RAIRO-Inf. Theor. Appl. **41** (2007) 191–214
DOI: 10.1051/ita:2007015

OBJECT ORIENTED INSTITUTIONS TO SPECIFY SYMBOLIC COMPUTATION SYSTEMS*

César Domínguez[1], Laureano Lambán[1] and Julio Rubio[1]

Abstract. The specification of the data structures used in EAT, a software system for symbolic computation in algebraic topology, is based on an operation that defines a link among different specification frameworks like hidden algebras and coalgebras. In this paper, this operation is extended using the notion of institution, giving rise to three institution encodings. These morphisms define a commutative diagram which shows three possible views of the same construction, placing it in an equational algebraic institution, in a hidden institution or in a coalgebraic institution. Moreover, these morphisms can be used to obtain a new description of the final objects of the categories of algebras in these frameworks, which are suitable abstract models for the EAT data structures. Thus, our main contribution is a formalization allowing us to encode a family of data structures by means of a single algebra (which can be described as a coproduct on the image of the institution morphisms). With this aim, new particular definitions of hidden and coalgebraic institutions are presented.

Mathematics Subject Classification. 68Q65, 68Q60.

1. Introduction

EAT (*effective algebraic topology*) [25] is a software system developed by Rubio and Sergeraert. It is devoted to symbolic computation in algebraic topology. Particularly, it carries out calculations of homology groups of complex topological spaces, namely iterated loop spaces. By means of EAT, some homology groups

Keywords and phrases. Institution, symbolic computation, specification, object orientation.

* *Partially supported by MEC, project MTM2006-06513 and by Comunidad Autónoma de La Rioja, project ANGI2005/19.*

[1] Departamento de Matemáticas y Computación, Universidad de La Rioja, Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño, La Rioja, Spain;
`cesar.dominguez,laureano.lamban,julio.rubio @dmc.unirioja.es`

that had never been obtained with any other method, neither theoretical nor automatic, have been computed (therefore the corresponding topological results have not been confirmed, but neither refuted; some examples can be found in [24] or [26]). In view of the obtained results, the formal analysis of the program might be interesting, as even if it does not lead to a complete proof of the program correctness, at least it would allow for some kind of formal reasoning on the internal calculation processes of this software system.

In particular, first we studied one of the most important aspects of EAT, that is, its data structures. In that study [21, 22] we found that there are two different *layers* of data structures in EAT. In the first layer, one finds the usual data structures. For instance, the system handles integer numbers, (finite) lists or trees of symbols (to represent linear combinations or polynomials) and so on. In the second layer, one must deal with algebraic structures like (graded) groups, rings, simplicial sets or chain complexes, whose elements are data that belong to the first layer. Besides, two additional features enrich even more the problem. On the one hand, structures of the two layers have to be created and handled at runtime. This should be compared with other mathematical packages (in the field of commutative algebra or in general systems such as Mathematica or Maple) where the two above mentioned layers obviously exist, but where only one (or very few) algebraic structure is created in each session (in a pre-processing time, so to speak) and then a massive work on usual (first layer) data is carried out inside this structure. In contrast, in a typical calculation with EAT, several hundreds of algebraic structures must be created and handled (see examples in [25, 26]). On the other hand, algorithms in algebraic topology use, in an essential way, *infinite* structures as intermediaries to compute the sought (finite) results. (The *loop space* is an important construction of this kind [25].) Let us stress that these infinite data structures are not the usual sequential or hierarchical structures, like infinite lists or streams or infinite trees, where canonical ways of traversing the structure are known. In EAT, one must implement infinite groups, rings, chain complexes and so on. For these structures, *behaviour* and *observation* are more important than storage and traversing aspects. The way chosen by Sergeraert to develop EAT was based in an intensively functional programming use (see [21,25]). In this program, an element of the second layer is encoded by means of a record of common lisp functions which has a field for each operation of the algebraic structure which is been represented. The elements of an algebraic structure (which is represented by an instance of the functional record) are elements of the first layer. These elements are fixed or constructed from fixed elements, and so they do not require an explicit storage.

These particular characteristics imply that, even if for *first-layer* data structures the usual algebraic specification techniques (the initial algebra construction, in particular) suffice for the modelling task, for *second-layer* data structures, a research effort is required, since it is not possible to apply directly known techniques.

We first realized that in a system such as EAT we are not only implementing an abstract data type, or, shortly, an ADT (as a group, for instance), but also dealing with implementations of ADTs (several hundreds of *implementations* of the ADT group would populate the program memory). In [21] a construction, which is called *imp* operation, was defined. This construction models the skip from a kind of structures to families of these structures. Roughly speaking, the syntactical part of this construction consists in adding to a signature Σ a new sort which is incorporated as first argument to each operation of the signature (the new signature is denoted by Σ_{imp}). Then, a Σ_{imp}-algebra defines a family of Σ-algebras (the carrier set for the distinguished sort acts as an index for this family). Besides, working with implementations in [21] we were able to prove that EAT (second-layer) data structures are as general as possible, in the sense that they are ingredients of final objects in certain categories of ADT implementations. Later on, led by this characterization of EAT data structures, in [22] we reinterpreted our results in terms of hidden algebras [15] and coalgebras [27], technologies that have been presented in the literature as related to the object-oriented paradigm [14, 18, 27]. This led us to a purely (co)algebraic setting, without taking into account implementation or programming language issues. So, through the *imp* operation different specification frameworks, namely the equational algebraic specification, the hidden specification and the coalgebraic frameworks, were related.

In this work we try to extend our results by using the terminology of the institutions [13] (a notion that summarizes the idea of an abstract framework of specification) such that the *imp* operation defines relationships among the institutions that correspond to the aforementioned object-oriented frameworks of specification. The relations defined by the *imp* operation among these different specification contexts will give rise to a commutative diagram of institution encodings, which displays different forms to understand this construction, either placing it in the equational algebraic institution, in a (redefined) hidden institution or in a new institution for a particular kind of coalgebras. As a result, we obtain a formalization of a coding of a set of Σ-algebras by a single Σ_{imp}-algebra. Moreover, this Σ_{imp}-algebra can be described as the coproduct of a set of "trivial" Σ_{imp}-algebras, which corresponds to the set of Σ-algebras that is being coded.

The interest of this work is twofold. From the algebraic specification perspective, this paper can be understood as a report on the application of the institution notion in a field (namely, symbolic computation) quite distant from the context in which the concept was introduced. In this work, we show different problems that we have found in the direct use of institutional machinery for the formalization of actual symbolic computation systems and how these problems can be tackled. Our study shows this approach suffers from the gap between the generality of the institutional framework and the particularity of the algebraic structures which are under consideration. One consequence of this is the institutional framework is stretched in such a way that in some cases we will use degenerated parts of some institutions and in other cases we will need to introduce "ad-hoc" institutions. Another consequence is that the object-oriented flavour of the institutions could be lost in these degenerated cases.

From the symbolic computation point of view, it shows an innovative way in which the specification of data structures can be carried out (since, up to the authors' knowledge, the only occurrence of institutions in relation with symbolic computation is in the work by Calmet *et al.* [4,5], where, in any case, the objectives were quite different).

The paper is organized as follows. In the section below some definitions and examples that will be used later on in this paper are introduced. In Section 3 the *imp* construction is briefly presented. The most important results of this work are included in Section 4. In that section, a commutative diagram of institution encodings is built. Section 5 is devoted to some relationships between our constructions an other hidden and coalgebraic institutions that have been developed in the literature. The paper ends with a section of conclusions and future work.

2. Preliminaries

In order to make the reading of the paper easier, we start by recalling some terminology on specifications [23] and institutions [13].

Definition 2.1 (signature and algebra)**.** A *signature* Σ is a pair (S, Ω) of sets, whose elements are called *sorts* and *operations* respectively. Each operation $\omega \in \Omega$ has associated a $(n+1)$-tuple $s_1, \dots, s_n, s \in S$ with $n \geq 0$ (it is represented by $\omega\colon s_1 \dots s_n \to s$). In the case $n = 0$, the operation $\omega\colon \to s$ is called a *constant of sort s*.

Let $\Sigma = (S, \Omega)$ and $\Sigma' = (S', \Omega')$ be two signatures. A *signature morphism* $\mu\colon \Sigma \to \Sigma'$ *from* Σ *to* Σ' is a pair $\mu = (\mu_S\colon S \to S', \mu_\Omega\colon \Omega \to \Omega')$ of maps such that for each operation $\omega\colon s_1 \dots s_n \to s \in \Omega$, there exists an operation $\omega'\colon \mu_S(s_1) \dots \mu_S(s_n) \to \mu_S(s)$, with $\mu_\Omega(\omega) = \omega'$. If no ambiguities arise we may write μ instead of μ_S or μ_Ω.

Let $\Sigma = (S, \Omega)$ be a signature. A Σ-*algebra* A assigns a set A_s to each sort $s \in S$ (A_s is called the *carrier set* for the sort s) and a total function $\omega_A\colon A_{s_1} \times \cdots \times A_{s_n} \to A_s$ to each operation $\omega\colon s_1 \dots s_n \to s \in \Omega$.

Let $\Sigma = (S, \Omega)$ be a signature and let A, B be two Σ-algebras. A Σ-*homomorphism* $h\colon A \to B$ *from* A *to* B is a family $\{h_s\colon A_s \to B_s\}_{s \in S}$ of maps such that for each $\omega\colon s_1 \dots s_n \to s \in \Omega$, the following condition holds:

$$h_s(\omega_A(a_1, \dots, a_n)) = \omega_B(h_{s_1}(a_1), \dots, h_{s_n}(a_n))$$

for all $a_i \in A_{s_i}$, $i = 1, \dots, n$.

