triangle\Gamma)$ *is a specification module, then they form a structured specification denoted* $SP = SP_{imp} + \triangle P$*, whose flattened signature is* Σ*.*

Intuitively, in most cases, $\triangle\Gamma$ contains sentences specifying only the new operations introduced by the signature morphism δ (belonging to $\Sigma \setminus \delta(\Sigma_{imp})$). Our structured specifications are defined in a similar way as [NOS95] [BEPP87] for simple enrichment. Let us note that usually, authors define multiple import of modules as a simple extension of the simple import defined above. They simply use unions of modules which are rather easy to define in an homogeneous framework (provided that Sig has pushouts, or that specifications have amalgamation [EGR94].

2 Heterogeneous frameworks

2.1 Definitions

Intuitively, we define a heterogeneous framework in order to provide a family of "bridges" between homogeneous frameworks. Let a specification module, written according to a homogeneous framework o, import another module, written according to a homogeneous framework i. We need to know how to transpose signatures from i to o, how to extract models from i to o and how to heterogeneously infer sentences from i to o. This is defined as follows.

Definition 4 *A* heterogeneous framework *is a quadruple* $(\mathcal{B}, Tr, Ext, \Vdash)$ *where:*

– $\mathcal{B}$ *is a set of homogeneous frameworks (Definition 1) whose elements are called* basic frameworks

– Tr *is a family indexed by a subset* $\mathcal{D}$ *of* $\mathcal{B} \times \mathcal{B}$*, such that for each* (i, o) *in* $\mathcal{D}$*,* Tr_i^o *is a functor from* Sig_i *to* Sig_o *, called* signature transposition functor *(from the input framework* i *to the output framework* o*)*

– Ext *is a family, also indexed by* $\mathcal{D}$*, such that* Ext_i^o *is a natural transformation from the functor* mod_i *to the functor* $mod_o \circ Tr_i^o$ *, each* $Ext_{i,\Sigma}^o : mod_i(\Sigma) \to mod_o(Tr_i^o(\Sigma))$ *(for* $\Sigma \in Sig_i$*) is called* model extraction functor

– $\Vdash$ *is a family indexed by the set* $\{(i, o, \Sigma) \mid (i, o) \in \mathcal{D}$ *and* $\Sigma \in Sig_i\}$*, such that* $\Vdash_{i,\Sigma}^o$ *is a binary relation included in* $\mathcal{P}(sen_i(\Sigma)) \times sen_o(Tr_i^o(\Sigma))$*, called* heterogeneous inference bridge

and the following properties are satisfied:

- *the domain* $\mathcal{D}$ *contains the diagonal* $\{(b,b) \mid b \in \mathcal{B}\}$
- $\forall b \in \mathcal{B},\ Tr_b^b = Id_{Sig_b}$
- $\forall b \in \mathcal{B},\ Ext_b^b = Id_{mod_b}$
- diagonal reflexivity*: for any* $b \in \mathcal{B}$*, any* $\Sigma \in Sig_b$ *, and any* $\Gamma \subseteq sen_b(\Sigma)$*,* $(\Gamma \Vdash^b_{b,\Sigma} \varphi \iff \varphi \in \Gamma)$
- $\Vdash$ *is monotonic*
- $\Vdash$-translation*: for any* $(i,o) \in \mathcal{D}$*, any isomorphism* $\sigma : \Sigma \to \Sigma'$ *in* Sig_i *, any* $\Gamma \subseteq sen_i(\Sigma)$*, and any* $\varphi \in sen_o(Tr_i^o(\Sigma))$ *,* $\Gamma \Vdash^o_{i,\Sigma} \varphi$ *if and only if* $\sigma(\Gamma) \Vdash^o_{i,\Sigma'} Tr_i^o(\sigma)(\varphi)$
- heterogeneous soundness*: for any* $(i,o) \in \mathcal{D}$*, for any* $\Gamma \subseteq sen_i(\Sigma)$*, for any* $\varphi \in sen_o(Tr_i^o(\Sigma))$*, if* $\Gamma \Vdash^o_{i,\Sigma} \varphi$ *then for all* $M \in mod_i(\Sigma)$*, we have* $[M \models_i \Gamma \implies Ext^o_{i,\Sigma}(M) \models_o \varphi]$

Since we want an inference bridge as elementary as possible, we generally prefer for $\Vdash$ to be a recursive set. Expressing this constraint is not easy [Mes89] [SS95].

Transitivity would be meaningless for $\Vdash$ because a sentence according to the framework o cannot be introduced back in the framework i in general. $\Vdash$ should be understood as an unique inference step (a "bridge") in the heterogeneous proof process. However, the whole heterogeneous proof process (Definition 7) will be transitive.

The syntax of structured specifications composed of modules from different frameworks can be defined as follows.

