ALGEBRAIC SEMANTICS OF EXCEPTION HANDLING

Gilles BERNOT*, Michel BIDOIT*, Christine CHOPPY*

ABSTRACT

In this paper, a new semantics for exception handling in algebraic specifications is provided. Our formalism allows all forms of exception and error handling (several error messages, implicit error propagation rule, exception recovery policy), while preserving the existence of *initial models*. It handles complex examples where various exceptional cases (leading to different processings) can be specified. The main concept of our approach is the distinction between *exception* and *error*. This formalism allows use of *congruences* in a similar manner as in the classical abstract data type theory. Moreover, we show how a functorial semantics of *enrichment* can be carried over to our framework, and we show how *hierarchical consistency* and *sufficient completeness* can be redefined. These results provide a firm basis for writing modular, structured specifications with exception handling features.

* **Laboratoire de Recherche en Informatique**
Bât 490, Université PARIS-SUD
F-91405 ORSAY CEDEX
FRANCE

1. INTRODUCTION

Since the pioneer work of [ADJ 76], specifying abstract data types with exception handling has turned out to be an especially difficult problem. Various solutions have been proposed, that range from the *algorithmic approach* [Loe 81, EPE 81] to the *partial approach* [BW 82]. But most approaches may be more or less related to the *error-algebra approach*, i.e. the algebra carrier sets are split into okay and erroneous values [ADJ 76, Gog 77, Gog 78, Pla 82, GDLE 84, Bid 84].

[Bid 84] shows that none of these approaches is completely satisfactory, and presents a new approach allowing all forms of error handling (error declaration, error propagation and error recovery). Unfortunately, even if the approach described in [Bid 84] seems to be promising, it does not solve the whole problem, since the existence of initial models is not guaranteed.
Although recent developments in algebraic specification languages [Wir 82, SW 83, Wir 83] propose an elegant algebraic semantics without requiring the existence of initials models, our claim is that initiality is a major tool to express the semantics of most specification-building primitives, at least if one wants to state the results in a categorical framework. Such an approach has been used in [ADJ 79, ADJ 80], [Bid 82] and [EKMP 80] to describe the semantics of *enrichment*, *parameterization* and *abstract implementation*.

An interesting formalism is described in [GDLE 84] that always provides initial objects : the signature of a specification is divided into *safe* operations that cannot add erroneous values (such as *succ* or + in natural numbers) and *unsafe* operations (such as *pred* or –). Unfortunately, all operations are unsafe in most cases (e.g. *succ* for bounded natural numbers) and therefore the ok-part of the initial algebra is reduced to safe constants (e.g. 0).

In this paper, we propose a new semantics for exception handling in algebraic specifications. Our formalism allows all forms of exception handling, including specification of several error messages, implicit error propagation and error recovery, while preserving the existence of initial objects at the semantical level. Moreover, the concepts of *enrichment*, *parameterization* and *abstract implementation* can easily be extended to our exception handling framework, as our semantics is entirely functorial.
In the next section, we explain the key ideas of our approach. In sections 3 through 7, we describe our formalism. The existence of an initial object is proved in Section 8 ; and Section 9 defines enrichment with exception handling. We assume that the reader is familiar with elementary results of category theory and the standard (ADJ) approach to abstract data types.

2. THE KEY IDEAS OF OUR APPROACH

Several criteria are very useful in defining a true exception handling policy :

- In order to avoid a large number of exception declarations, it is of first interest to have *implicit exception and error propagation rules*. We will show that implicit exception propagation and implicit error propagation are semantically encoded in our *exception-algebras*.
- In most realistic examples, it is necessary to be able to *recover* various exceptions. Thus, realistic exception handling formalisms must provide error recovery features. In our formalism, error recoveries are specified by means of *generalized axioms*. In particular, we can specify *non strict operations*.
- Moreover, we have the possibility to specify several *error messages*, by means of *exception labelling*. For instance, distinct error messages are associated with *pred(0)* and (*x div 0*) ; and there are different recoveries as well. This feature is not provided for in any of the previous works, but is essential for a realistic exception recovery policy.

