DIAGRAMMES

GILLES BERNOT

# Good functors... are those preserving philosophy !

*Diagrammes*, tome 22 (1989), p. 15-29
<http://www.numdam.org/item?id=DIA_1989__22__15_0>

ACTES DES JOURNEES E.L.I.T.
(UNIV. PARIS 7, 27 JUIN-2 JUILLET 1988)

# GOOD FUNCTORS ... ARE THOSE PRESERVING PHILOSOPHY !

**Gilles BERNOT**

L.I.E.N.S.
U.R.A. C.N.R.S. 1327
Ecole Normale Supérieure
45 Rue d'Ulm
75230 Paris Cedex 05
FRANCE

**Abstract :**

Le but de cet article est de mettre en garde le chercheur en types abstraits algébriques contre une utilisation abusive de la théorie des catégories. Quelques propriétés peu souhaitables du (pourtant classique) *foncteur de synthèse* sont décrites, spécialement si l'on s'intéresse aux sémantiques dites "loose". Tous les résultats énoncés ici sont particulièrement simples, sinon triviaux ; néanmoins, ils illustrent des faits donnant lieu à de nombreuses erreurs dans le cadre des types abstraits algébriques. Ces erreurs résultent souvent d'une inadéquation entre certains outils catégoriques bien connus et le concept informatique que l'on souhaite modéliser. Enfin, une approche hiérarchique fondée sur la catégorie des modèles "protégeant les sortes prédéfinies" est proposée, et les premières propriétés en sont dégagées.

**Mots clés** : complétude, consistance, modèle initial, spécifications abstraites, spécifications structurées, théorie des catégories, types abstraits algébriques.

## 1. Introduction

In the following pages, we focus our attention on results which seem to be "trivially ensured" in the basic abstract data type framework. We sometimes give proofs ... often counter-examples of such results. In order to get striking counter-examples, we provide very simple ones, if not trivial (mainly based on elementary algebraic properties of

Déjà paru dans les Proc. de "Category Theory and Computer Science", LNCS 283, 1987.

reçu le 06/07/1989

natural numbers). Nevertheless, many common errors, or misinterpretations found in the abstract data type litterature result from similar mechanisms. This emphasizes the fact that category theory should be carefully used in the abstract data type field, including for (very) low level concepts.
More provocatively: this paper mainly points out the fact that the synthesis functor $F$ of abstract data types "does not preserve philosophy." However, since about teen years [ADJ 76], it is well known that this functor is crucial for defining a hierarchical, modular approach of abstract specifications!

Some elementary reminders about abstract data types are given in the next section (Section 2). Section 3 discusses about the well known *forgetful* and *synthesis functors*, $U$ and $F$, associated with a hierarchical approach. Section 4 shows the difficulty of properly defining sufficient completeness and hierarchical consistency with loose semantics. In Section 5, we show what happens when combining enrichments. Lastly, Section 6 discusses about a loose semantics obtained by "protecting" predefined sorts.

The following discussions are mainly centered on pairs [positive fact / proof] (respectively: [negative fact / counter-example]).

## 2. Elementary reminders

Let us begin with basic definitions and properties [ADJ 76]:
Given a *signature* $\Sigma$ (i.e. a finite set S of *sorts* and a finite set $\Sigma$ of *operation*-names with arity in S), a $\Sigma$-algebra, $A$, is a heterogeneous set partitioned as $\{ A_s \}_{s \in S}$, and for each operation-name $op: s_1 \cdots s_{n-1} \rightarrow s_n$ of $\Sigma$ there is an operation $op_A: A_{s_1} \times \cdots \times A_{s_{n-1}} \rightarrow A_{s_n}$. A $\Sigma$-*morphism* from $A$ to $B$ is a sort-preserving, operation-preserving application from $A$ to $B$. This defines a category, denoted by Alg($\Sigma$) ; it has an initial object: the ground-term algebra $T_\Sigma$.
In the following, a *specification* SPEC will be defined by a signature $\Sigma$ and a finite set E of *positive conditional equations* of the form:

$$v_1 = w_1 \wedge \cdots \wedge v_{n-1} = w_{n-1} \Rightarrow v_n = w_n$$

where $v_i$ and $w_i$ are $\Sigma$-terms with variables.
Given a specification SPEC, Alg(SPEC) is the full sub-category of Alg($\Sigma$) whose objects are the $\Sigma$-algebras which validate each axiom of E. The category Alg(SPEC) has an initial object, denoted by $T_{\mathrm{SPEC}}$ [BPW 82].
Since $T_{\mathrm{SPEC}}$ exists, Gen(SPEC) can be defined as the full sub-category of Alg(SPEC) such that the initial morphism is an epimorphism (i.e. is *surjective*, in our framework). Gen(SPEC) is the category of the *finitely generated algebras*. Our first "fact" will be devoted to the following remark:
It is well known that Gen(SPEC) is a particularly interesting category for the abstract

data type computer scientist; nevertheless, this is not exactly due to its large spectrum of morphisms, as reminded below.

**Fact 1 :** *Morphisms from a finitely generated algebra*

Let $\Gamma$ be an object of Gen(SPEC) and $A$ an object of Alg(SPEC). The set $Hom_{\mathrm{Alg(SPEC)}}(\Gamma,A)$ contains *at most one element.* Consequently, for all objects $X$ and $Y$ of Gen(SPEC), $Hom_{\mathrm{Gen(SPEC)}}(X,Y)$ contains at most one morphism.

**Proof :**

By initiality properties, if there exists a morphism $\mu$, then the following triangle commutes:

*(For printing facilities, our triangles will become squares!)*

$$\begin{tikzcd} \Gamma \arrow[r, "\mu"] & A \\ T_{\mathrm{SPEC}} \arrow[u, "\text{initial morphism}"] \arrow[r, equal] & T_{\mathrm{SPEC}} \arrow[u, "\text{initial morphism}"'] \end{tikzcd}$$

Thus, the unicity of $\mu$ results from the surjectivity of the initial morphism associated with $\Gamma$. □

One of the most important aspect of abstract data types is its structured, hierarchical, modular approach. This is obtained by means of *presentations.* A presentation PRES over SPEC is a new "part of specification" PRES=<S',Σ',E'> such that the disjoint union SPEC+PRES=<S∪S',Σ∪Σ',E∪E'> is a specification. Sorts and operations of SPEC are often called the *predefined* sorts and operations. Relations between the categories Alg(SPEC) and Alg(SPEC+PRES) are handled by the well known *forgetful functor* and *synthesis functor*:

($U$: Alg(SPEC+PRES)→Alg(SPEC)) and ($F$: Alg(SPEC)→Alg(SPEC+PRES)).

The functor $F$ is a left adjoint for the functor $U$. Consequently, for each SPEC-algebra $A$, there is a particular morphism from $A$ to $U(F(A))$ : the morphism deduced from the adjunction unit (or *adjunction morphism*). This morphism is absolutely crucial for the hierarchical approach: it allows to evaluate the modifications performed on $A$ under the action of PRES.

**Example 0 :**

If $A$ is equal to N over the signature {*0,succ_*} (without axioms) and if PRES adds *pred_* with the axioms [*pred(succ(n))=succ(pred(n))=n*] , then $U(F(\mathrm{N}))$ is isomorphic to Z. The unit of adjunction leads to the natural inclusion; and this morphism permits to show that N has been modified by adding negative values.

If the axioms were [$pred(succ(n))=n$ and $pred(0)=0$], then the unit of adjunction leads to the identity over N showing that this second specification of *pred* does not change N.

## 3. Forgetful and synthesis functors

We first present a rather obvious reminder about the *forgetful functor*. Let $B$ be a SPEC+PRES-algebra. The forgetful functor removes all subsets $B_s$ where $s \in S'$, and all operations of $\Sigma'$ are forgotten (including those with arity in S only), but *it does not remove any value of predefined sort*: $U(B_s)=B_s$ for each $s \in S$. For instance, in Example 0, $U(Z)=Z \neq N$.