Definition 5 *A heterogeneous framework* $(\mathcal{B}, Tr, Ext, \Vdash)$ *being given, the set Spec of all* heterogeneous structured specifications *is recursively defined as follows:*

- *Any presentation* $(\Sigma, \Gamma) \in Pres_o$ *for some basic framework* $o \in \mathcal{B}$ *is a heterogeneous structured specification. Its* flattened signature *is* Σ *(which belongs to the basic framework* o*).*
- *If* SP_ι *is a heterogeneous structured specification whose flattened signature* Σ_ι *belongs to some basic framework* $i \in \mathcal{B}$*, and if* $\triangle P = (\delta : \Sigma_{imp} \to \Sigma\ ,\ \triangle\Gamma)$ *is a specification module belonging to some basic framework* $o \in \mathcal{B}$ *such that* $(i,o) \in \mathcal{D}$ *and* $Tr_i^o(\Sigma_\iota) = \Sigma_{imp}$*, then* SP_ι *and* $\triangle P$ *form a structured specification denoted* $SP = SP_\iota + \triangle P$*, whose flattened signature is* Σ *(which belongs to the basic framework* o*).*

Moreover, with the previous notations,

- $\triangle P$ *is called the top level, or leading module of* SP*,*
- o *is called the* leading basic framework *of* SP*,*
- *and we extend sign to Spec by* $sign(SP) = \Sigma$ *(the flattened signature).*

2.2 Heterogeneous semantics

In Definition 6 below, we follow an idea similar to [BB91] in order to give modular semantics to our heterogeneous structured specifications. The semantics of a structured specification SP is supposed to represent a set of "acceptable programs" for SP. We assume moreover that the modularity of the considered programs matches the structure of SP. Then, the semantics of a program module can be represented by a δ-encapsulated functor. As in [BB91], if the semantics of a program module is correct with respect to the corresponding specification module $\triangle P$, then it means that necessarily, it is independent of the chosen implementation of the imported module.

Definition 6 *The* modular semantics *of a heterogeneous structured specification SP, denoted $Het(SP)$, is the category recursively defined as follows:*

- *If SP is reduced to a homogeneous presentation $P \in Pres_o$, then*[5] $Het(SP) = Mod_o(P)$
- *If $SP = SP_i + \triangle P$, where the leading basic framework of SP_i is i and the leading module $\triangle P = (\delta : \Sigma_{imp} \rightarrow \Sigma\,,\ \triangle\Gamma)$ belongs to the basic framework o, let $\mathcal{F}$ be the class of all δ-encapsulated functors F from $Ext^o_{i,\Sigma_i}(Het(SP_i))$ to $Mod_o(\Sigma, \triangle\Gamma)$, then $Het(SP)$ is the class of all $F(Ext^o_{i,\Sigma_i}(M))$ such that M belongs to $Het(SP_i)$.*

Let us notice that $Het(SP)$ is included in $mod_o(sign(SP))$. Thus, the basic framework of the top module of SP "hides" all the other imported modules. We choose this approach in order to avoid the "complexity explosion" of semantics carrying more or less the union of the frameworks appearing in the specification. From a methodological point of view, our approach has the advantage to allow the user of a heterogeneous specification to only take care about one formalism, the leading basic formalism. This will facilitate reuse purposes and the management of heterogeneous specification libraries.

Remark The class of functors $\{\, F \circ Ext^o_{i,\Sigma_i}|_{Het(SP_i)} : Het(SP_i) \rightarrow Het(SP) \mid F \in \mathcal{F} \,\}$ is called "the semantics of the module $\triangle P$ over SP_i." Notice that $\mathcal{F}$ can be empty, in which case $Het(SP)$ is also empty and the structured specification SP is said *inconsistent*.

2.3 Heterogeneous proofs

Definition 7 *A heterogeneous framework $(\mathcal{B}, Tr, Ext, \Vdash)$ being given, the corresponding* heterogeneous inference system *is the least binary relation* $\Vdash\!\!\!\Vdash \subseteq Spec \times \coprod_{b \in \mathcal{B}} sen_b(Sig_b)$ *such that, for any $SP \in Spec$ (with leading framework o):*

[5] In [BB91] the authors prefer to restrict the semantics of basic cases to initial or minimal models only. It would not be difficult to follow a similar approach here, however the heterogeneous dimension of our approach allows to have dedicated basic frameworks in $\mathcal{B}$ in order to reach such features.

– $\Vdash\!\!\vdash$ is compatible with $\vdash_o$: *when* SP *is of the form* $SP_\iota + (\delta\,,\,\triangle\Gamma)$ *(resp.* (Σ, Γ)*), for any* $\varphi \in sen_o(sign(SP))$ *if* $\triangle\Gamma \vdash_o \varphi$ *(resp.* $\Gamma \vdash_o \varphi$*) then* $SP \Vdash\!\!\vdash \varphi$.