The Σ-algebras and Σ-homomorphisms define a category $Alg(\Sigma)$.

Definition 2.2 (terms)**.** Let $\Sigma = (S, \Omega)$ be a signature and let $X = (X_s)_{s \in S}$ be a set of *variables* for Σ. The set of $\Sigma(X)$-*terms*, denoted by $T_{\Sigma(X)}$, is the family of sets $(T_{\Sigma(X)s})_{s \in S}$ which are defined by: $X_s \subseteq T_{\Sigma(X)s}$, if ω is a constant of sort s then $\omega \in T_{\Sigma(X)s}$, and, finally, if $\omega\colon s_1 \dots s_n \to s \in \Omega$, $n \geq 1$, and $t_i \in T_{\Sigma(X)s_i}$, $i = 1, \dots, n$, then $\omega(t_1, \dots, t_n) \in T_{\Sigma(X)s}$. An element of $T_{\Sigma(X)s}$ is called a $\Sigma(X)$-*term of sort s*.

An institution tries to gather all the components present in a specification framework: signatures, sentences, models and satisfaction relation. From now on in this paper, *Set* will denote the category of sets and maps and *Cat* will denote the category of categories and functors (in order to avoid foundational problems related to *Cat* we can restrict our work to locally small categories [1]). Moreover, if C is a category C^{op} denotes the opposite category of C.

Definition 2.3 (institution)**.** An *institution* $\mathcal{I}$ consists of:

(1) a category $SIG_{\mathcal{I}}$, whose objects are called *signatures*;
(2) a functor $Sen_{\mathcal{I}}\colon SIG_{\mathcal{I}} \to Set$, giving for each signature Σ a set whose elements are called Σ-*sentences*;
(3) a functor $Mod_{\mathcal{I}}\colon SIG_{\mathcal{I}}^{op} \to Cat$, giving for each signature Σ a category whose objects are called Σ-*models*, and whose arrows are called Σ-*morphisms*; and
(4) for each $\Sigma \in Obj(SIG_{\mathcal{I}})$, a relation $\models^{\mathcal{I}}_{\Sigma} \subseteq Obj(Mod_{\mathcal{I}}(\Sigma)) \times Sen_{\mathcal{I}}(\Sigma)$, called Σ-*satisfaction*,

such that for all the morphisms $\mu\colon \Sigma \to \Sigma'$ in $SIG_{\mathcal{I}}$, the *satisfaction condition*

$$A' \models^{\mathcal{I}}_{\Sigma'} Sen_{\mathcal{I}}(\mu)(e) \quad \text{iff} \quad Mod_{\mathcal{I}}(\mu)(A') \models^{\mathcal{I}}_{\Sigma} e$$

holds for each $A' \in Obj(Mod_{\mathcal{I}}(\Sigma'))$ and each $e \in Sen_{\mathcal{I}}(\Sigma)$.

It is natural to establish a relationship between two institutions through the corresponding notion of morphism. An institution morphism is given by a functor between the categories of signatures and two natural transformations (one on sentences and the other on models) such that the satisfaction relations are preserved. In the usual notion of morphism, the transformation for the sentences "goes" in the opposite direction to the relationship between signatures and models. Nevertheless, there are examples where it is more natural to consider these three components as going in the same direction. This originates the concept of *institution encoding*. We introduce the definition of this concept as it was presented by Tarlecki in [28]. For a survey on different notions of morphisms between institutions, see [17].

Definition 2.4 (institution encoding)**.** Let $\mathcal{I}$ and $\mathcal{I}'$ be institutions. An *institution encoding* from $\mathcal{I}$ to $\mathcal{I}'$ consists of:

(1) a functor $\Phi\colon SIG_{\mathcal{I}} \to SIG_{\mathcal{I}'}$;
(2) a natural transformation $\alpha\colon Sen_{\mathcal{I}} \Rightarrow Sen_{\mathcal{I}'} \circ \Phi$; and
(3) a natural transformation $\beta\colon Mod_{\mathcal{I}} \Rightarrow Mod_{\mathcal{I}'} \circ \Phi^{op}$

such that the following *satisfaction condition* holds:

$$A \models^{\mathcal{I}}_{\Sigma} e \quad \text{iff} \quad \beta_{\Sigma}(A) \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_{\Sigma}(e)$$

for each signature Σ, each Σ-model A and each Σ-sentence e from $\mathcal{I}$.

The first institution we are going to introduce represents the equational algebraic specification framework [13]. This institution will be called *equational algebraic institution* and we will use $\mathcal{E}$ to denote it.

Example 2.5. The equational algebraic institution $\mathcal{E}$ is given by:

- the category $SIG_{\mathcal{E}}$ is the category whose objects are the signatures and whose morphisms are the signature morphisms;
- the functor $Sen_{\mathcal{E}}\colon SIG_{\mathcal{E}} \to Set$ gives for each signature Σ the set of algebraic equations $\forall X.t = t'$, where X is a set of variables for Σ and t, t' are $\Sigma(X)$-terms of the same sort, and for each signature morphism the translation along this morphism which is defined essentially by replacing the operation names as indicated by this morphism;
- the functor $Mod_{\mathcal{E}}\colon SIG_{\mathcal{E}}^{op} \to Cat$ gives for each signature Σ the category $Alg(\Sigma)$ of Σ-algebras, and for each signature morphism the reducts along this morphism which is defined in the usual way;
- for each signature Σ, the satisfaction relation $\models_{\Sigma}^{\mathcal{E}}$ is the usual equational satisfaction relation, which comes from the interpretation of the Σ-terms in the Σ-algebras.

The following example of institution is related to the hidden specification [15] and it will be called *hidden institution* [14].

Definition 2.6 (hidden signature)**.** Let $V\Sigma = (VS, V\Omega)$ be a signature. Let us fix a $V\Sigma$-algebra $D_{V\Sigma}$ and let us include in $V\Omega$, as constants, the elements of the carrier sets of $D_{V\Sigma}$ which do not correspond to constants previously in $V\Omega$. The elements of VS are called *visible sorts*, the elements of $V\Omega$ are called *visible operations* and the $V\Sigma$-algebra $D_{V\Sigma}$ is called *data domain.* A *hidden signature*, on $V\Sigma$ and $D_{V\Sigma}$, is a signature $H\Sigma = (S, \Omega)$ such that $S = HS \uplus VS$, $\Omega = H\Omega \uplus V\Omega$ (the elements of HS and $H\Omega$ are called *hidden sorts* and *hidden operations*, respectively), and for each operation $\omega\colon s_1 \dots s_n \to s$ in $H\Omega$ there is at least one hidden sort in $\{s_1, \dots, s_n, s\}$ and at most one hidden sort in $\{s_1, \dots, s_n\}$.

An operation $\omega\colon s_1 \dots s_n \to s \in H\Omega$ is called *constructor* if $s \in HS$ and *deconstructor* otherwise. Besides, a constructor is called *constructor from data* when $s_1, \dots, s_n$ are visible sorts, *constructor from objects* (and data) when there is one hidden sort among the arguments that is different from s, and *updating operation* otherwise. In this paper, we are going to use the terminology for the hidden operations introduced in [22], which does not agree with the usual one in the field of hidden specification [15].

Definition 2.7 (hidden signature morphism)**.** Let $H\Sigma$ and $H\Sigma'$ be two hidden signatures, on $V\Sigma$ and $D_{V\Sigma}$. A *hidden signature morphism* $\mu\colon H\Sigma \to H\Sigma'$ is a signature morphism such that it is the identity on visible sorts and visible operations, defines an injective map on hidden sorts and if $\omega'\colon s'_1 \dots s'_n \to s'$ belongs to $H\Omega'$ and some sort in $\{s'_1, \dots, s'_n\}$ lies in $\mu(HS)$, then $\omega' = \mu(\omega)$ for some $\omega \in H\Omega$. This last condition will be called *encapsulation condition.*

Note that, the one-to-one condition on hidden sorts is necessary in order to obtain a category for hidden signatures. For instance, we can consider three hidden signatures: Σ_1 with one operation $\{+\colon h_1 \to v\}$, Σ_2 with two operations $\{+\colon h_1 \to v, *\colon h_2 \to v\}$, Σ_3 with two operations $\{+\colon h_1 \to v, *\colon h_1 \to v\}$. These three signatures Σ_1, Σ_2 and Σ_3 are defined over the same visible signature (which has a unique visible sort v and no operations). So, $\{h_1\}$ is the hidden sort set for the first and third hidden signatures and $\{h_1, h_2\}$ for the second one. Now, we define two hidden morphisms among them. First, the hidden morphism μ_1 which is the "inclusion" from Σ_1 to Σ_2. Second, the hidden morphism μ_2 between Σ_2 and Σ_3 such that $\mu_2(h_1) = \mu_2(h_2) = h_1$, and $\mu_2(+) = +$, $\mu_2(*) = *$. This second hidden morphism satisfies all the conditions of a hidden morphism except the one-to-one condition on hidden sorts. But, it is clear that the composition $\mu_2 \circ \mu_1$ of these hidden morphisms is not a hidden morphism (since $\mu_2 \circ \mu_1(h_1) = h_1$ and does not exist an operation in Σ_1 whose image is the operation $*\colon h_1 \to v$ of Σ_3). This condition is not taken into account in the hidden categories defined in [14] and [18].