Let us remind the classical definition of the *synthesis functor* (although classical, this definition is the starting point of some misinterpretations!): Let $A$ be a SPEC-algebra and let $T_{\Sigma+\Sigma'(A)}$ be the algebra of $\Sigma+\Sigma'$-terms with variables in $A$; we denote by *eval*: $U(T_{\Sigma+\Sigma'(A)}) \rightarrow A$ the canonical evaluation morphism. $F(A)$ is the quotient of $T_{\Sigma+\Sigma'(A)}$ by the smallest congruence containing both the fibers of *eval* and the close instanciations of E+E'.
Because E+E' is required in the definition of $F$ (instead of E' alone), $F(A)$ does not only depend on $A$ and PRES ; it also depends on SPEC.

**Fact 2 :**

Given a presentation PRES, the action of the synthesis functor $F$ over a given, fixed algebra $A$ is highly dependent of the predefined specification.
As outlined in the following example, this fact considerably restricts the possibility of writing "implementation independent" specifications (see for instance [EKMP 80], [SW 82], or [BBC 86a] about abstract implementations).

**Example 1 :**

Let SPEC be a classical specification of *NAT* with operations *0*, *succ_* and *_+_* :

$$\begin{aligned} x + 0 &= x \\ x + succ(y) &= succ(x+y) \end{aligned}$$

Let SPEC' be the specification obtained by adding the following axiom to SPEC :

$$x + y = x + z \quad \Rightarrow \quad y = z$$

The specifications SPEC and SPEC' have clearly the same initial object: N . In fact, they have the same finitely generated algebras because the previous axiom can be proved from SPEC via structural induction.
Let PRES be the presentation adding no sort, adding the operation _×_, and adding the axioms:

$$
\begin{array}{rcll}
x \times succ(0) & = & x & \text{(1 is neutral)} \\
x \times succ(y) & = & x + (x \times y) & \text{(recursive definition)}
\end{array}
$$

When **PRES** is shown as a presentation over **SPEC**, $F$(N) is a model where all terms containing a multiplication by $0$ cannot be evaluated. When **PRES** is shown as a presentation over **SPEC'**, $F$(N) is isomorphic to **N**, because:

$$x + 0 = x = x \times succ(0) = x + (x \times 0)$$

and the simplification axiom of **SPEC'** leads to $0 = x \times 0$ .

Notice that, in spite of the fact that **SPEC** and **SPEC'** have the same finitely generated models and the same initial algebra, the presentation **PRES** is not completely specified over the first specification, but is completely specified over the second one.

## 4. Consistency and completeness

The subject of this section is an examination of some *a priori* possible definitions of the notions of *sufficient completeness* and *hierarchical consistency* with loose semantics. We start with the most loose semantics: the entire category Alg(SPEC). We will show that the simplest definitions are unacceptable for abstract specification purposes.

All the counter-examples provided in this section are based on the following specification+presentation example. Hopefully, we believe that this counter-example cannot be suspected to be too much unusual, complicated or *ad hoc*.

**Example 2 :**

Let **SPEC** be a specification of natural numbers (for instance the specification given in Example 1) together with a sort *BOOL* and boolean operations *True* and *False*. We consider the presentation **PRES** enriching **SPEC** by an equality predicate *eq?* :

$$
\begin{array}{rcl}
eq?(0,0) & = & True \\
eq?(0,succ(n)) & = & False \\
eq?(succ(m),0) & = & False \\
eq?(succ(m),succ(n)) & = & eq?(m,n)
\end{array}
$$

Looking at this presentation **PRES**, we can affirm that a "good notion" of sufficient completeness (resp. hierarchical consistency) should be satisfied by **PRES**. This example is simply written by taking into account each possible value for the arguments of *eq?*, with respect to the constructors of **SPEC**, moreover there are no axioms between constructors (fair presentation [Bid 82]).