– $\Vdash\!\!\vdash$ is compatible with $\Vdash$: *when* SP *is of the form* $SP_\iota + (\delta\,,\,\triangle\Gamma)$*, for any* $\Gamma' \subseteq sen_i(sign(SP_\iota))$*, and for any* $\varphi \in sen_o(Tr_i^o(sign(SP_\iota)))$*, if* $SP_\iota \Vdash\!\!\vdash \Gamma'$ *and* $\Gamma' \Vdash_i^o \varphi$ *then* $SP \Vdash\!\!\vdash \delta(\varphi)$.

– $\Vdash\!\!\vdash$ is transitive*: when* SP *is of the form* $SP = SP_\iota + (\delta\,,\,\triangle\Gamma)$*, for any* $\Gamma' \subseteq sen_o(sign(SP))$*, and for any* $\varphi \in sen_o(sign(SP))$*, if* $SP \Vdash\!\!\vdash \Gamma'$ *and* $(SP_\iota + (\delta\,,\,\triangle\Gamma \cup \Gamma')) \Vdash\!\!\vdash \varphi$ *then* $SP \Vdash\!\!\vdash \varphi$.

Notice that $SP \Vdash\!\!\vdash \varphi$ implies that the leading basic framework of SP is the framework of φ, and the flattened signature of SP is the underlying signature of φ. The leading basic framework given by the top level module masks all the subspecification frameworks. Properties inherited from subspecifications are filtered by the leading basic framework. These properties recursively come up through the specification structure.

To prove from SP a sentence φ by means of our heterogeneous proof process ($SP \Vdash\!\!\vdash \varphi$), we have to find a set of lemmas Γ_o (possibly empty) which, added to the axioms of the top level module, infers φ in the leading basic framework o (via $\vdash_o$). Then for each lemma ψ in Γ_o, we look for a set of lemmas Γ_i in the leading basic framework of the direct subspecification such that $\Gamma_i \Vdash_i^o \psi$. Then we recursively prove each sentence of Γ_i in the same way.

Theorem 8 *For every heterogeneous framework, the corresponding heterogeneous inference system is sound. This means that for any heterogeneous structured specification* SP *and for any sentence* φ *belonging to the leading basic framework* o *of* SP*, we have:*

$$SP \Vdash\!\!\vdash \varphi \quad \Longrightarrow \quad (\forall M \in Het(SP),\ M \models_o \varphi)$$

Proof: By induction on the proofs. If the last proof step comes from the first property of Definition 7 (compatibility with $\vdash_o$), then $SP \Vdash\!\!\vdash \varphi$ comes from $\triangle\Gamma \vdash_o \varphi$ (resp. $\Gamma \vdash_o \varphi$). Since $\vdash_o$ is sound and $Het(SP) \subseteq Mod_o(\Sigma, \triangle\Gamma)$ (resp. $Het(SP) \subseteq Mod_o(\Sigma, \Gamma)$) we have $Het(SP) \models_o \varphi$. For a step using the compatibility with $\Vdash$, the heterogeneous soundness of $\Vdash$ and the δ-encapsulation property ensure the soundness of $\Vdash\!\!\vdash$. For a $\Vdash\!\!\vdash$ transitivity step, $SP \Vdash\!\!\vdash \varphi$ comes from $SP \Vdash\!\!\vdash \Gamma'$ and $(SP_\iota + (\delta, \triangle\Gamma \cup \Gamma')) \Vdash\!\!\vdash \varphi$ with $SP = SP_\iota + (\delta : \Sigma_\iota \to \Sigma, \triangle\Gamma)$.

Let $\mathcal{F} = \{F : Ext^o_{i,\Sigma_\iota}(Het(SP_\iota)) \to Mod_o(\Sigma, \triangle\Gamma) \mid F \text{ is a } \delta\text{-encapsulated functor }\}$

let $\mathcal{F}' = \{F : Ext^o_{i,\Sigma_\iota}(Het(SP_\iota)) \to Mod_o(\Sigma, \triangle\Gamma \cup \Gamma') \mid F \text{ is a } \delta\text{-encapsulated functor }\}$

by induction hypothesis, $SP \Vdash\!\!\vdash \Gamma'$ is sound, then $\forall F \in \mathcal{F}, \forall M \in F(Ext^o_{i,\Sigma_\iota}(Het(SP_\iota)))$, $M \models_\Sigma \Gamma'$, so we have $\mathcal{F} = \mathcal{F}'$, and then $Het(SP_\iota + (\delta, \triangle\Gamma \cup \Gamma')) = Het(SP)$. By induction hypothesis, $(SP_\iota + (\delta, \triangle\Gamma \cup \Gamma')) \models_\Sigma \varphi$. It results that $SP \models_\Sigma \varphi$. □

The heterogeneous soundness property ensures that properties inherited via the heterogeneous inference bridge are also transmitted through the model extraction. Conversely, we can define a property

of "local heterogeneous completeness" which ensures that each sentence preserved by model extraction can be inferred via the heterogeneous bridge [Cou95]. Unfortunately, when all the homogeneous inference systems $\vdash$ are complete, this local heterogeneous completeness is not sufficient to ensure the completeness of the heterogeneous inference system $\Vdash$ (except if the specification contains at most one heterogeneous bridge).