In addition to these ideas, our formalism is based on two main concepts : the *okay standard forms* and the distinction between *exceptional cases* and *erroneous values*. These two concepts can be handled due to the fact that for each exception-algebra, A, the semantics is handled using the free algebra over A : $T_{\Sigma(A)}$. In the following subsections, we explain the reasons why these two features are needed in exception handling ; and we sketch out the reasons why the use of $T_{\Sigma(A)}$ is crucial for our purposes.

2.1. Exceptions and errors

As in most formalisms already put forward, we make use of *Ok-axioms* which describe the okay cases, and we also use another set of axioms for the erroneous and recovery cases. But recovery cases and okay axioms often lead to inconsistencies. For instance, let us specify the bounded natural numbers with the operations *0*, *succ* and *pred*. Let *Maxint* be the upper bound. We have the Ok-equation : *pred(succ(n)) = n* . Assume that we want to recover all values greater than *Maxint* with the recovery axiom : *succ(Maxint) = Maxint* . The term *pred(succ(Maxint))* is then equal to *Maxint*, but it is

also equal to *pred(Maxint)* ; which results to inconsistencies.

In fact, it is necessary to distinguish between the *term succ(Maxint)* that is *exceptional* (thus, Ok-equations should not be applied), and its *class* which is an *okay value* (*Maxint*).
This problem may be solved in the ground term algebra (and in the finitely generated algebras) by using the Ok-axioms *before* the recovery axioms are applied (as *succ(Maxint)* has not been recovered, it is not yet okay). But finitely generated algebras are not powerful enough to cope with enrichment, parameterization or abstract implementation.
In our approach, this difficulty is avoided as follows : for each *exception-algebra*, A, we work in the free algebra of **Σ**-terms with variables in *A*, instead of working directly in *A*. We denote by $T_{\Sigma(A)}$ this **Σ**-algebra. Constructions that can usually be done at the ground term level can also be done at the $T_{\Sigma(A)}$ level, since we can consider the elements of *A* as additional constants. Then the morphism *eval* (which *evaluates* the terms of $T_{\Sigma(A)}$ into *A*) carries the constructions made at the $T_{\Sigma(A)}$ level over to *A*.

2.2. The okay standard forms

Since Ok-axioms only concern okay terms, it is necessary to characterize these terms. But it is not possible to characterize *all* the okay terms (*succ(0), 0+1, pred(succ(succ(0)))* ...). We can only characterize some reference terms. These reference terms may be chosen in different manners. In most examples, normal forms guide the appropriate choice. Unfortunately normal forms are not always unique (e.g. integers : $pred^n(0)$ or $Op(succ^n(0))$). Therefore this choice must be declared in the specification.
Since our axioms are not always equivalent to canonical term rewriting systems, we call our reference terms *standard forms*. It is not necessary for these standard forms to be canonical ones, even if this is true in most examples. In our framework, the *standard forms* are characterized by means of recursive declarations. For instance, we declare the standard forms of integers in the following manner :

0 and succ(0) are standard forms
if succ(z) is a standard form then succ(succ(z)) is a standard form
if succ(z) is a standard form then Op(succ(z)) is a standard form

another possibility would be

0, succ(0) and pred(0) are standard forms
if succ(z) is a standard form then succ(succ(z)) is a standard form
if pred(z) is a standard form then pred(pred(z)) is a standard form

We have now standard forms, but there are still some exceptional standard forms such as $succ^{Maxint+1}0$. The *okay* standard forms are the non exceptional ones. Thus, we declare in a similar way the exceptional standard forms (called the *standard exceptions*). The *okay standard forms* are obtained from the standard forms by removing the standard exceptions (Section 4, Definition 8).
This construction can be done in the ground term algebra. We generalize it for the non finitely generated algebras by working in the free algebra over A, $T_{\Sigma(A)}$.