We may of course imagine more sophisticated presentation examples, in particular examples which add new sorts to **SPEC**. But our goal is simply to prevent the abstract data type researcher from using a naive, rather unrealistic definition of sufficient completeness or hierarchical consistency.

## 4.1. Sufficient completeness

In the initial approach, sufficient completeness is defined as follows [Gau 78].

*"The adjunction morphism associated with the initial algebra is surjective:"*

$$T_{\mathrm{SPEC}} \rightarrow U(F(T_{\mathrm{SPEC}}))$$

This condition exactly means that PRES does not add new values to $T_{\mathrm{SPEC}}$ . Remind that $F(T_{\mathrm{SPEC}})=T_{\mathrm{SPEC+PRES}}$ , due to adjunction properties.

**Fact 3 :**

The following definition of sufficient completeness is not suitable in the general case:

"PRES *is sufficiently complete if and only if for all algebras in* Alg(SPEC) *the adjunction morphism is surjective*".

Using Example 2, we convince ourselves of this fact by considering the SPEC-algebra obtained by two copies of N. This algebra, (N×{0,1} and {*True,False*}) , is not finitely generated, but is an object of Alg(SPEC) by sending the operation-name *0* over the element *(0,0)* , and *succ((n,a))=(succ(n),a)* . Terms of the form *eq?((n,0),(m,1))* cannot be evaluated using the **PRES** axioms of Example 2. Consequently, they add new boolean values, and the adjunction morphism is not surjective.

**Fact 4 :**

The following two definitions of sufficient completeness are logically equivalent:

1) *the adjunction morphism associated with the initial algebra* $T_{\mathrm{SPEC}}$ *is surjective*

2) *for all algebras in* Gen(SPEC) *the adjunction morphism is surjective.*

**Proof :**

[2⟹1] is trivial because the initial algebra is finitely generated.

[1⟹2] : let A be a finitely generated SPEC-algebra. By construction of *F*, *F*(*A*) is finitely generated over the signature of SPEC+PRES. Consequently, the image of the initial morphism via the forgetful functor is surjective:

$$U(init_A)\colon U(F(T_{\mathrm{SPEC}}))=U(T_{\mathrm{SPEC+PRES}}) \rightarrow U(F(A))$$

Our conclusion results from the commutativity of the following diagram:

$$\begin{tikzcd}[row sep=large, column sep=huge]
A \arrow[r, "\text{adjunction}"] & U(F(A)) \\
T_{\mathrm{SPEC}} \arrow[u, "\textit{surjective}"] \arrow[r, "\text{surjective}"] & U(T_{\mathrm{SPEC+PRES}}) \arrow[u, "\textit{surjective}"']
\end{tikzcd}$$

Restricting ourselves to finitely generated algebras has several disadvantages. For instance, parameterized presentations require a non finitely generated semantics [ADJ 80].

## 4.2. Hierarchical consistency

In the initial approach, hierarchical consistency is defined as follows:

''*the adjunction morphism associated with the initial algebra is a monomorphism*''

(i.e. is *injective* in our framework).

**Fact 5 :**

The following definition of hierarchical consistency is not suitable in the general case:

''**PRES** *is hierarchically consistent if and only if for all algebras in* Alg(SPEC) *the adjunction morphism is injective*''.

Let us return to Example 2. If we consider the SPEC-algebra **Z** (which is a non finitely generated algebra), we get the following inconsistency:

$$True = eq?(0,0) = eq?(0,succ(-1)) = False$$

Restricting hierarchical consistency checks to finitely generated algebras does not yield better results:

**Fact 6 :**

The following definition of hierarchical consistency is not suitable in the general case:

''**PRES** *is hierarchically consistent if and only if for all algebras in* Gen(SPEC) *the adjunction morphism is injective*''.