Definition 2.8 (hidden algebra)**.** A *hidden algebra* A for a hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, is an $H\Sigma$-algebra such that $A_{V\Sigma} = D_{V\Sigma}$ (in other words, the *reduct* of A to the visible part of the signature is equal to the data domain $D_{V\Sigma}$). A *hidden morphism* between two hidden algebras for a hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, is an $H\Sigma$-homomorphism h such that h_{VS} is the identity on $D_{V\Sigma}$. The hidden algebras for a hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, together with the hidden morphisms, define a category which will be denoted by $HAlg(H\Sigma)$.

Hidden algebra allows to model the distinction in the object-oriented paradigm between data that are used for values and data that are used for internal states (*e.g.*, of objects). It assigns the former to visible sorts and the latter to hidden sorts. States are "hidden" in that they are only observed through *contexts*, *i.e.*, special terms that returns visible data values, as defined below.

Let $H\Sigma = (S, \Omega)$ be a hidden signature, on $V\Sigma$ and $D_{V\Sigma}$. An $H\Sigma$-*context* is a visible $H\Sigma(\{z\})$-term that has a single occurrence of a variable symbol z. A context is said to be *appropriate* for a term t iff the sorts of t and z are the same. We write $c(t)$ for the result of the substitution of the term t for the occurrence of the variable z in the context c. It is said that a hidden $H\Sigma$-algebra A *behaviourally satisfies* an algebraic $H\Sigma$-equation $\forall X.t = t'$ iff for each $H\Sigma$-context c appropriate for t (and t'), $A \models^{\mathcal{E}}_{H\Sigma} \forall X.c(t) = c(t')$. When considering equations of visible sort the behavioural satisfaction is equivalent to the equational satisfaction. Two elements $a, a' \in A_s$, $s \in S$, are *behaviourally equivalent* iff A behaviourally satisfies the $H\Sigma$-equation $\forall\emptyset.a = a'$. Behavioural equivalence defines a congruence relation on A.

Let us fix a set U as sort *universe*. Moreover, for each $s \in U$, a non-empty set D_s is also fixed. The family $D = \{D_s\}_{s\in U}$ is called *data universe*. In the following example, we define a hidden institution on this data universe.

Example 2.9. The hidden institution on the data universe D, denoted by $\mathcal{H}^D$, is given by:

- the category $SIG_{\mathcal{H}^D}$ is the category whose objects are the hidden signatures defined on a visible signature $V\Sigma = (VS, V\Omega)$ and a data domain $D_{V\Sigma}$, such that $VS \subseteq U$ and $(D_{V\Sigma})_s = D_s$ for each $s \in VS$, and whose morphisms are the hidden signature morphisms;

- the functor $Sen_{\mathcal{H}^D}\colon SIG_{\mathcal{H}^D} \to Set$ gives for each signature the set of algebraic equations;
- the functor $Mod_{\mathcal{H}^D}\colon SIG^{op}_{\mathcal{H}^D} \to Cat$ gives for each signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, the category $HAlg(H\Sigma)$ of hidden algebras,
- the satisfaction relation is the behavioural satisfaction.

The institution $\mathcal{H}^D$ is slightly different from the hidden institution in [3, 14]. In these papers, the category of signatures is defined on a unique fixed visible signature and data domain. Then, this visible signature is exactly the visible part of each hidden signature in the institution. In the hidden institution $\mathcal{H}^D$ a universe of sorts and a data universe for these sorts are considered. Then, hidden signatures are defined on a visible signature and a data domain included on these universes. Note in addition that $\mathcal{H}^D$ is not a generalization of the hidden institution in [14] or [3] because in our hidden signature morphisms the visible parts of the source and the target are identical. This condition suffices for the specification of the data structures presented in EAT. An interesting problem, which is not dealt with in this paper, consists in trying to relax this condition, for example, allowing to define inclusions between sets of visible sorts.

Moreover, an institution morphism from the hidden institution in [3, 14] to the equational algebraic institution is defined in these papers. In the following example we introduce, as a first example of institution encoding, a direct redefinition of that morphism where the source institution is the hidden institution $\mathcal{H}^D$. This encoding is similar to the morphism in [3, 14] since the natural transformation for sentences are identity maps, hence the direction of this natural transformation is unimportant.

Example 2.10. We consider the institution encoding from $\mathcal{H}^D$ to $\mathcal{E}$ defined by:

- the functor between categories of signatures forgets the distinction between visible and hidden sorts;
- for each hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, $\alpha_{H\Sigma}\colon Sen_{\mathcal{H}^D}(H\Sigma) \to Sen_{\mathcal{E}}(H\Sigma)$ is the identity map;
- for each hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, $\beta_{H\Sigma}\colon Mod_{\mathcal{H}^D}(H\Sigma) \to Mod_{\mathcal{E}}(H\Sigma)$ is the functor that gives for a hidden algebra the quotient algebra that is obtained by identifying those elements which are behaviourally equivalent (and for a hidden morphism between two hidden algebras the quotient homomorphism defined by the behavioural congruence).

3. The *imp* construction

We have pointed out that in the EAT system [25] two *layers* of data structures coexist. In a first layer, one finds the usual data structures like integer numbers, (finite) lists, trees of symbols (to represent linear combinations or polynomials) and so on. In the second layer, one must deal with algebraic structures like (graded) groups, rings, simplicial sets or chain complexes, whose elements are data belonging to the first layer. These algebraic structures are first order elements in algebraic topology and then, a system such as EAT does not work with just one of these structures, but handles families of them at runtime.

To carry out the specification of this type of structures, an operation between abstract data types was defined in [21]. This operation models the step from working with a data type to working with a type whose "elements" are families of the former. The operation is called *imp* construction (this name has been chosen because this kind of specifications are related to implementations of structures rather than to the structures themselves, *i.e.*, the treatment at low level that EAT makes of them). Moreover, it was obtained that EAT data structures are final objects of suitable categories of implementations of abstract data types [21]. Therefore, we can conclude that the data representations chosen in EAT to implement its structures is the most "general" possible (in the sense that any other implementation can be seen as a part of the implementation chosen in EAT).

The following simple example is used to explain the syntactic aspects of this construction. Let GRP be a signature with a unique sort g and three operations:

$prd\colon g\ g \to g$
$inv\colon g \to g$
$unt\colon\ \to g$

This signature is obviously the basis of the algebraic specification for a group, whose underlying set is abstracted by the sort g. But if, as it is usual in symbolic computation systems, it is necessary to handle several groups on the same underlying data set, a new sort, which remains hidden in the signature GRP, has to be considered: the type of groups represented on g. If we make explicit this invisible (or hidden) type, we obtain a new signature, that is denoted by $\mathtt{GRP}_{imp}$, with a new sort $imp\_\mathtt{GRP}$, and operations:

$imp\_prd\colon imp\_\mathtt{GRP}\ g\ g \to g$
$imp\_inv\colon imp\_\mathtt{GRP}\ g \to g$
$imp\_unt\colon imp\_\mathtt{GRP}\ \to g$

In general, for a given signature $\Sigma = (S, \Omega)$, a new signature $\Sigma_{imp} = (S_{imp}, \Omega_{imp})$ can be defined as follows:

- $S_{imp} = S \cup \{imp_\Sigma\}$ with $imp_\Sigma \notin S$;
- for each operation $\omega\colon s_1 \dots s_n \to s$ in Ω, an operation $imp\_\omega\colon imp_\Sigma s_1 \dots s_n \to s$ is included in Ω_{imp}.

The $\mathtt{GRP}_{imp}$-algebras satisfy the property that each element of the carrier set for the distinguished sort allows to retrieve a GRP-algebra. To be more precise, given a $\mathtt{GRP}_{imp}$-algebra A, each element $a \in A_{imp\_\mathtt{GRP}}$ defines the GRP-algebra $A_a = \langle A_g, imp\_prd(a, -, -), imp\_inv(a, -), imp\_unt(a)\rangle$. Note that the operations of A_a are obtained by fixing the element a as first argument of the operations of A. In this way, each ADT for GRP (*i.e.*, each category of GRP-algebras closed under isomorphism [23]) has associated an ADT for $\mathtt{GRP}_{imp}$ (those algebras verifying that each GRP-algebra obtained from its "*imp*-elements" belongs to the ADT for GRP). It is clear that this construction can be generalized to any signature Σ_{imp}.

In [22], the *imp* construction was studied from the point of view that a signature and its corresponding "*imp*-signature" belong to two different specification frameworks: the equational algebraic specification for the former and the hidden specification (or the coalgebraic) framework for the latter. In the following section, we will extend this operation to signature morphisms, sentences and models in order to obtain institution morphisms.

4. A diagram of institution encodings to model the *imp* construction

The aim of this section is to extend and to explain in terms of morphisms between institutions the *imp* operation presented in the previous section. At this point, it is necessary to decide what type of morphism is convenient for this situation. In the literature several notions for a relationship between institutions have been developed, and each one of these notions tries to represent a different kind of link between institutions [17]. In our construction, the starting point is given by signatures, sentences and models for a given structure and we get the corresponding components for families of that structure. Thus, we are interested in translating all the components (signatures, sentences, models...) from the equational algebraic context into other specification frameworks. Then, the most suitable notion of institution morphism for this situation is that of institution encoding (the three components of an institution encoding, that is, its functor between signatures and the natural transformations on models and sentences, "go" in the same direction). This notion has been used in a study of behavioural interpretation of specifications by Bidoit and Hennicker in [2]. But, the use of an institution encoding to model an operation between different specification frameworks is, up to the authors' knowledge, new in the literature.

In this section, we are going to define three institution encodings that place our construction in three different specification frames. First, in the equational algebraic institution, second in a hidden institution and finally in a coalgebraic institution. They can be seen as three ways of understanding the *imp* construction. Moreover, if the three morphisms are put together, they form a commutative diagram of institution encodings.