The following sections describe our formalism. An *exception specification* will be defined by :

SPEC = < **S** , **Σ** , **L** , **St-Frm** , **St-Exc** , **Ok-Ax** , **Lbl-Ax** , **Gen-Ax** >

where <**S**,**Σ**,**L**> is an *exception signature*, **St-Frm** is a *standard form declaration*, **St-Exc** is a *standard exception declaration*, **Ok-Ax** is a set of *okay axioms*, **Lbl-Ax** is a set of *labelling axioms* and **Gen-Ax** is a set of *generalized axioms*. All these parts are successively defined in sections 3 through 7.
Such a syntax may seem complicated, but this complexity reflects the complexity of the various examples that we can modelize, and is not inherent to our formalism. For instance, if we are not interested with error messages (as in all formalisms already put forward), we specify **L**=∅ and **Lbl-Ax**=∅ ; if we are not interested with "bounded" data structures (as in [GDLE 84]), we specify **St-Exc**=∅ ; and if we are not interested with recovery features (as in all the partial algebra approaches), we specify **Gen-Ax**=∅. Then, we obtain a syntax which looks like a classical one (ADJ) together with *standard forms* (**St-Frm**) ; however, all our properties remain true (existence of initial object and functorial aspect of our semantics). Thus, our approach generalizes all the above mentioned ones.
In the same way, an exception specification where **L**, **St-Frm**, **St-Exc**, **Ok-ax** and **Lbl-Ax** are empty is equivalent to a classical (ADJ) specification, because the semantics of **Gen-Ax** is exactly the same as the usual semantics of classical (positive conditional) axioms.

3. EXCEPTION SIGNATURE

Definition 1 : An *exception signature* is a classical signature together with a set of *exception labels* : $\mathbf{\Sigma}$-*Exc* = <**S**,$\mathbf{\Sigma}$,**L**> where **S** is a finite set of *sorts* ; $\mathbf{\Sigma}$ is a finite set of *operations* with arity in **S** ; and **L** is a finite set of *exception labels*.

Intuitively, these exception labels correspond to the "error messages" of the data type. For instance, the exception signature of bounded natural numbers will contain the set **L** = {*NEGATIVE,TOO-LARGE*} .

Definition 2 : An *exception algebra* over the exception signature $\mathbf{\Sigma}$-*Exc*, is a classical (heterogeneous) $\mathbf{\Sigma}$-algebra, *A*, together with a family of subsets, $\{A_l\}$, indexed by $\mathbf{L}\cup\{Ok\}$: A=($A,\{A_l\}$) .
The subset A_{Ok} is the set of all *okay values* of A. The subsets A_l are *not* necessarily disjointed, and they can intersect several sorts.

Example 1 : Let $\mathbf{\Sigma}$-*Exc* be an exception signature of bounded natural numbers, *NAT*. The algebra **N**, with *pred(0)=0* and with $\mathbf{N}_{Ok}$=[*0,Maxint*] $\mathbf{N}_{NEGATIVE}$={0} and $\mathbf{N}_{TOO\text{-}LARGE}$=]Maxint,+∞[, is an example of $\mathbf{\Sigma}$-*Exc*-algebra. The intuitive meaning of this algebra is that every value greater than *Maxint* is erroneous with the *TOO-LARGE* exception label, while the negative values are recovered into the constant 0.

Definition 3 : Let A and B be two $\mathbf{\Sigma}$-*Exc*-algebras. An *exception morphism* from A to B is a classical $\mathbf{\Sigma}$-morphism, μ, that preserves the labeled subsets. This means that for all labels $l \in \mathbf{L}\cup\{Ok\}$, $\mu(A_l)$ is included in B_l.

Our first result is quite easy : the category of $\mathbf{\Sigma}$-*Exc*-algebras has final and initial objects : the trivial algebra, **S**, together with $\mathbf{S}_l$ equal to **S** for all labels, is a final algebra ; the ground term algebra, $T_{\mathbf{\Sigma}}$, together with $T_{\mathbf{\Sigma},l}$ equal to ∅ for all labels, is initial. We denote this initial algebra by $T_{\mathbf{\Sigma}\text{-}Exc}$. There is no labeled ground term and no okay ground term in $T_{\mathbf{\Sigma}\text{-}Exc}$, as nothing is specified about labels in the signature.