3 Examples

3.1 Homogeneous modular semantics

A specification where all the modules are expressed in a unique basic framework b obviously corresponds to a structured specification in the homogeneous framework b as defined in Definition 1. Our heterogeneous framework gives modular semantics for such homogeneous structured specifications.

Of course, the resulting semantics for such homogeneous specifications meet the classical requirements for modular design of software [BB91], [NOS95]. All the authors share a common principle expressing that a module preserves the import part. More precisely, the semantics of a module is such that the application of the forgetful functor to any model of the global specification gives a model of the subspecification. If the underlying homogeneous framework is an institution, then the satisfaction condition and this semantic conservativity property ensures the δ-encapsulation. This allows to perform structured proofs accordingly to the structure of the specification (as we do in Section 7). As seen in Section 1, our definition of homogeneous framework admits institutions as a particular case.

3.2 Using maps between formalisms

To face the multiplicity of specification formalisms, many previous works contributed to give an unifying presentation of these formalisms. They abstract details which are specific to each specification formalism. For example, institutions, pre-institutions, specification frameworks or general logics are such "meta-formalisms." They are used to characterize some general properties of specification formalisms and to relate different specification formalisms. These relations are given by means of some kinds of translations between two formalisms. Their purpose is to be able either to compare their expressive power or to take benefit, for the second formalism, of results or tools already available in the first formalism (e.g., a theorem prover). For this aim, they characterize different kinds of translation between formalisms defined according to their underlying meta-formalism. Depending on the context, these translations are called morphisms, transformations, simulations or maps. They share some common properties. First, these translations are expressed in term of natural transformation in order to ensure a certain compatibility with the signature category. They also impose an equivalence between the two satisfaction relations involving models and sentences corresponding each other by the translation.

This provides means to combine two specification formalisms, but it does not really correspond to our notion of heterogeneousness since they do not look for integrating heterogeneous specification modules inside a unique global specification. Nevertheless, if we choose two formalisms related by a translation, it gives us examples for free of tuples $(i, o) \in \mathcal{D}$ for our heterogeneous framework (even if it may need some adaptations). Intuitively, their satisfaction conditions are very close to our heterogeneous soundness (Definition 4). We sketch below how to make use of their works, provided that, from a technical point of view, the considered specification formalisms fit our homogeneous framework definition[6] :

- Our first example concerns institutions and institution morphisms defined in [GB84]. (It could also be relevant for semi-institution morphisms as in [ST88], provided that we add a translation of sentences.) Let $\mathcal{I} = (Sign, sen, mod, \models)$ and $\mathcal{I}' = (Sign', sen', mod', \models')$ two institutions and an institution morphism $\mu : \mathcal{I} \rightarrow \mathcal{I}'$. By definition, μ consists of a functor $\mu_{Sign} : Sign \rightarrow Sign'$, a natural family of functions $\mu_{sen\Sigma} : sen'(\mu_{Sign}(\Sigma)) \rightarrow sen(\Sigma)$ and a natural family of functors $\mu_{mod\Sigma} : mod(\Sigma) \rightarrow mod'(\mu_{Sig}(\Sigma))$ such that:

  $$\forall \Sigma \in Sign, \forall \varphi' \in sen'(\Sigma), \forall M \in mod(\Sigma), M \models_{\Sigma} \mu_{sen\Sigma}(\varphi') \Longleftrightarrow \mu_{mod\Sigma}(M) \models_{\mu_{Sig}(\Sigma)} \varphi'$$

  It suffices to let $Tr^{\mathcal{I}'}_{\mathcal{I}} = \mu_{Sign}$, $Ext^{\mathcal{I}'}_{\mathcal{I},\Sigma} = \mu_{mod\Sigma}$ and $\Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma}$ is defined by: $\forall \varphi' \in sen'(\mu_{Sign}(\Sigma)), \forall \Gamma \subseteq \mathcal{P}(sen(\Sigma)), (\mu_{sen\Sigma}(\varphi') \in \Gamma \Leftrightarrow \Gamma \Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma} \varphi')$. With these definitions, the heterogeneous soundness condition is obviously satisfied.