Before establishing the morphisms, in the following subsection we are going to define the source institution of all of them. This institution is a "subinstitution" of the equational algebraic institution and reflects the idea of fixing the data sets for the elements of the algebraic structures to be modelled.

4.1. The source institution

From a programming point of view, when you are dealing with implementations of algebraic structures (groups, for instance), usually you are only interested in structures whose elements share the same syntactic pattern. In this way, the equational algebraic institution is a too general context for this aim. When you translate this restriction to the model level, given a signature, you must restrict yourself to algebras with the same carrier sets. From the equational algebraic institution, an institution that captures this characteristic can be defined.

Let us fix a set U that will be called *sort universe.* For each $s \in U$, a non-empty set D_s is also fixed. The family $D = \{D_s\}_{s \in U}$ is called *data universe.* This data universe was previously considered in Example 2.9 and it is a fixed universe hereinafter in this paper.

The *equational algebraic institution on a data universe* D, denoted by $\mathcal{E}^D$, is given by:

- the category $SIG_{\mathcal{E}^D}$ is the category whose objects are signatures $\Sigma = (S, \Omega)$ with $S \subseteq U$, and whose morphisms are signature morphisms that are the identity on sorts;
- the functor $Sen_{\mathcal{E}^D} \colon SIG_{\mathcal{E}^D} \to Set$ gives for each signature the set of algebraic equations;
- the functor $Mod_{\mathcal{E}^D} : SIG^{op}_{\mathcal{E}^D} \to Cat$ gives for each signature Σ the category whose objects are the Σ-algebras A such that $A_s = D_s$, for each sort s in Σ, and whose morphisms are the identity morphisms;
- for each signature Σ in $SIG_{\mathcal{E}^D}$ the satisfaction relation $\models^{\mathcal{E}^D}_{\Sigma}$ is the restriction of the equational algebraic satisfaction relation to $Obj(Mod_{\mathcal{E}^D}(\Sigma)) \times Sen_{\mathcal{E}^D}(\Sigma)$.

Note that we are fixing the underlying set of a structure but not its operations. The strong condition imposed on signature morphisms implies that it is not possible to change the data domain assigned to a sort in two related signatures. Note that no condition is imposed on the operation component of signature morphisms. Thus, it is possible, for example, to define an inclusion signature morphism between a signature for semigroups and a signature for groups. On the other hand, the category of models of a signature is discrete (thus it is a set, rather than a proper category). This restriction is due to the fact that they will correspond to the visible part of hidden morphisms between hidden algebras.

4.2. A first institution encoding

A first attempt to specify the *imp* construction consists in using as target institution the well-know equational algebraic specification. This leads us to define an institution encoding from $\mathcal{E}^D$, the equational algebraic institution on a data universe D, to $\mathcal{E}$, the equational algebraic institution.

Let $\Phi_{eq} = (\Phi, \alpha, \beta)$ be the institution encoding from $\mathcal{E}^D$ to $\mathcal{E}$, $\Phi_{eq} \colon \mathcal{E}^D \to \mathcal{E}$, defined as follows.

- The functor $\Phi \colon SIG_{\mathcal{E}^D} \to SIG_{\mathcal{E}}$ gives for each signature $\Sigma = (S, \Omega)$ in $SIG_{\mathcal{E}^D}$ the signature Σ_{imp} such that: $\Sigma_{imp} = (S_{imp}, \Omega_{imp} \cup C)$ where $S_{imp} = S \cup \{imp_\Sigma\}$, imp_Σ being a fresh symbol (*i.e.* $imp_\Sigma \notin U$), and $\Omega_{imp} = \{imp\_\omega \colon imp_\Sigma\ s_1 \ldots s_n \to s \mid \omega \colon s_1 \ldots s_n \to s \in \Omega\}$ (*i.e.* the operations of Ω_{imp} are obtained from the operations of Ω by adding the new symbol as first argument sort). The set C contains a constant $d \colon \ \to s$

for each $d \in D_s$ and each $s \in S$[1]. Given a signature morphism $\mu\colon \Sigma \to \Sigma'$ from $\Sigma = (S, \Omega)$ to $\Sigma' = (S', \Omega')$ in $SIG_{\mathcal{E}^D}$, $\Phi(\mu)$ is the signature morphism $\mu_{imp}\colon \Sigma_{imp} \to \Sigma'_{imp}$ such that $\mu_{imp}(s) = s$ for each $s \in S$, $\mu_{imp}(imp_\Sigma) := imp_{\Sigma'}$, if $\omega \in \Omega$ is an operation such that $\mu(\omega) = \omega'$ then $\mu_{imp}(imp\_\omega) := imp\_\omega'$ and, finally, $\mu_{imp}(d) := d$ for each $d \in C$.

- The natural transformation $\alpha\colon Sen_{\mathcal{E}^D} \Rightarrow Sen_{\mathcal{E}} \circ \Phi$ is defined by the family of maps $\alpha_\Sigma\colon Sen_{\mathcal{E}^D}(\Sigma) \to Sen_{\mathcal{E}}(\Sigma_{imp})$, one map α_Σ for each signature $\Sigma = (S, \Omega)$ in $SIG_{\mathcal{E}^D}$. The map α_Σ acts on a sentence in Σ by adding a new variable z_{imp} of sort imp_Σ and replacing each occurrence of each operation ω of Σ by the corresponding operation imp_ω in Σ_{imp} (the sentence is completed by putting the new variable where it is required). To be more precise, if e is an equation $\forall X.t = u$ in $Sen_{\mathcal{E}^D}(\Sigma)$, $\alpha_\Sigma(e)$ is the equation $\forall X \cup \{z_{imp}\}.\rho(t) = \rho(u)$, where $\rho = (\rho_s\colon T_{\Sigma(X)s} \to T_{\Sigma_{imp}(X \cup \{z_{imp}\})s})_{s \in S}$ is the family of maps defined as follows: if $t = x \in X_s$ then $\rho_s(x) := x$, if $t = \omega(t_1, \ldots, t_n)$, with $\omega\colon s_1 \ldots s_n \to s \in \Omega$, $n \geq 0$ and for each $i \in \{1, \ldots, n\}$, $t_i \in T_{\Sigma(X)s_i}$ then $\rho_s(\omega(t_1, \ldots, t_n)) := imp\_\omega(z_{imp}, \rho_{s_1}(t_1), \ldots, \rho_{s_n}(t_n))$.
- The natural transformation $\beta\colon Mod_{\mathcal{E}^D} \Rightarrow Mod_{\mathcal{E}} \circ \Phi^{op}$ is defined by the family of functors $\beta_\Sigma\colon Mod_{\mathcal{E}^D}(\Sigma) \to Mod_{\mathcal{E}}(\Sigma_{imp})$, one functor β_Σ for each signature $\Sigma = (S, \Omega)$ in $SIG_{\mathcal{E}^D}$. Given a Σ-algebra A, $\beta_\Sigma(A)$ is the Σ_{imp}-algebra such that $\beta_\Sigma(A)_{imp_\Sigma} := \{*\}$ (*i.e.* the carrier set for imp_Σ is the singleton $\{*\}$) and $\beta_\Sigma(A)_s := D_s$, for each $s \in S$. The interpretation of the Σ_{imp}-operations in $\beta_\Sigma(A)$ is the natural one:

$$imp\_\omega_{\beta_\Sigma(A)}(*, d_1, \ldots, d_n) = \omega_A(d_1, \ldots, d_n).$$

As we have explained in the previous section, the signature Σ_{imp} can be seen as the signature for families of Σ-algebras. An algebra for the signature Σ_{imp} satisfies the sentence $\alpha_\Sigma(e)$ if and only if every algebra of the family satisfies the sentence e. This fact is formally stated in the following proposition, whose proof is an easy exercise.

Proposition 4.1. *Let $\Sigma = (S, \Omega)$ be a signature. Given a Σ_{imp}-algebra $A = \langle A_{imp_\Sigma}, (D_s)_{s \in S}, (imp\_\omega_A\colon A_{imp_\Sigma} \times D_{s_1} \times \cdots \times D_{s_n} \to D_s)_{\omega\colon s_1 \ldots s_n \to s \in \Omega}, (d_A = d)_{d \in C}\rangle$, for each element $a \in A_{imp_\Sigma}$, we consider the Σ-algebra $A_a = \langle (D_s)_{s \in S}, (imp\_\omega_A(a, -)\colon D_{s_1} \times \cdots \times D_{s_n} \to D_s)_{\omega\colon s_1 \ldots s_n \to s \in \Omega}\rangle$. Then, for each Σ-sentence e in $\mathcal{E}^D$:*

$$A \models^{\mathcal{E}}_{\Sigma_{imp}} \alpha_\Sigma(e) \quad \textit{iff} \quad A_a \models^{\mathcal{E}^D}_{\Sigma} e, \textit{ for each } a \in A_{imp_\Sigma}.$$

Taking into account the previous proposition, the proof of the following is straightforward.

Theorem 4.2. *The triple $\Phi_{eq} = (\Phi, \alpha, \beta)$ defines an institution encoding between $\mathcal{E}^D$ and $\mathcal{E}$.*

[1]Note that we are using the same notation for the "*imp*-signatures" as in [21] and Section 3 above. Nevertheless, we have to introduce here the set of constants in these signatures in order to obtain the institution encoding from the equational algebraic institution on a data universe to a hidden institution, which will be defined later on in this paper.