Using Example 2 again, we consider a finitely generated algebra of the form $\frac{\mathbf{Z}}{n\mathbf{Z}}$ , and we get the following inconsistency:

$$True = eq?(0,0) = eq?(0,n) = eq?(0,succ(n-1)) = False$$

These facts prove that "defining sufficient completeness on Alg(SPEC)" , "defining hierarchical consistency on Alg(SPEC)" or "defining hierarchical consistency on Gen(SPEC)" are too strong requirements. Extension from the purely initial semantics to a loose semantics must be done more carefully.

# 5. Combining presentations

In the remainder of this paper, we simply follow the definitions of sufficient completeness and hierarchical consistency given at the beginning of sections 4.1 and 4.2 (i.e. the initial approach). Given a specification SPEC, we consider two presentations $PRES_1$ and $PRES_2$ *with disjoint signatures*.
Let PRES be the union of $PRES_1$ and $PRES_2$ , we care about the sufficient completeness and hierarchical consistency of PRES. In spite of the strong hypothesis described here, we have sometimes to be careful, as detailed in the following two subsections.

## 5.1. Sufficient completeness

**Fact 7 :**

If $PRES_1$ and $PRES_2$ are both sufficiently complete over SPEC, then $PRES=PRES_1+PRES_2$ remain sufficiently complete. Moreover, under the same hypothesis, $PRES_2$ is sufficiently complete over $SPEC+PRES_1$.

**Proof :** *(using elementary tools)*

$T_{SPEC+PRES_1+PRES_2}$ is the quotient of $T_{\Sigma+\Sigma_1+\Sigma_2}$ by the smallest congruence containing the close instanciations of the $SPEC+PRES_1+PRES_2$ axioms [BPW 82]. Consequently, it suffices to prove that each $\Sigma+\Sigma_1+\Sigma_2$-ground-term of sort in S (resp. in $S+S_1$) belongs to the equivalence class of a $\Sigma$-term (resp. $\Sigma+\Sigma_1$-term). This can be trivially proved via structural induction. □

Obviously, the converse is false: the sufficient completeness of PRES does not imply the sufficient completeness of $PRES_1$ or $PRES_2$ .

## 5.2. Hierarchical consistency

**Fact 8 :**

The hierarchical consistency of $PRES_1$ and $PRES_2$ over SPEC *does not imply* the hierarchical consistency of $PRES=PRES_1+PRES_2$ over SPEC.

**Example 3 :**

Let SPEC be a specification of natural numbers. Let $PRES_1$ be the presentation simply containing the following axiom:

$$succ(n) = 0 \quad \Rightarrow \quad n = 0$$

$PRES_1$ is clearly consistent (in fact, the premise cannot be satisfied in the initial object, thus this axiom is never applied). Let $PRES_2$ be the presentation adding the operation *pred_* with $[pred(succ(n))=succ(pred(n))=n]$ . $PRES_2$ is clearly hierarchically consistent over natural numbers (even though it is not sufficiently complete). The union $PRES=PRES_1+PRES_2$ is not hierarchically consistent because from $succ(pred(0))=0$ we get:

$$0=pred(0)\ , \quad \text{which leads to} \quad succ(0)=succ(pred(0))=0$$

Another example of the same fact is the following:

**Example 4 :**

Let $PRES_1$ be the presentation described in Example 2 (adding equality predicate to natural numbers), and let $PRES_2$ be the same presentation as Example 3 before (adding *pred*). $PRES_1$ and $PRES_2$ are clearly hierarchically consistent over natural numbers, but the union $PRES=PRES_1+PRES_2$ is not hierarchically consistent because:

$$True = eq?(0,0) = eq?(0,succ(pred(0))) = False$$

(a similar example was first presented in [EKP 80], for abstract implementation purposes).

**Fact 9 :**

If $PRES=PRES_1+PRES_2$ is hierarchically consistent over SPEC , then $PRES_1$ and $PRES_2$ are hierarchically consistent over SPEC.