  From a technical point of view, such uses of institution morphisms prove to be very suitable for our purpose because they directly give the property of heterogeneous soundness.
- The second example concerns institutions and simulations defined in [AC92]. Let us consider again two institutions $\mathcal{I}$ and $\mathcal{I}'$ and a simulation $\mu : \mathcal{I} \rightarrow \mathcal{I}'$. By definition, μ consists of a functor $\mu_{Sign} : Sign \rightarrow Sign'$, a natural family of functions $\mu_{sen\Sigma} : sen(\Sigma) \rightarrow sen'(\mu_{Sign}(\Sigma))$ and a natural family of partial representative[7] functors $\mu_{mod\Sigma} : mod'(\mu_{Sign}(\Sigma)) \rightarrow mod(\Sigma)$ such that: $\forall \Sigma \in Sign, \forall \varphi \in sen(\Sigma), \forall M' \in mod'(\mu_{Sign}(\Sigma)), \mu_{mod\Sigma}(M') \models_{\Sigma} \varphi \Longleftrightarrow M' \models_{\mu_{Sig}(\Sigma)} \mu_{sen\Sigma}(\varphi)$. It suffices to let $Tr^{\mathcal{I}'}_{\mathcal{I}} = \mu_{Sign}$, to choose $Ext^{\mathcal{I}'}_{\mathcal{I},\Sigma}$ in such a way that $\mu_{mod\Sigma} \circ Ext^{\mathcal{I}'}_{\mathcal{I},\Sigma} = Id_{mod(\Sigma)}$ (it is possible thanks to the surjectivity condition on $\mu_{mod\Sigma}$) and $\Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma}$ is defined by: $\forall \varphi \in sen(\Sigma), \forall \Gamma \subseteq sen(\Sigma), (\varphi \in \Gamma \Longleftrightarrow \Gamma \Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma} \mu_{sen\Sigma}(\varphi))$. As for the previous case, the heterogeneous soundness condition still holds.
- An example very similar is the one of logics and maps of logics defined in [Mes89]. Let us consider two logics $\mathcal{L} = (Sign, sen, mod, \vdash, \models)$ and $\mathcal{L}' = (Sign', sen', mod', \vdash', \models')$ and a map of logics $\mu : \mathcal{L} \rightarrow \mathcal{L}'$. This map is built from a functor $\mu_{Th_o} : Th_o \rightarrow Th'_o$ where Th_o denotes the category whose objects are theories (Σ, Γ) with $\Gamma \subset sen(\Sigma)$ and morphisms $\sigma : (\Sigma, \Gamma) \rightarrow (\Xi, \Delta)$ define a signature morphism $\sigma : \Sigma \rightarrow \Xi$ and are axiom-preserving ($\sigma(\Gamma) \subseteq \Delta$). The map μ also comprises two natural families $\mu_{sen(\Sigma,\Gamma)} : sen((\Sigma, \Gamma)) \rightarrow sen'(\mu_{Th_o}((\Sigma, \Gamma)))$ and $\mu_{mod(\Sigma,\Gamma)} : mod'(\mu_{Th_o}((\Sigma, \Gamma))) \rightarrow mod((\Sigma, \Gamma))$

[6] if necessary, by taking the satisfaction relation for defining the inference system as in the example of institutions

[7] It means surjective both on the objects and the arrows

verifying that the functor μ_{Th_o} is μ_{sen}-sensible[8]. Moreover, with the notation $\mu_{Th_o}((\Sigma,\emptyset)) = (\Sigma', \Gamma'_\Sigma)$, the two following properties are satisfied : for any $\Gamma \subseteq sen(\Sigma)$, and for any $\varphi \in sen(\Sigma)$, $\Gamma \vdash_\Sigma \varphi \Longrightarrow \mu_{sen\Sigma}(\Gamma) \cup \Gamma'_\Sigma \vdash'_{\Sigma'} \mu_{sen\Sigma}(\varphi)$ and for any model M' in $mod'((\Sigma', \Gamma'_\Sigma))$, $M' \models_{\Sigma'} \mu_{sen\Sigma}(\varphi) \Longleftrightarrow \mu_{mod(\Sigma,\emptyset)}(M') \models_\Sigma \varphi$. If we assume that the functors $\mu_{mod(\Sigma,\emptyset)}$ are representative, we can define heterogeneous use as above with simulations : it suffices to let $Tr^{\mathcal{L}'}_{\mathcal{L}}(\Sigma) = sign'(Th_o((\Sigma,\emptyset)))$, to choose $Ext^{\mathcal{L}'}_{\mathcal{L},\Sigma}$ in such a way that $\mu_{mod(\Sigma,\emptyset)} \circ Ext^{\mathcal{L}'}_{\mathcal{L},\Sigma} = Id_{mod(\Sigma)}$ and $\Vdash^{\mathcal{L}'}_{\mathcal{L},\Sigma}$ is defined by: $\forall \varphi \in sen'(\Sigma'), \forall \Gamma \subseteq sen(\Sigma)((\varphi \in \mu_{sen\Sigma}(\Gamma) \ \vee \ \varphi \in \Gamma'_\Sigma) \Longleftrightarrow \Gamma \Vdash^{\mathcal{L}'}_{\mathcal{L},\Sigma} \varphi)$. Once again, the heterogeneous correction condition holds.