The basic idea of the above "canonical" encoding is that a Σ-algebra can be encoded as a family of Σ-algebras with only one member. So, from a semantic point of view, the models in the image represent trivial families of algebras. In the following sections, we will redefine this morphism by using a variant of the hidden institution and a particular coalgebraic institution as target institutions. In those frameworks, it will be possible to sum (as a coproduct) these apparent trivial images and this construction will provide us a new Σ_{imp}-algebra. That algebra results to be the final algebra in the category that represents the families of Σ-algebras and it will correspond to the final algebra obtained in [22] as a specification of the EAT data structures. Moreover, the fact included in the previous proposition that an algebra for the signature Σ_{imp} represents a family of Σ-algebras will be expressed below through an institution encoding between the coalgebraic and the hidden institutions which completes a diagram of institution morphisms.

4.3. A more suitable morphism

In the morphism of the previous section, we have bounded the domain. We have considered $\mathcal{E}^D$ as the domain of our institution encoding because this institution accurately reflects the EAT way of working, that is, we fix a syntactic pattern for the elements of the mathematical structures that are been represented in the system. Likewise, it is quite clear that the range $\mathcal{E}$ is too wide: only the Σ_{imp}-algebras based on D are relevant in our approach. There is an institution that allows us to consider this fundamental characteristic: the *hidden institution on D*, $\mathcal{H}^D$ (see [14] and Sect. 2 in this paper). The "*imp*-signatures" will be perfectly integrated into that institution, it will be enough to declare the new sort as a hidden sort. So, we will be able to distinguish between fixed visible data (the elements of the algebraic structures) and hidden objects (the algebraic structures).

Note that the hidden institution $\mathcal{H}^D$ imposes a very strong condition on signature morphisms: the *encapsulation condition*, which is necessary to obtain the satisfaction condition of that institution. But, considering the correspondence between the signature morphisms in the institution morphism which has been defined in the previous section, it is clear that there are signature morphisms in $\mathcal{E}^D$ that cannot be translated into signature morphisms in $\mathcal{H}^D$. A simple example are the inclusions between signatures (a semigroup and a group, for instance).

The "*imp*-signatures" define a very particular type of hidden signatures. They have a unique hidden sort that never appears as target sort in any operation. This type of sorts will be called *deconstructor sorts*. It is important to remark that deconstructor sorts represent *immutable objects*, *i.e.* there are no updating operations that allow a change of state. Thus, the only requirements they admit are *via direct observations* (by using deconstructor operations). For example, when you consider a group as an argument of an operation, you can use the product operation of that group or you can ask if it satisfies the commutative property, but you cannot change its internal structure. This implies that there are no terms of a deconstructor sort apart from the variables, and therefore there are no equations of that sort different from equalities between variables (it is clear that this trivial equations can be excluded without reducing the expressiveness of the institution). In that case, we can skip over the encapsulation condition for a deconstructor sort in order to obtain a hidden institution. This leads us to a definition of hidden signature morphisms with an encapsulation condition which is weaker that the usual one.

Definition 4.3 (deconstructor sort)**.** A hidden sort h in a hidden signature $H\Sigma$ is said to be *deconstructor* if h does not appear as target sort of any operation on $H\Sigma$.

Definition 4.4 (deconstructor hidden signature morphism)**.** Let $H\Sigma$ and $H\Sigma'$ be two hidden signatures, on $V\Sigma$ and $D_{V\Sigma}$. A *deconstructor hidden signature morphism* $\mu\colon H\Sigma \to H\Sigma'$ is a signature morphism such that it is the identity on visible sorts and visible operations, defines an injective map on hidden sorts whose image by the signature morphism are not deconstructor sorts and if $\omega'\colon s'_1 \dots s'_n \to s'$ belongs to $H\Omega'$, some sort in $\{s'_1, \dots, s'_n\}$ lies in $\mu(HS)$ and this sort is not a deconstructor sort then $\omega' = \mu(\omega)$ for some $\omega \in H\Omega$.

Thus, a deconstructor hidden signature morphism is a hidden signature morphism such that it is not required that a deconstructor sort whose image by the signature morphism is a deconstructor sort satisfies the encapsulation condition. Moreover, the one-to-one condition on these deconstructor sorts is not required either.

Now, we can integrate this weaker encapsulation condition into the definition of the hidden institution in Example 2.9. In this way, we obtain a new hidden institution, which will be denoted by $\bar{\mathcal{H}}^D$, given by:

- the category $SIG_{\bar{\mathcal{H}}^D}$ is the category of hidden signatures on a data universe D with deconstructor hidden signature morphisms;
- the functor $Sen_{\bar{\mathcal{H}}^D}\colon SIG_{\bar{\mathcal{H}}^D} \to Set$ gives for each hidden signature $H\Sigma$ the set of equations $Sen_{\mathcal{H}^D}(H\Sigma)$ except the equalities between two hidden variables;
- the categories of models and the satisfaction relations are defined as in the institution $\mathcal{H}^D$.

Now, the institution encoding Φ_{eq} from $\mathcal{E}^D$ to $\mathcal{E}$ defined in the previous section can be used as a guide in order to obtain a morphism, denoted by Φ_{hd}, which has as target institution $\bar{\mathcal{H}}^D$ instead of $\mathcal{E}$. The morphism Φ_{hd} is given by:

- The functor between the categories of signatures gives for each signature Σ in $\mathcal{E}^D$ the signature Σ_{imp}, being the distinguished sort imp_Σ (which does not belong to the universe sort U) the unique hidden sort. With regard to signature morphisms, the image of a morphism $\mu\colon \Sigma \to \Sigma'$ in $\mathcal{E}^D$ is the corresponding morphism $\mu_{imp}\colon \Sigma_{imp} \to \Sigma'_{imp}$. Since the hidden sort of an "*imp*-signature" is deconstructor, it is clear that μ_{imp} is a deconstructor signature morphism.
- The natural transformations on sentences and models are defined as the corresponding components in Φ_{eq}. Note that a sentence is never applied to an equation of hidden variables.

If we try to redefine the institution morphism from $\mathcal{H}^D$ to $\mathcal{E}$ in Example 2.10 in order to obtain a morphism from $\bar{\mathcal{H}}^D$ to $\mathcal{E}$, we cannot use the behavioural congruence. This is due to the weaker notion of hidden signature morphism. Nevertheless, it is possible to define a "stronger" behavioural congruence where the identification on models in the carrier set for a deconstructor sort is lost: the equality is adopted in these sorts.

Definition 4.5 (deconstructor behavioural equivalence)**.** Let $H\Sigma = (S, \Omega)$ be a hidden signature, on $V\Sigma$ and $D_{V\Sigma}$, and let A be a hidden $H\Sigma$-algebra. Two elements $a, a' \in A_s$, with $s \in S$, are *deconstructor behaviourally equivalent* iff $a = a'$ if s is a deconstructor sort and a is behaviourally equivalent to a' otherwise.

This new definition of behavioural equivalence can seem unexpected for a "hidden" institution since for deconstructor sorts behavioural equivalence is not considered. The reason for this constraint is there are no useful equations in our context for these very particular sorts. Then, behavioural equivalence is not suitable for our applications. A more detailed discussion about this point will be presented below and in Section 5.

Now, it is trivial to see that the deconstructor behavioural equivalence is a congruence. Now, we can define an institution encoding, which can be denoted by Φ_{pr}, from $\bar{\mathcal{H}}^D$ to $\mathcal{E}$ as follows:

- the functor between categories of signatures is defined as the corresponding functor in the morphism in Example 2.10;
- for each hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, $\alpha_{H\Sigma} \colon Sen_{\bar{\mathcal{H}}^D}(H\Sigma) \to Sen_{\mathcal{E}}(H\Sigma)$ is the inclusion map (equalities between hidden sorts have been excluded);
- for each hidden signature $H\Sigma$, on $V\Sigma$ and $D_{V\Sigma}$, $\beta_{H\Sigma} \colon Mod_{\bar{\mathcal{H}}^D}(H\Sigma) \to Mod_{\mathcal{E}}(H\Sigma)$ is the functor that gives for a hidden algebra the quotient algebra given by the deconstructor behavioural congruence relation and for a hidden morphism the corresponding quotient homomorphism.

It is important to note that the condition in a deconstructor hidden signature morphism, that establishes that the deconstructor sorts whose images by the morphism are non deconstructor sorts must verify the encapsulation condition, is only necessary to define the morphism Φ_{pr}. Nevertheless, it is possible to define a hidden institution such that it does not include this condition. The result is an institution more general than $\bar{\mathcal{H}}^D$. Besides, an institution encoding from $\mathcal{E}^D$ to this last institution is possible too.

The interpretation that can be done of the institution encoding Φ_{hd} is the following: the objects in $Mod_{\mathcal{E}^D}(\Sigma)$ are algebraic structures defined on D (that is, whose underlying set is D) and the objects in $Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$ are indexed families of such structures, where the elements of the carrier sets for the hidden sort imp_Σ act as *indexes* for the algebraic structures of the family. A hidden algebra in $Mod_{\mathcal{E}^D}(\Sigma)$ which is the image of an algebra in $\mathcal{E}^D$ by the morphism Φ_{hd} corresponds with a family of algebras which only has one algebra. This hidden algebra has as carrier set for the only deconstructor sort a singleton which is used as a pointer to the algebra. Besides, in the specification on the hidden algebra none equation of the deconstructor sort is defined and then the standard behavioural equivalence is not used on this sort, as explained above, and it is interpreted as equality (according to Def. 4.5). So, from a semantic point of view, the models in the image represent trivial families of algebras. But this apparent trivial families can be "summed" (as a coproduct) in order to obtain a new hidden algebra which is the final algebra in our category of hidden algebras. This final algebra in $Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$ corresponds with the final objects proposed in [22] to model the EAT data structures. In this new context, we obtain a formalization of a coding of a set of Σ-algebras defined on D by a single Σ_{imp}-algebra.