**Proof :**

Assume that $PRES_1$ is not consistent: the morphism from $T_{SPEC}$ to $U(T_{SPEC+PRES_1})$ is not injective. Since the following diagram commutes, the adjunction morphism from $T_{SPEC}$ to $T_{SPEC+PRES_1+PRES_2}$ is not injective:

$$\begin{array}{ccc}
U_1(T_{SPEC+PRES_1}) & \rightarrow & U(T_{SPEC+PRES_1+PRES_2}) \\
\uparrow & & \uparrow \\
PRES_1 & & PRES_1+PRES_2 \\
\textit{adjunction morphism} & & \textit{adjunction morphism} \\
| & & | \\
T_{SPEC} & = & T_{SPEC}
\end{array}$$

SPEC+$PRES_1$).

It results that PRES is not hierarchically consistent over SPEC. □

**Fact 10 :**

If $PRES_1$ and $PRES_2$ are both hierarchically consistent and sufficiently complete over SPEC, then $PRES=PRES_1+PRES_2$ too. Moreover, under the same hypothesis, $PRES_2$ is hierarchically consistent and sufficiently complete over $SPEC+PRES_1$.

(This fact is well known; a demonstration with conditional axioms, including exception handling, can be found in [Ber 86]).

## 6. Loose semantics with "Protect"

Clearly, abstract specifications do not necessarily directly lead to executable specifications. It is often convenient to specify some operations via "universal properties." For instance the subtraction can be specified via:

$$z - y = x \quad \Leftrightarrow \quad x + y = z$$

Sometimes, such axioms may lead to uncompletely specified presentations, as in the following example.

**Example 5 :**

Let SPEC be an initial specification of integers with operations *0*, *succ_*, *pred_*, _+_, _–_ and _×_ . Let us specify a presentation PRES adding the operation *_div_* as follows:

$$\begin{aligned} 0 \leq (a - (b \times (a\ div\ b))) &= True \\ (a - (b \times (a\ div\ b))) < b &= True \end{aligned}$$

These axioms characterize *(a div b)* among all integers *finitely generated* with respect to *succ* and *pred*. However, in the initial model $T_{SPEC+PRES}$ , the term *(a div b)* is not reached by *succ* and *pred* . Its value is only a unreachable value such that the (unreachable) remainder $(a - (b \times (a\ div\ b)))$ returns the specified boolean values when compared with *0* and *b* .

Consequently, this presentation is uncompletely specified according to the usual definition of sufficient completeness.

In such examples, the only interesting models are those which do not modify the predefined initial model (Z). This leads to a (loose) semantics where models are those *protecting* predefined sorts [Kam 80]. Indeed, when writing relatively large specifications, this semantics seems to be highly suitable (ASL [Wir 82] [SW 83], PLUSS [Gau 84], OBJ [FGJM 85], LARCH [GH 83] ... ).

Let us define the associated category:

**Definition :** *The "Protect" category*

Let SPEC be a specification and let PRES be a presentation over SPEC. The category of PRES-*models protecting* SPEC is the full subcategory of Alg(SPEC+PRES) whose objects are the SPEC+PRES-algebras $A$ such that $U(A)$ is isomorphic to the initial predefined algebra $T_{SPEC}$ . We denote this category by Pro(SPEC,PRES) .

Notice that the object class of Pro(SPEC,PRES) can be empty.

**Fact 11 :**

If Pro(SPEC,PRES) is not an empty category, then PRES is hierarchically consistent over SPEC.

(Here, consistency is defined with respect to the initial algebra $T_{SPEC}$ only)

**Proof :**

If $T_{SPEC+PRES}$ is inconsistent over $T_{SPEC}$, then a fortiori all SPEC+PRES-algebras are inconsistent over $T_{SPEC}$ (because $T_{SPEC+PRES}$ is minimal). □

**Fact 12 :**

Even if PRES is consistent over SPEC, Pro(SPEC,PRES) may be empty.