– The last example concerns pre-institutions and pre-institutions transformations defined in [SS92]. Let $\mathcal{I} = (Sign, sen, mod, \models)$ and $\mathcal{I}' = (Sign', sen', mod', \models')$ two pre-institutions and a pre-institution transformation $\mu : \mathcal{I} \to \mathcal{I}'$. By definition, μ consists of a functor $\mu_{Sign} : Sign \to Sign'$, a family of natural functions $\mu_{sen\Sigma} : \mathcal{P}(sen(\Sigma)) \to \mathcal{P}(sen'(\mu_{Sig}(\Sigma)))$ and a family of natural functions $\mu_{mod\Sigma} : mod(\Sigma) \to \mathcal{P}(mod'(\mu_{Sig}(\Sigma)))$ ($\mu_{mod\Sigma}(M)$ being nonempty) such that: $\forall \Sigma \in Sign, \forall \Gamma \subseteq sen(\Sigma), \forall M \in mod(\Sigma), M \models_\Sigma \Gamma \Longleftrightarrow (\forall M' \in \mu_{mod\Sigma}(M), M' \models_{\mu_{Sig}(\Sigma)} \mu_{sen\Sigma}(\Gamma))$. It suffices to let $Tr^{\mathcal{I}'}_{\mathcal{I}} = \mu_{Sign}$, to choose when it is possible $Ext^{\mathcal{I}'}_{\mathcal{I},\Sigma}(M)$ among $\mu_{mod\Sigma}(M)$ in such a way that $Ext^{\mathcal{I}'}_{\mathcal{I}}$ defines a functor and to define $\Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma}$ by: $\forall \Sigma \in Sign, \forall \Gamma \subseteq sen(\Sigma), \forall M \in mod(\Sigma), \forall \varphi' \in sen(\mu_{Sig}(\Sigma)), (\varphi' \in \mu_{sen\Sigma}(\Gamma) \Leftrightarrow \Gamma \Vdash^{\mathcal{I}'}_{\mathcal{I},\Sigma} \varphi')$. Once again, the heterogeneous soundness condition is obviously satisfied.

Using examples of the translations mentioned above and adapting them with the translation schemes developed above, we inherit numerous examples for our heterogeneous framework. However, these examples are not all as intuitive as each other. One should be aware of some "semantic shift" introduced by heterogeneity.

– The first example which can be brought up for our purpose is the one of the heterogeneous use of first order logic with equality $\mathcal{FOEQ}$ by equational logic $\mathcal{EQ}$ defined by means of institution morphism (this example is given in [GB84]). Intuitively, this case is convenient for us because this heterogeneous use just amounts to forget extra-informations. Thanks to this heterogeneous use, one can benefit in a somehow low-level specification module of properties inherited from high-level specification module.
– Another example which clearly exhibits what we call "semantic shift," is the one of heterogeneous use of partial algebras with predicates ($\mathcal{PAP}$) by total algebras with predicates ($\mathcal{TAP}$) defined by means of simulations (this example is given in [AC92]). The principle is well-known and consists in introducing a special constant denoted $\perp_s$ per sort s and a particular predicate D_s. The extracted total algebras are such that the predicate D_s exactly characterizes the definition domain for the sort s, the value of $\perp_s$ being the only one undefined. When translating sentences of $\mathcal{PAP}$, the predicates D_s are used to reduce the scope of sentences to the defined values.

[8] It means that their natural transformations $\mu_{sen(\Sigma,\Gamma)}$ only depend on the signatures . They are now denoted by $\mu_{sen\Sigma}$.

Let us remind that our purpose when defining heterogeneous modular semantics is to encapsulate the imported specification by the higher level module in order to only filter properties expressible in the leading formalism. Here, such an ulterior heterogeneous use would consider $\bot_s$ and D_s as any constant or predicate, forgetting the intuition which has motivated their introduction. In particular, there is no obligation to continue to propagate bottom values. Thus, the specifiers of the higher level module have to explicitely manage the bottom values.

As a conclusion, we share with all these works on meta-formalisms a common technical background and thus, we take advantage of their numerous results. However, before extracting a result from these works for our purpose, we should look at the accordance with the intuition of what heterogeneous modularity should be.

3.3 An example of heterogeneous proof

Let us consider a simple example where $\mathcal{B} = \{\mathcal{E}, \mathcal{O}\}$ and $\mathcal{D} = \mathcal{B}^2$, $\mathcal{E}$ being the classical first order logic with equality $\mathcal{FOEQ}$ [EM85], ($\mathcal{E}$ for short), and $\mathcal{O}$ being the simple theory of equational observational data types with sort observation (for $\mathcal{O}$, a signature is a triple (S, F, Obs) such that $Obs \subseteq S$, see e.g., [BBK94]).

For the tuple $(\mathcal{E}, \mathcal{O})$, we can define $Tr^{\mathcal{O}}_{\mathcal{C}}((S, F)) = (S, F, S)$, which meets the intuition that all imported sorts defined in $\mathcal{E}$ can be considered as observable. Then, the model extraction is the identity and the heterogeneous bridge is defined by " $\Gamma \Vdash^{\mathcal{O}}_{\mathcal{C}} \varphi$ if and only if $\varphi \in \Gamma \cap sen_{\mathcal{O}}(\Sigma)$," which makes sense because the set of the $\mathcal{O}$-sentences is included in the one of the $\mathcal{E}$-sentences. It just amounts to relate the equality predicates symbols of the two formalisms. Intuitively, this heterogeneous bridge is sound because the semantics of the equality for the observable sorts is the same as in $\mathcal{E}$.