4. CHARACTERIZATION OF OKAY STANDARD FORMS

Definition 4 : A *standard form declaration* over $\mathbf{\Sigma}$-*Exc*, denoted by **St-Frm**, is a finite set of elementary declarations as follows :

$$[t_1 \in \text{St-Frm} \wedge \cdots \wedge t_n \in \text{St-Frm} \wedge v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Rightarrow \quad t \in \text{St-Frm}$$

where t_i, v_j, w_j and *t* are $\mathbf{\Sigma}$-terms with variables [*].

Each variable occurring in *t* must also occur in one (at least) of the t_i. (*n* or *m* may be equal to 0).

Example 2 : A standard form declaration over *NAT* can be specified by means of two elementary declarations :

$$0 \in \text{St-Frm}$$
$$n \in \text{St-Frm} \quad \Rightarrow \quad succ(n) \in \text{St-Frm}$$

As outlined above, the semantics of **St-Frm** is provided in $T_{\Sigma(A)}$, in order to cope with the non-finitely generated algebras.

Definition 5 : The set of standard forms of $T_{\Sigma(A)}$, denoted by $St\text{-}Frm_{\mathrm{A}}$, is the smallest subset of $T_{\Sigma(A)}$ such that :

- □ $St\text{-}Frm_{\mathrm{A}}$ contains A_{Ok} (since A_{Ok} is a subset of *A*, its elements are constants of $T_{\Sigma(A)}$).
- □ For each elementary declaration of **St-Frm** of the form :

  $$[t_1 \in \text{St-Frm} \wedge \cdots \wedge t_n \in \text{St-Frm} \wedge v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Rightarrow \quad t \in \text{St-Frm}$$

  and for each substitution, σ, with range in $T_{\Sigma(A)}$, the following holds :
  if $\sigma(t_i) \in St\text{-}Frm_{\mathrm{A}}$ for all *i=1..n* , and $eval[\sigma(v_j)] = eval[\sigma(w_j)]$ for all *j=1..m* , then $\sigma(t)$ belongs to $St\text{-}Frm_{\mathrm{A}}$.

The second condition defines exactly the recursive characterization of the standard forms. The first condition means that, since okay standard forms are in particular standard forms, all Ok-values of A must be (constant) standard forms in $T_{\Sigma(A)}$. For instance, if A=**N** is the algebra of integers, the term *succ(succ(succ(0)))* is a standard form in $T_{\mathbf{\Sigma}(\mathbf{N})}$; but we would also like the terms *succ(2)* or 3 to be standard forms : this is obtained from the first condition. Notice that the existence of $St\text{-}Frm_{\mathrm{A}}$ is clear.

---

[*] for each *j*, v_j and w_j must belong to the same sort, of course.

Our next goal is to remove the exceptional standard forms (e.g. *succ(Maxint)*).

Definition 6 : A *standard exception declaration* over **Σ**-*Exc*, denoted by **St-Exc**, is a finite set of elementary declarations as follows :

$$[t_1 \in \text{St-Exc} \wedge \cdots \wedge t_n \in \text{St-Exc}] \quad \Rightarrow \quad t \in \text{St-Exc}$$

where t_i and t are **Σ**-terms with variables. Each variable occurring in one of the t_i must also occur in t.

Example 3 : In the bounded natural numbers, our standard exception declaration is reduced to one elementary declaration :

$$succ^{Maxint+1}(0) \in \text{St-Exc}$$

It is not necessary to declare exceptional forms greater than *(Maxint+1)*. These exceptions are automatically handled by *implicit exception propagation* encoded in the semantics.

The semantics of standard exceptions is provided in $T_{\Sigma(A)}$. The set of standard exceptions is defined via three main properties : the recursive characterization associated with **St-Exc**, the implicit exception propagation rule, and the stability under *partial evaluations*.