**Example 6 :**

Let SPEC be the boolean specification with *True* and *False*. Let PRES be a specification of *SET(BOOL)* with $\varnothing$, *insert*, $\in$ and *choose* :

$$\begin{aligned} True \in \varnothing &= False \\ False \in \varnothing &= False \\ b \in insert(b',X) &= b{=}b' \ or \ b{\in}X \\ choose(X) \in X &= True \end{aligned}$$

This specification is clearly hierarchically consistent (even though it is not sufficiently complete). However, the Protect category is empty, because the term *choose*($\varnothing$) can neither be equal to *True* nor to *False* (both choices induce *True=False*).

(Fortunately, this example can be easily specified without inconsistency using abstract data types with exception handling [Bid 84] [GDLE 84] [BBC 86b] [Ber 86], or with partial functions [BW 82].)

**Fact 13 :**

Even if Pro(SPEC,PRES) contains models, it has not necessarily an initial object.

**Example 7 :**

Let SPEC be the boolean specification with *True* and *False*. Let PRES be the presentation adding the constant operation *maybe*, without any axiom. Pro(SPEC,PRES) contains two models, no one is initial.

**Fact 14 :**

If PRES is sufficiently complete over SPEC, then either Pro(SPEC,PRES) has an initial object, either it is empty.

**Proof :**

If PRES is consistent, then the initial model $T_{\mathrm{SPEC+PRES}}$ belongs to Pro(SPEC,PRES); it is then necessarily initial in Pro(SPEC,PRES). If PRES is not hierarchically consistent, then Fact 11 implies that Pro(SPEC,PRES) has no object. □

**Fact 15 :**

There are presentations PRES which are not sufficiently complete over SPEC, such that Pro(SPEC,PRES) is not empty and has an initial object.

It suffices to refer to Example 5, where the axioms characterize *div* by a "universal property among integers." The division is incompletely specified according to classical initial definition of sufficient completeness, but Pro(SPEC,PRES) only contains one model (Z) which is necessarily initial.

## 7. Conclusion

We have investigated how a hierarchical approach of abstract data types, with the notions of *hierarchical consistency* and *sufficient completeness*, could be defined when dealing with so-called *loose semantics*. The results shown in sections 2 to 5 seem to be somewhat pessimistic:

- The synthesis functor is "implementation dependent" with respect to the predefined specification (Fact 2).
- Sufficient completeness cannot be checked on all models (Fact 3).
- Hierarchical consistency cannot be checked on all models (Fact 5).
- Hierarchical consistency cannot be checked on all finitely generated models, a smaller class of models must be investigated (Fact 6).
- Combining hierarchically consistent presentations does not result on a hierarchically consistent presentation (Fact 8).

However, we showed some positive results:

- Checking sufficient completeness on all finitely generated algebras is equivalent to check it on the initial algebra only (Fact 4).
- Combining sufficiently complete presentations results on sufficiently complete presentations (Fact 7); the same occurs for presentations that are both sufficiently complete and hierarchically consistent (Fact 10).

In the last section (Section 6), we defined the category of models *protecting predefined sorts*. We have investigated the relations between the classical notions of completeness/consistency and the elementary properties of this category:

- The category is empty if the presentation is not hierarchically consistent, but the converse is false (Facts 11 & 12).
- The category has not necessarily initial models (Fact 13).
  It has initial models if the presentation is sufficiently complete and hierarchically consistent, but the converse is false (Facts 14 & 15).

In conclusion: From facts 2, 3, 5, 6 and 8, we showed that the synthesis functor of classical abstract data types "does not always preserve philosophy" when dealing with loose semantics. Moreover, with a loose semantics based on protection of predefined sorts, the corresponding category has few systematic relations with sufficient completeness or hierarchical consistency (facts 11 to 15).

Acknowledgements :

It is a pleasure to express gratitude to Michel Bidoit, Christine Choppy and Marie-Claude Gaudel for encouragements to write this paper and careful proof readings. The title was suggested by Stephane Kaplan.

## 8. References


[ADJ 76] **Goguen J., Thatcher J., Wagner E.** : "*An initial algebra approach to the specification, correctness, and implementation of abstract data types*", Current Trends in Programming Methodology, Vol.4, Yeh Ed. Prentice Hall, 1978. Also : IBM Report RC 6487, Oct. 1976.