Note that in order to build this institution encoding we could define an "*ad-hoc*" institution as target institution with the required characteristics for our construction. This could be a variant of the equational institution with two types of sorts: one with a fixed interpretation and another one with a loose interpretation. Another possibility is to modify the hidden institution to reach the same goal. Between these two options we have preferred to follow the hidden approach since the role of visible and hidden is exactly the one we need. Another minor point is the fact that a data domain is fixed with visible sorts, and this accurately to our concrete application field. In addition, these points, which should be introduced anew in an ad-hoc modification of the standard algebraic institution, are widely presented in the literature [3, 14].

The existence or not of final objects in "hidden categories" has been widely studied in the literature. For instance, a description of these final objects is given through a "magical formula" in [15]. Nevertheless, in our very particular case, the final object in $Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$ admits an intuitive description. In our case, the elements of its carrier set for the hidden sort are functional tuples, where each tuple encodes the functions of one model in $\mathcal{E}^D$ (see [21] for a more detailed explication). This result was suggested by the way chosen by Sergeraert to develop EAT, using intensively functional programming (see [21, 25]). In this program, an element of the "*imp* sort" is encoded by means of a record of Common Lisp functions which has a field for each operation in Σ. The elements of each algebraic structure (which is represented by an instance of the record) are elements of D (which is fixed), or constructed from initial semantics, and so they do not require an explicit storage. When each particular implementation (*i.e.*, each particular instance of the record) is summed, a practical interpretation of the following result is obtained. It shows that the models in the image of the institution encoding $\Phi_{hd} = (\Phi, \alpha, \beta)$ allows us to build that final object. It suffices to consider the coproduct of the objects that belong to the image of this morphism.

Theorem 4.6. *Given a signature Σ in $\mathcal{E}^D$, the coproduct in $Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$ of the objects in $\beta_\Sigma(Mod_{\mathcal{E}^D}(\Sigma))$ is the final object in $Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$.*

The proof of this theorem is easy taking into account that D is fixed.

4.4. A third institution encoding

The third formalism we are going to consider is that of *coalgebras*. The coalgebraic approach provides a general framework in which object-oriented features can be studied in a very close way to hidden specifications (see [27] or [6] for instance).

Definition 4.7 (coalgebra)**.** Let $F\colon Set \to Set$ be an endofunctor of Set. A *F-coalgebra* is a couple (X, c), where X is a set and $c\colon X \to F(X)$ is a map. A *morphism between two F-coalgebras* (X, c) and (Y, d) is a map $f\colon X \to Y$ such that $F(f) \circ c = d \circ f$.

The F-coalgebras, together with morphisms between them, define a category which will be denoted by $CoAlg(F)$.

In order to include coalgebras in our institutional framework, we are going to introduce the concept of *coalgebraic institution associated to an institution.*

Let $\mathcal{I} = (SIG_{\mathcal{I}}, Sen_{\mathcal{I}}, Mod_{\mathcal{I}}, \models^{\mathcal{I}})$ be an institution. We assume that for each signature $\Sigma \in Obj(SIG_{\mathcal{I}})$, the category $Mod_{\mathcal{I}}(\Sigma)$ is small (in consequence, each *class* $Obj(Mod_{\mathcal{I}}(\Sigma))$ is a set). We define the *coalgebraic institution associated to* $\mathcal{I}$, denoted by $CoAlg(\mathcal{I})$, as follows:

- the category $SIG_{CoAlg(\mathcal{I})}$ is $SIG_{\mathcal{I}}$;
- the functor $Sen_{CoAlg(\mathcal{I})}$ is $Sen_{\mathcal{I}}$;
- the functor $Mod_{CoAlg(\mathcal{I})}\colon SIG^{op}_{CoAlg(\mathcal{I})} \to Cat$ is defined by: $Mod_{CoAlg(\mathcal{I})}(\Sigma) := CoAlg(F_\Sigma)$ for each object Σ in $SIG^{op}_{CoAlg(\mathcal{I})}$, where F_Σ is the endofunctor of the category Set that is constant on $Obj(Mod_{\mathcal{I}}(\Sigma))$[2]. With regard to the morphisms, if $\mu\colon \Sigma \to \Sigma'$ is a morphism in $SIG^{op}_{CoAlg(\mathcal{I})}$ and (X', c') is a $F_{\Sigma'}$-coalgebra then $Mod_{CoAlg(\mathcal{I})}(\mu)((X', c'))$ is the F_Σ-coalgebra $(X', Mod_{\mathcal{I}}(\mu) \circ c')$. Moreover, given a $F_{\Sigma'}$-morphism $f\colon X' \to Y'$ between two $F_{\Sigma'}$-coalgebras (X', c') and (Y', d'), the same map f defines a F_Σ-morphism between $Mod_{CoAlg(\mathcal{I})}(\mu)((X', c'))$ and $Mod_{CoAlg(\mathcal{I})}(\mu)((Y', d'))$.
- for a given signature Σ, the satisfaction relation $\models^{CoAlg(\mathcal{I})}_{\Sigma}$ is defined by:

$$(X, c) \models^{CoAlg(\mathcal{I})}_{\Sigma} e \quad \text{iff} \quad c(x) \models^{\mathcal{I}}_{\Sigma} e,\ \forall x \in X,$$

  where $(X, c\colon X \to Obj(Mod_{\mathcal{I}}(\Sigma)) \in CoAlg(F_\Sigma)$ and $e \in Sen_{CoAlg(\mathcal{I})}(\Sigma)$.

This particular institution tries to represent, in a coalgebraic language, the institution for the families of models of the former institution. Note that a coalgebra (X, c) is just a family of algebras labelled by elements of X.

When each morphism in the categories of $\mathcal{I}$-models is an endomorphism, we can define a canonical institution encoding from $\mathcal{I}$ to $CoAlg(\mathcal{I})$. This morphism is given by:

- the identity functor between the categories of signatures;
- for each signature Σ, $\alpha_\Sigma\colon Sen_{\mathcal{I}}(\Sigma) \to Sen_{CoAlg(\mathcal{I})}(\Sigma)$ is the identity map;
- for each signature Σ, $\beta_\Sigma\colon Mod_{\mathcal{I}}(\Sigma) \to Mod_{CoAlg(\mathcal{I})}(\Sigma)$ is the functor giving for each Σ-algebra $A \in Obj(Mod_{\mathcal{I}}(\Sigma))$ the F_Σ-coalgebra $(\{*\}, i_A\colon \{*\} \to Mod_{\mathcal{I}}(\Sigma))$, defined by $i_A(*) = A$, that is, the coalgebra that represents the family of Σ-algebras with a single element, the

[2]Note that $F_\Sigma\colon Set \to Set$ is not constant on a particular object of $Mod_{\mathcal{I}}(\Sigma)$, but it is constant on the whole $Obj(Mod_{\mathcal{I}}(\Sigma))$, since, $Mod_{\mathcal{I}}(\Sigma)$ being a small category, $Obj(Mod_{\mathcal{I}}(\Sigma))$ is simply a set.

algebra A. Note that the unique morphism between the F_Σ-coalgebras $\beta_\Sigma(A) = (\{*\}, i_A)$ and $\beta_\Sigma(B) = (\{*\}, i_B)$, with $A, B \in Mod_{\mathcal{I}}(\Sigma)$, is the identity map $1_{\{*\}}$. So, in order to obtain that the functors β_Σ are well-defined it is necessary that in $Mod_{\mathcal{I}}(\Sigma)$ there exist only endomorphims.

It is straightforward to check that $CoAlg(\mathcal{I})$ is an institution and that the above mappings define an institution encoding.

Theorem 4.8. *Let $\mathcal{I}$ be an institution so that for each $\Sigma \in Obj(SIG_{\mathcal{I}})$, $Mod_{\mathcal{I}}(\Sigma)$ is a small category. Then $CoAlg(\mathcal{I})$ is an institution. Moreover, if each morphism in $Mod_{\mathcal{I}}(\Sigma)$ is an endomorphism, then there exists a canonical institution encoding between $\mathcal{I}$ and $CoAlg(\mathcal{I})$.*

If this construction is particularized to $\mathcal{I} := \mathcal{E}^D$, the equational algebraic institution on D (this is possible because all the categories of algebras in $\mathcal{E}^D$ are small, since D is fixed, and the morphisms between models are the identities), we obtain the corresponding coalgebraic institution $CoAlg(\mathcal{E}^D)$ and an institution encoding from $\mathcal{E}^D$ to $CoAlg(\mathcal{E}^D)$. This morphism will be denoted by Φ_{CoAlg}.