Definition 7 : The set of standard exceptions of $T_{\Sigma(A)}$, denoted by $St\text{-}Exc_A$, is the smallest subset of $T_{\Sigma(A)}$ such that :

- □ for each elementary declaration of **St-Exc** of the form :

  $$[t_1 \in \text{St-Exc} \wedge \cdots \wedge t_n \in \text{St-Exc}] \quad \Rightarrow \quad t \in \text{St-Exc}$$

  and for each substitution, σ, with range in $T_{\Sigma(A)}$, if $\sigma(t_i)$ belongs to $St\text{-}Exc_A$ for all $i=1..n$, then $\sigma(t)$ belongs to $St\text{-}Exc_A$
- □ if u is a term of $T_{\Sigma(A)}$ that belongs to $St\text{-}Exc_A$, then every term t of $T_{\Sigma(A)}$ that contains u as a subterm belongs to $St\text{-}Exc_A$
- □ for each term t in $St\text{-}Exc_A$, and for each strict subterm u of t, the term t' , obtained by substituting the constant *eval(u)* for u in t, is still an element of $St\text{-}Exc_A$.

We are then able to define the okay standard forms of $T_{\Sigma(A)}$, and the validation of **St-Frm** and **St-Exc** for A.

Definition 8 : The set of *okay standard forms* of $T_{\Sigma(A)}$, $Ok\text{-}Frm_A$, is defined by :

$$Ok\text{-}Frm_A = St\text{-}Frm_A - St\text{-}Exc_A .$$

Moreover, the exception algebra A *validates* **St-Frm** and **St-Exc** iff $eval(Ok\text{-}Frm_A) \subset A_{Ok}$. This means that each okay standard form of $T_{\Sigma(A)}$ must have an okay value in A, after the evaluation is done. [*]

Example 4 : The okay standard forms of $T_{\Sigma(\mathbf{N})}$ are the terms of the form $succ^n(m)$ where $0 \le n+m \le Maxint$. The evaluation of such terms is equal to the *value (n+m)*. Thus, the evaluation of each okay standard form of $T_{\Sigma(\mathbf{N})}$ belongs to $\mathbf{N}_{Ok}=[0,Maxint]$. Consequently, **N** (Example 1) validates **St-Frm** and **St-Exc**.

5. OKAY AXIOMS

Our *okay axioms* are *positive conditional axioms* ; we denote a finite set of okay axioms by **Ok-Ax**. The associated semantics is described by means of (classical) *congruences*. But since the distinction between *exceptional cases* and *erroneous values* cannot be made in A, we define the congruence associated to **Ok-Ax** in $T_{\Sigma(A)}$; and then, we define the validation of **Ok-Ax** via *eval*.

Proposition 1 : Given **Σ**-*Exc*, **St-Frm**, **St-Exc**, and **Ok-Ax**, consider a **Σ**-*Exc*-algebra A . There is a least congruence over $T_{\Sigma(A)}$, denoted by $\equiv_{Ok}$, satisfying the following "IF..THEN" condition :
For each substitution, σ, with range in $T_{\Sigma(A)}$, and for each axiom of **Ok-Ax**,

$$[v_1 = w_1 \wedge \cdots \wedge v_n = w_n] \quad \Rightarrow \quad v = w \quad [**],$$

(let $\sigma(v) = op(t_1, \ldots, t_m)$), **IF** the three following conditions holds :

- □ $eval[\sigma(v_i)] = eval[\sigma(w_i)]$ for all $i=1..n$
- □ there are okay standard forms $\alpha_1, \ldots, \alpha_m$ ($\in Ok\text{-}Frm_A$) such that $t_j \equiv_{Ok} \alpha_j$ and $op(\alpha_1, \ldots, \alpha_m) \notin St\text{-}Exc_A$

---

[*] Notice that the reverse inclusion is always satisfied.

[**] or $w = v$, our axioms are not oriented.

□ there is an okay standard form α ($\in Ok\text{-}Frm_A$) such that $\sigma(w) \equiv_{Ok} \alpha$.

THEN $\sigma(v) \equiv_{Ok} \sigma(w)$.