Conversely, for the tuple $(\mathcal{O}, \mathcal{E})$, we can define $Tr^{\mathcal{E}}_{\mathcal{O}}((S, F, Obs)) = (S, F)$. The heterogeneous bridge can be defined by " $\Gamma \Vdash^{\mathcal{C}}_{\mathcal{O}} \varphi$ if and only if $\varphi \in \Gamma$ ". The model extraction is a little bit more complex. The identity would not suffice because values which are different but observationally equal have to become actually equal with respect to $\mathcal{E}$. The solution is to quotient the imported observational model with the observational equivalence, so that for all observational model M, $Ext^{\mathcal{E}}_{\mathcal{O}}(M) = M/_{\approx_{\mathcal{O}}}$. This is well defined because $\approx_{\mathcal{O}}$ is a congruence [BHW95] and the quotient as extraction ensures that the heterogeneous bridge is sound.

Now, let us consider the specification SP which consists of a Stack module according to $\mathcal{E}$, which imports a Set module according to $\mathcal{O}$. Booleans and elements are supposed to be specified in sub-specifications (according to $\mathcal{E}$ or $\mathcal{O}$ indifferently).

Module: Set	Module: Stack
Framework: $\mathcal{O}$	Framework: $\mathcal{E}$
Use: Elem	Use: Set with δ the inclusion of Σ_{Set} in Σ_{Stacks}
with δ the inclusion of Σ_{Elem} in Σ_{Set}	Sorts: $stack$
Sorts: set	Operations:
Observable: $elem, bool$	$empty \ : \rightarrow stack$
Operations:	$push \_\_ : set\ stack \rightarrow stack$
$\emptyset \ : \rightarrow set$	...
$ins \_\_ : elem\ set \rightarrow set$	$monopush \_\_ : set\ stack \rightarrow stack$
$\_ \in \_ : elem\ elem \rightarrow bool$	Axioms $\triangle\Gamma_{Stack}$:
Axioms $\triangle\Gamma_{Set}$:	...
$x \in \emptyset = false$	$X = Y \Rightarrow monopush(X, push(Y, P)) = push(Y, P)$
$x \in (ins\ y\ X) = (eq\ x\ y)\ or\ (y \in X)$	$X \neq Y \Rightarrow monopush(X, push(Y, P)) = push(X, push(Y, P))$
where $x, y : elem, X : set$	where $X, Y : set, P : stack$

Let us prove the sentence φ from the specification Stack :

$$(\varphi) \qquad monopush(ins(a, ins(b, \emptyset)), push(ins(b, ins(a, \emptyset)), empty) = push(ins(b, ins(a, \emptyset)), empty)$$

From the before last axiom of Stack, it suffices to prove the sentence ψ :

$$(\psi) \qquad ins(a, ins(b, \emptyset)) = ins(b, ins(a, \emptyset))$$

Via the heterogeneous bridge $\Vdash^{\mathcal{E}}_{\mathcal{O}}$, it suffices to prove $\triangle\Gamma_{Set} \vdash_{\mathcal{O}} \psi$. The only operation which allows to distinguish sets is the membership operation "$\in$" and it is easy to prove ψ by context induction (see [Hen91]).

This example of heterogeneous proof is an illustration that our heterogeneous proof system is built in such a way that the proof of lemmas concerning imported data types are delegated to the proof systems of the subspecifications. Moreover, this example shows that besides the compatibility between model extraction and provability, we endeavour to give a solution which cope with the intuition.

Conclusion

We have proposed a definition of heterogeneous framework which is, roughly speaking, a family of "homogeneous frameworks" together with relationships that allow to define heterogeneous structured specifications. In a heterogeneous specification, each module can be written according to its own homogeneous framework. Moreover, the heterogeneous specification inherits the formalism of its top level module. This approach limits the number of "visible" formalisms and avoids using the union of all frameworks appearing in the specification. We also give a heterogeneous proof principle where properties inherited from

imported specifications can be translated as lemmas in order to prove sentences in the top level formalism. Lastly, a small typical example has been developed.

Our approach is clearly a nice way to unify numerous specification formalisms. Moreover, this unification faithfully preserves all the specificities of each formalism. The relationships between formalisms remain as simple as possible in a heterogeneous framework because we combine them only two by two. It allows a local solving of difficulties, with minimal loss of information from a module to another one.