The morphism Φ_{CoAlg} allows us to recover the final objects of the categories of coalgebras $CoAlg(F_\Sigma)$. For this particular functor (see [27] for general results on finality in categories of coalgebras), the final coalgebra can be described as the F_Σ-coalgebra $(Obj(Mod_{\mathcal{E}^D}(\Sigma)), id_{Obj(Mod_{\mathcal{E}^D}(\Sigma))})$. From a theoretical perspective, this is not a relevant result. However, this coalgebra is a suitable idealization of the representation used in EAT for the families of mathematical structures: the set in the coalgebra acts as an *index* for the algebraic structures of the family (this representation has been briefly explained in the previous section; we refer again to [25] and [21] for a detailed description). Taking into account our institution encoding $\Phi_{CoAlg} = (\Phi, \alpha, \beta)$, the final object can be described in the following way:

Theorem 4.9. *Given a signature Σ in $\mathcal{E}^D$, the coproduct in $CoAlg(F_\Sigma)$ of the objects in $\beta_\Sigma(Mod_{\mathcal{E}^D}(\Sigma))$ is the final object in $CoAlg(F_\Sigma)$.*

4.5. A diagram of institution encodings

The three institution encodings previously introduced show three ways of explaining the *imp* construction which is the central component of the formal specification of the structures in the EAT system. In addition, the relations among them are expressed in the following commutative diagram:

$$\begin{tikzcd}
 & CoAlg(\mathcal{E}^D) \arrow[d, "\Phi_i"] \\
\mathcal{E}^D \arrow[ur, "\Phi_{CoAlg}"] \arrow[r, "\Phi_{hd}"] \arrow[dr, "\Phi_{eq}"] & \bar{\mathcal{H}}^D \arrow[d, "\Phi_{pr}"] \\
 & \mathcal{E}
\end{tikzcd}$$

The only morphism that has not been defined yet is the institution encoding, denoted by Φ_i, from $CoAlg(\mathcal{E}^D)$ to $\bar{\mathcal{H}}^D$. This morphism tries to formalize the relationship between our particular coalgebras and hidden algebras [6] using an institutional language. This morphism is defined as follows:

- the functor between the categories of signatures and the natural transformation between the sets of sentences are defined as the respective components in the morphism Φ_{hd};
- for each signature $\Sigma = (S, \Omega)$ in $SIG_{CoAlg(\mathcal{E}^D)}$, $\beta_\Sigma \colon Mod_{CoAlg(\mathcal{E}^D)}(\Sigma) \to Mod_{\bar{\mathcal{H}}^D}(\Sigma_{imp})$ is the functor giving for each F_Σ-coalgebra $(X, c) \in Obj(Mod_{CoAlg(\mathcal{E}^D)}(\Sigma))$ the hidden algebra $\beta_\Sigma((X, c))$ defined by the carrier set $\beta_\Sigma((X, c))_{imp_\Sigma} = X$ and the function $imp\_\omega_{\beta_\Sigma((X,c))}(a, d_1, \ldots, d_n) := \omega_{c(a)}(d_1, \ldots, d_n)$ for each hidden operation $imp\_\omega \colon imp_\Sigma s_1 \ldots s_n \to s \in \Omega_{imp}$, $n \geq 0$, each $a \in X$ and $d_i \in D_i$, $\forall i = 1, \ldots, n$. A morphism $f \colon X \to Y$ between the coalgebras (X, c) and (Y, c) is mapped to the hidden morphism between the corresponding hidden algebras defined by f, that is, $f_{imp_\Sigma} = f$ and $f_s = 1_s$ for each sort s of Σ.

The satisfaction condition of this morphism corresponds to the fact expressed in Proposition 4.1, *i.e.* a (hidden) algebra $\beta_\Sigma((X, c))$ for the signature Σ_{imp} satisfies the sentence $\alpha_\Sigma(e)$ if and only if every algebra of the family (X, c) (in the coalgebra) satisfies the sentence e. So, we obtain the following theorem.

Theorem 4.10. *The triple $\Phi_i = (\Phi, \alpha, \beta)$ defines an institution encoding between $CoAlg(\mathcal{E}^D)$ and $\bar{\mathcal{H}}^D$.*

Now, in the diagram the role of the two data layers in EAT becomes clear. The set D fixes a pattern for data of the first layer, which are used as elements of the algebraic structures in the second data layer specified in $\mathcal{E}^D$. The hidden sort in $\bar{\mathcal{H}}^D$ is used to specify families of structures in the second data layer. The morphism Φ_{hd} formalized the relation between the two layers, where they are perfectly distinguished as visible and hidden data. If we forget this distinction, we obtain, by composition, the morphism Φ_{eq}. Finally, the morphism Φ_{CoAlg} is based on the description in coalgebraic language of those hidden algebras that represent families in the second data layer.

In [10] we tried to introduce a similar diagram with the usual notion of institution morphism (non forward, *i.e.* where sentences are translated contravariantly with respect to models). For example, to define an institution morphism from $\mathcal{E}^D$ to $\bar{\mathcal{H}}^D$, we must give a natural transformation from the sentences in the hidden institution to the sentences in the equational institution. Each sentence for a hidden signature Σ_{imp} has the particularity of including only variables as *subterms* of the hidden sort. Then, from a Σ_{imp}-sentence (different from an equality of hidden variables) we can define a Σ-sentence by eliminating all the hidden variables and replacing each occurrence of an operation imp_ω with the corresponding operation ω in Σ. (Here it is necessary that each model of $\mathcal{E}^D$ includes as constants the elements of the carrier set of D. This could have been proposed in the definition of the institution $\mathcal{E}^D$, but it is not needed in the construction of the institution encodings in the previous sections.) If this technique is intended to be used in the relationship between the coalgebraic institution and the hidden institution, we find that it is only possible if the hidden institution is restricted to equations in one hidden variable. This detail, that was overlooked in [10], expresses the relationship between hidden algebras and coalgebras included in [6].

5. Relation with other hidden or coalgebraic institutions

This section is devoted to establish relationships among the institutions that have been used in this paper and other hidden or coalgebraic institutions developed in the literature. First, two institutions that could be used instead of our hidden institution are briefly presented. Later, the links between our particular coalgebraic institution and other institutions for coalgebras are analyzed. We are not going to define these institutions in detail but we will only give some ideas about their components. We refer the interested reader to the cited references.

The hidden institution defined in [16] is a generalization of that included in [14] in at least two ways. On the one hand, it allows operations with multiple hidden arguments. This characteristic makes that the categories of hidden algebras lose the existence of final algebra. Obviously, this is not suitable for us. On the other hand, it tries to include a weaker encapsulation condition. In particular, a subset of operations, called *behavioural operations*, is distinguished and then, only these operations are used as *observers*. In this case, to define an institution, it is only required that the behavioural operations satisfy the encapsulation condition. In this framework, the specifier has to decide which operations are behavioural.

It has been remarked in Section 4.3 that there are no equations of deconstructor hidden sorts (different from equalities between variables), so the operations of these sorts do not take part as observers for any equation. Then, there is no difference (with regard to observation) if they are declared as behavioural operations or not. Therefore, to avoid the encapsulation condition, it is advisable to consider (as a rule for the specifier) this type of operations as non behavioural. With this rule, this institution can be considered as a generalization of our hidden institution.

In [19], an institution for *observational logic* is defined. The signatures of these institutions have a distinguished subset of sorts that are called *observable sorts* and include, apart from sorts and operations, a new set of pairs (op, i) called *observers*, where $op\colon s_1 \dots s_n \to s$ is an operation and i is a number that marks a non observable argument sort of the operation. An observer (op, i) is said to be a *direct observer* of s_i if s is an observable sort. The *observable contexts* are terms of observable sort built from observers with a unique distinguished variable of the sort which is intended to be observed. Now, a satisfaction relation between (first order) equations and algebras can be defined with these contexts (in a similar way to behavioural satisfaction). Let us note that a signature morphism in this institution must verify a condition similar to the encapsulation condition for the observers (again, this condition is necessary in order to obtain the satisfaction property on equations of non observable sorts).

In our case, a hidden signature with a unique *deconstructor sort* can be described as an observational signature by declaring this sort as non observable. Then, there is no difference if a direct observer is defined or not for each operation having this non observable sort as argument. It is because there are no terms of this non observable sort (different from variables) and so this direct observations do not take part in the observation of any term. Therefore, they should not be defined as observers in order to avoid the encapsulation condition. Then, this kind of institution could be used in our construction instead of our hidden institution.

Coalgebraic institutions are considered in several works such as [20] or [7]. The institution for observational specifications described in [20] has as signatures tuples of two functors (Ω, Ξ), an algebraic functor and a coalgebraic one. The models are pairs of structures algebra-coalgebra such that algebraic operations are *compatible* with coalgebraic observations. Signature morphisms must verify an encapsulation condition on the coalgebraic part. Finally, the satisfaction relation (using first order equations) is defined by the notion of observational satisfaction with respect to the coalgebraic part. An interesting comment included in this work is that the proof of the satisfaction condition of that institution suggests the authors that generalizations of this notion of signature morphism are possible.

We can obviously define an institution encoding from our coalgebraic institution $CoAlg(\mathcal{E}^D)$ to this institution. For each signature $\Sigma \in SIG_{CoAlg(\mathcal{E}^D)}$, the image of Σ has as coalgebraic part the constant functor $\Xi\colon Set \to Set$ such that $\Xi(X) = \Pi_{\sigma\colon \omega \to v \in \Sigma} D_v^{D_\omega} \simeq Obj(Mod_{\mathcal{E}^D}(\Sigma))$, and as algebraic part the functor *void*, that is, $\Omega(X) = \emptyset$ for each set X. The basic idea is to consider a coalgebra as a pair algebra-coalgebra such that the algebraic component is void. Nevertheless, if we try to extend this definition to signature morphisms, we obtain that the encapsulation condition of the target institution prevents the definition. To solve this problem, we claim that for the particular case of a "state" sort [20] with a constant coalgebraic functor and an algebraic functor void, *i.e.* it is a *deconstructor* sort, it is possible to forget the encapsulation condition for this sort and thus a slight generalization of the signature morphisms is obtained. With this slight modification, the institution encoding can be easily completed.

The coalgebraic institution developed in [7] can be regarded, to a certain extent, as a generalization of the approach included in [8]. It allows for observers whose result type is structured as a coproduct of basic types. Since we are not interested in such result types, we are not going to study the relation of our approach with this institution and we prefer analyzing the interesting approach of [8].