[ADJ 80] **Ehrig H., Kreowski H., Thatcher J., Wagner J., Wright J.** : "*Parameterized data types in algebraic specification langages*", Proc. 7th ICALP, July 1980.

[BBC 86.a] **Bernot G., Bidoit M., Choppy C.** : "*Abstract implementations and correctness proofs*", Proc. 3rd STACS, January 1986, Springer-Verlag LNCS 210, January 1986. Also : LRI Report 250, Orsay, Dec. 1985.

[BBC 86b] **G. Bernot, M. Bidoit, C. Choppy** : "*Abstract data types with exception handling : an initial approach based on a distinction between exceptions and errors*", Theoretical Computer Science, Vol. 46, n° 1, p. 13-45, November 1986.

[Ber 86] **Bernot G.** : "*Une sémantique algébrique pour une spécification différenciée des exceptions et des erreurs : application à l'implémentation et aux primitives de structuration des spécifications formelles*", Thèse de troisième cycle, LRI, Université de Paris-Sud, Orsay, Février 1986.

[Bid 82] **Bidoit M.** : "*Algebraic data types: structured specifications and fair presentations*", Proc. AFCET Symposium on Mathematics for Computer Science, Paris, March 1982.

[Bid 84] **Bidoit M.** : "*Algebraic specification of exception handling by means of declarations and equations*", Proc. 11th ICALP, Springer-Verlag LNCS 172, July 1984.

[BPW 82] **Broy M., Pair C., Wirsing M.** : "*A systematic study of models of abstract data types*", Theoretical Computer Sciences, p. 139-174, vol. 33, October 1984.

[BW 82] **Broy M., Wirsing M.** : "*Partial abstract data types*", Acta Informatica, Vol.18-1, Nov. 1982.

[EKMP 80] **Ehrig H., Kreowski H., Mahr B., Padawitz P.** : "*Algebraic implementation of abstract data types*", Theoretical Computer Science, Oct. 1980.

[EKP 80] **Ehrig H., Kreowski H., Padawitz P.** : "*Algebraic implementation of abstract data types: concept, syntax, semantics and correctness*", Proc. ICALP, Springer-Verlag LNCS 85, 1980.

[FGJM 85] **Futatsugi K., Goguen J., Jouannaud J-P., Meseguer J.** : "*Principles of OBJ2*", Proc. 12th ACM Symp. on Principle of Programming Languages, New Orleans, January 1985.

[Gau 78] **Gaudel M-C.** : "*Spécifications incomplètes mais suffisantes de la représentation des types abstraits*", Laboria Report 320, 1978.

[Gau 84] **Gaudel M-C.** : "*A first introduction to PLUSS*", LRI Report, Orsay, December 1984.

[GDLE 84] **Gogolla M., Drosten K., Lipeck U., Ehrich H.D.** : "*Algebraic and operational semantics of specifications allowing exceptions and errors*", Theoretical Computer Science 34, North Holland, 1984.

[GH 83] **Guttag J.V., Horning J.J.** : "*An introduction to the LARCH shared language*", Proc. IFIP 83, REA Mason ed., North Holland Publishing Company, 1983.

[Kam 80] **Kamin S.** : "*Final data type specifications : a new data type specification method*", Proc. of the 7th POPL Conference, 1980.

[SW 82] **Sannella D., Wirsing M.** : "*Implementation of parameterized specifications*", Report CSR-103-82, Department of Computer Science, University of Edinburgh.

[SW 83] **Sannella D., Wirsing M.** : "*A kernel language for algebraic specification and implementation*", Proc. Intl. Conf. on Foundations of computation Theory, Springer-Verlag, LNCS 158, 1983.

[Wir 82] **Wirsing M.** : "*Structured algebraic specifications*", Proc. of AFCET Symposium on Mathematics for Computer Science, Paris, March 1982.