Nevertheless, our approach is only a first proposal for heterogeneity within specifications. Some improvements can be studied:

- To only consider heterogeneous soundness is not sufficient. A notion of heterogeneous *completeness* is missing. It is not difficult to define a notion of "local heterogeneous completeness" (which ensures that $\Vdash_i^o$ is as powerful as Ext_i^o). Unfortunately, this local completeness is not sufficient to ensure the completeness of $\Vdash$ in general. We will steadily work on this problem.
- In the homogeneous case, *multiple imports* in a module are rather easily defined as an extension of the single import, using pushouts of specifications. Pushouts of specifications which do not belong to the same homogeneous formalism are more complicated to apprehend, thus multiple imports (where the Tr_i^o are taken into account) have to be studied in more details.
- Our weakening of the notion of general logic (by restricting $\vdash$-translation and the satisfaction condition to signature isomorphisms only) is rather *ad hoc* (in order to capture proofs with structural induction). It is not directly the purpose of our work, however we would like to find a better solution, without distorting the meaning of Tr, Ext and $\Vdash$.
- Lastly, we are looking for a good definition of a category of heterogeneous structured signatures $Sig_{\mathcal{B}}$ (e.g., similar to [DR94]), as well as $sen_{\mathcal{B}}$ and $\models_{\mathcal{B}}$, in order to turn $(Sig_{\mathcal{B}}, sen_{\mathcal{B}}, Het, \Vdash, \models_{\mathcal{B}})$ into a homogeneous framework (Definition 1) itself. But it is indeed a rather esthetic consideration for the time being.

Acknowledgments: We would like to thank Marc Aiguier for a proofreading of the draft version of this article. This work has been partly supported by CEC under ESPRIT-III WG6112 COMPASS, and by the French "GDR de programmation."

References

[AC92] E. Astesiano and M. Cerioli. Relationships between logical frameworks. In LNCS, editor, *Recent Trends in Data Type Specification*, volume 655, pages 101–126, Dourdan, 1992.

[BB91] G. Bernot and M. Bidoit. Proving the correctness of algebraically specified software modularity and observability issues. In *Proc. of AMAST-2, Second Conference of Algebraic Methodology and Software Technology*, May 1991. Iowa City, Iowa, USA.

[BBK94] G. Bernot, M. Bidoit, and T. Knapik. Observational approaches to algebraic specifications: A comparative study. *Acta Informatica*, 31:651–671, 1994.

[BEPP87] E.K. Blum, H Ehrig, and F. Parisi-Pressicce. Algebraic specification of modules and their basic interconnections. *Journal of Computer Systems Science*, 34:293–339, 1987.

[BHW95] M. Bidoit, R. Hennicker, and M. Wirsing. Behavioural and abstractor specifications. *Science of Computer Programming*, 25:149–186, 1995.

[BW90] M. Barr and C. Wells. *Category Theory for Computer Science*. Prentice Hall, 1990.

[CM93] M. Cerioli and J. Meseguer. Can I borrow your logic ? In *Proc. Int. Mathematical Foundations of Computer Science, MFC'93, Gdansk*, pages 342–351, 1993.

[Cou95] S. Coudert. Vers une sémantique des spécifications hétérogènes. Université d'Evry, Rapport de DEA, 1995.

[CR94] M. Cerioli and G. Reggio. Institutions for very abstract specifications. In LNCS, editor, *Recent Trends in Data Type Specification, Caldes de Malavella*, volume 785, pages 113–127, 1994.

[DR94] E. David and C. Roques. An institution for modular specifications. *Proc. of the 10th British Colloquium on Theoretical Computer Science*, Bristol, 1994.

[EGR94] H. Ehrig and M. Grosse-Rhode. Functorial theory of parameterized specifications in a general specification framework. *Theoretical Computer Science*, pages 221–266, 1994. Elsevier Science Pub. B.V. (North-Holland).

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1. Equations and initial semantics*, volume 6. Springer-Verlag,EATCS Monographs on Theoretical Computer Science, 1985.

[GB84] J.A. Goguen and R.M. Burstall. Introducing institutions. In Springer-Verlag LNCS 164, editor, *Proc. of the Workshop on Logics of Programming*, pages 221–256, 1984.

[Hen91] R. Hennicker. Context induction: a proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3(4):326–345, 1991.

[Mes89] J. Meseguer. General logics. In North-Holland, editor, *Proc. Logic. Colloquium '87*, Amsterdam, 1989.

[ML71] S. Mac Lane. *Categories for the working mathematician*, volume 5 of *Graduate texts in mathematics*. Springer-Verlag, 1971.

[NOS95] M. Navarro, F. Orejas, and A. Sanchez. On the correctness of modular systems. *Theoretical Computer Science*, 140:139–177, 1995.

[SS92] A. Salibra and G. Scollo. A soft stairway to institutions. In LNCS, editor, *Recent Trends in Data Type Specification*, volume 655, pages 320–329, Dourdan, 1992.

[SS95] A. Sernadas and C. Sernadas. Theory spaces. Technical report, IST, Lisboa, 1995.

[ST88] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: Implementations revisited. *Acta Informatica*, 25:233–281, 1988.

[Wol95] U. Wolter. Institutional frames. In LNCS, editor, *Recent Trends in Data Type Specification*, volume 906, pages 469–482, 1995.