In [8], a coalgebraic institution is not defined but it contains enough tools to make it. The signatures are *coalgebraic* signatures with a unique hidden sort (and some visible sorts), and a set of operations with (at least) the hidden sort as argument sort. The operations are divided in methods (with the hidden sort as target sort) and attributes. Besides, the signatures include a function that maps each visible sort to a non-empty set which represents a fixed *interpretation* of the visible sort in each signature. A model of a given signature consists of a set (the carrier set for the hidden sort) and functions for the operations. The morphisms between two algebras are defined in the usual way. The unconventional point is that the sentences for a signature are pairs of observations (or constants of visible sort). An *observation* is a term of "visible sort" that contains exactly one occurrence of a variable, which must be of hidden sort. These equations are valid in a model if the two observations give the same result for all the states of the carrier. Since there are only equations of visible sort, the satisfaction relation is actually the equational algebraic one.

It is easy to complete the above definitions in order to obtain a coalgebraic institution. It is important to note that since there are no hidden equations, the encapsulation condition is not necessary. This is exactly the characteristic that appears in our coalgebraic institution: there are no equations of the implicitly hidden sort. In the first case, since equations are not allowed, the result of an observation cannot be of hidden sort; in the second one, they are not feasible because only direct observations are possible on deconstructor sorts. Now, an institution morphism (non forward) can be defined in a natural way from our coalgebraic institution to this new coalgebraic institution. It is not forward because in our institution there are more equations than the direct observational ones. In particular, terms with multiple (implicit) occurrences of the state variable can appear in an equation. In [8], this type of sentences are proposed to be included in a generalization of the framework via conditional equations schemes.

6. Conclusions and further work

In this paper we complete the analysis of the specification of the complex structure system of the EAT program. This specification required a construction between abstract data types, called *imp* operation. This operation relates different specification frameworks, such as the equational algebraic specification and other object-oriented specification frameworks (namely the hidden specification and the coalgebraic frameworks). We extend here this operation to form three institution encodings that show three views of the construction situated in an equational algebraic institution, in a hidden institution or in a coalgebraic institution. These morphisms are presented in a commutative diagram which establishes relationships among these specification frameworks. Besides, these morphisms are used to obtain a new description of the final object in these frameworks, which is a suitable model of the implementation of EAT structures.

Trying to do an applied research using the specification techniques developed in the literature, in order to obtain a model for our particular structures, some new constructions have been found. These constructions suppose slight generalizations of some existing techniques or particular, but significant, cases of others. For example, we give a new definition of the *encapsulation condition* for a hidden institution, a coalgebraic institution associated to an institution and a new institutional relationship between this coalgebraic institution and a hidden institution.

In this work, we have found some difficulties in the direct use of the institutional machinery in the formalization of actual symbolic computation systems and we have shown how these problems can be tackled. Our study has pointed out this approach suffers from the gap between the generality of the institutional framework and the particularity of the algebraic structures which are under consideration. The consequence of this, assuming that one chooses the use of institutions, is that the institutional framework is stretched in such a way that in some cases we need to introduce "*ad-hoc*" institutions and in other cases we use degenerated parts of other institutions.

This work can be continued in at least two ways. On the one hand, the institutional description could be extended to other parts of the EAT system or introduced into the specification of structures in the Kenzo system [11], an object-oriented Sergeraert's program (successor of EAT). For example, in [10] the specification of the functorial relationship between data structures of EAT is studied and in [9] the inheritance relationship between Kenzo structures is tackled, although both works lack an institutional description (if such a thing is possible). Besides, it should be established whether the theoretical results obtained here can be applied to other symbolic computation systems apart from EAT or Kenzo. On the other hand, our construction can be studied using other techniques different from institutions. For example, we could use Duval's diagrammatic specifications [12] in order to obtain a better understanding of our job and significant examples in this theory.

Acknowledgements. We wish to thank the referees for valuable comments which have improved the final version of this paper.

References

[1] M. Barr and Ch. Wells, *Category Theory for Computer Science*. Prentice Hall International (1995).

[2] M. Bidoit and R. Hennicker, *Constructor-based observational logic*. Technical Report LSV-03-9, Lab. Specification et Verification, ENS de Cachan, Cachan, France (2003).

[3] R.M. Burstall, R. Diaconescu, *Hiding and behaviour: an institutional approach*, in A Classical Mind: Essays in Honour of C.A.R. Hoare, edited by A. William Roscoe. Prentice-Hall, Englewood Cliffs, NJ (1994) 75–92.

[4] J. Calmet and I.A. Tjandra, A unified-algebra-based specification language for symbolic computing, in *Design and Implementation of Symbolic Computation Systems (DISCO'93)*, edited by A. Miola, Springer, Berlin. *Lect. Notes Comput. Sci.* **722** (1993) 122–133.

[5] J. Calmet, K. Homann and I.A. Tjandra, Unified domains and abstract computational structures, in Artificial Intelligence and Symbolic Mathematical Computation (AISMC'92), edited by J. Calmet and J.A. Campbell, Springer, Berlin. *Lect. Notes Comput. Sci.* **737** (1993) 166–177.

[6] C. Cîrstea, Coalgebra semantics for hidden algebra: parameterised objects and inheritance, in *Recent Trends in Algebraic Development Techniques*, edited by F. Parisi-Presicce, Springer, Berlin. *Lect. Notes Comput. Sci.* **1376** (1998) 174–189.

[7] C. Cîrstea, A coalgebraic equational approach to specifying observational structures. *Theoret. Comput. Sci.* **280** (2002) 35–68.

[8] A. Corradini, A completeness result for equational deduction in coalgebraic specification, in Recent Trends in Algebraic Development Techniques, edited by F. Parisi-Presicce, Springer, Berlin. *Lect. Notes Comput. Sci.* **1376** (1998) 190–205.

[9] C. Domínguez, J. Rubio, Modeling inheritance as coercion in a symbolic computation system, in *International Symposium on Symbolic and Algebraic Computation (ISSAC'2001)*, edited by B. Mourrain, ACM Press (2001) 107–115.

[10] C. Domínguez, L. Lambán, V. Pascual and J. Rubio, Hidden specification of a functional system, in Computer Aided Systems Theory (EUROCAST'2001), edited by R. Moreno-Díaz, B. Buchberger, J.L. Freire, Springer, Berlin. *Lect. Notes Comput. Sci.* **2178** (2001) 555–569.

[11] X. Dousson, F. Sergeraert and Y. Siret, The Kenzo program, Institut Fourier, Grenoble, (1999), Available at `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo`

[12] D. Duval, Diagrammatic Specifications. *Math. Structures Comput. Sci.* **13** (2003) 857–890.

[13] J.A. Goguen and R.M. Burstall, Institutions: Abstract model theory for specification and programming. *J. ACM* **39** (1992) 95–146.

[14] J.A. Goguen and R. Diaconescu, Towards an algebraic semantics for the object paradigm, in Recent Trends in Data Type Specification, edited by H. Ehrig and F. Orejas Springer, Berlin. *Lect. Notes Comput. Sci.* **785** (1994) 1–29.

[15] J.A. Goguen and G. Malcolm, A hidden agenda. *Theoret. Comput. Sci.* **245** (2000) 55–101.

[16] J.A. Goguen, G. Roşu, Hiding more of hidden algebra, in Formal Methods (FM'99), edited by J.M. Wing, J. Woodcook, J. Davies, Springer, Berlin. *Lect. Notes Comput. Sci.* **1709**, (1999) 1704–1719.

[17] J.A. Goguen and G. Roşu, Institution morphisms. *Form. Asp. Comput.* **13** (2002) 274–307.

[18] J.A. Goguen, G. Malcolm and T. Kemp, A hidden herbrand theorem: combining the object and logic paradigms. *J. Log. Algebr. Program.* **51** (2002) 1–41.

[19] R. Hennicker and M. Bidoit, Observational logic, in Algebraic Methodology and Software Technology (AMAST'98), edited by A.M. Haeberer, Springer, Berlin. *Lect. Notes Comput. Sci.* **1584** (1999) 263–277.

[20] A. Kurz and R. Hennicker, On institutions for modular coalgebraic specifications. *Theoret. Comput. Sci.* **280** (2002) 69–103.

[21] L. Lambán, V. Pascual and J. Rubio, Specifying implementations, in *International Symposium on Symbolic and Algebraic Computation (ISSAC'99)*, edited by S. Dooley. ACM Press, (1999) 245–251.

[22] L. Lambán, V. Pascual and J. Rubio, An object-oriented interpretation of the EAT system, Applicable Algebra in Engineering, *Comm. Comput.* **14** (2003) 187–215.

[23] J. Loeckx, H.D. Ehrich and M. Wolf, *Specification of Abstract Data Types.* Wiley and Teubner, New York (1996).

[24] J. Rubio, F. Sergeraert, Constructive algebraic topology. *Bull. Sci. Math.* **126** (2002) 389–412.

[25] J. Rubio, F. Sergeraert and Y. Siret, EAT: Symbolic Software for Effective Homology Computation, Institut Fourier, Grenoble, 1997. Available at `ftp://fourier.ujf-grenoble.fr/pub/EAT`

[26] J. Rubio, F. Sergeraert and Y. Siret, Overview of EAT, a System for Effective Homology Computation. *The SAC Newsletter* **3** (1998) 69–79.

[27] J.J.M.M. Rutten, Universal coalgebra: a theory of systems, *Theoret. Comput. Sci.* **249** (2000) 3–80.

[28] A. Tarlecki, Towards heterogeneous specifications, in Frontiers of Combinig Systems (FroCos'98), Research Studies Press/Wiley, edited by D.M. Gabbay, M. de Rijke. *Stud. Logic Comput.* **7** (2000) 337–360.

Communicated by W. Thomas.
Received September 24, 2004. Accepted September 11, 2006.