Proof : given in appendix.

The three premisses included in the IF statement are explained as follows :

□ the first condition is the validation of the premisses of our okay axiom

□ the second condition reflects an *innermost evaluation* : to apply the okay axiom, every subterm of $\sigma(v)$ must already have an okay standard form, and the resulting term $op(\alpha_1, \ldots, \alpha_m)$ must not be exceptional

□ the third condition limits the effect of the okay axiom to the okay terms only.

Thus, $\equiv_{Ok}$ has two purposes : it deduces the *okay terms* of $T_{\Sigma(A)}$ from the reference terms of $Ok\text{-}Frm_A$, and it creates the "okay equivalence class" corresponding to each okay standard form. Okay axioms only handle the normal cases of the data type.

Example 5 : Okay axioms of bounded natural numbers are specified as usual :

$$\begin{aligned} pred(succ(n)) &= n \\ n + 0 &= n \\ n + succ(m) &= succ(n) + m \\ n - 0 &= n \\ n - succ(m) &= pred(n) - m \end{aligned}$$

Assume that we want to evaluate the term $succ(pred(succ^{Maxint}(0)))$. We first must evaluate the subterm $pred(succ^{Maxint}(0))$. Its okay standard form is $succ^{Maxint-1}(0)$ (first axiom), and thus we can apply the first okay axiom over $succ(pred(succ^{Maxint}(0)))$, which gives $succ^{Maxint}(0)$, and since $succ^{Maxint}(0)$ is also an okay standard form, we have $succ(pred(succ^{Maxint}(0))) \equiv_{Ok} succ^{Maxint}(0)$.

On the other hand, assume that we want to evaluate the term $pred(succ(Maxint))$. We must first evaluate the subterm $succ(Maxint)$. But our okay axioms cannot associate any okay standard form to $succ(Maxint)$ (this term is exceptional). Thus, the first axiom cannot be applied, and the class of $pred(succ(Maxint))$ via $\equiv_{Ok}$ is reduced to $\{pred(succ(Maxint))\}$. Nevertheless, generalized axioms (Section 7) may handle the evaluation of such terms.

Definition 9 : An exception algebra, A, validates **Ok-Ax** iff the morphism *eval* is compatible with $\equiv_{Ok}$. This means :

$$\neg\forall t \in T_{\Sigma(A)}\ \neg\forall t' \in T_{\Sigma(A)}\ ,\quad [t \equiv_{Ok} t' \ \Rightarrow\ eval(t) = eval(t')]$$

6. GENERALIZED LABELLING

Definition 10 : We denote by **Lbl-Ax**, a finite set of labelling axioms over **Σ**-*Exc* as follows :

$$[t_1 \in l_1 \wedge \cdots \wedge t_n \in l_n \ \wedge\ v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Rightarrow \quad t \in l$$

where t_i, v_j, w_j and t are **Σ**-terms with variables, l_i are members of $\mathbf{L} \cup \{Ok\}$, and l is a member of **L** [*].

Example 6 : Labelling axioms of bounded natural numbers can be specified as follows :

$$\begin{aligned} succ^{Maxint+1}0 &\in TOO\text{-}LARGE \\ n \in TOO\text{-}LARGE &\Rightarrow succ(n) \in TOO\text{-}LARGE \\ n \in TOO\text{-}LARGE &\Rightarrow n + 0 \in TOO\text{-}LARGE \\ (succ(n) + m) \in TOO\text{-}LARGE &\Rightarrow (n + succ(m)) \in TOO\text{-}LARGE \\ pred(0) &\in NEGATIVE \\ n \in NEGATIVE &\Rightarrow pred(n) \in NEGATIVE \\ (n - succ(n)) &\in NEGATIVE \\ (n-m) \in NEGATIVE &\Rightarrow (n - succ(m)) \in NEGATIVE \end{aligned}$$

Equations in the premisses are useful ; for instance, given the operation _ <_ , the following labelling axiom is specified :

$$n<m = True \quad \Rightarrow \quad (n-m) \in NEGATIVE$$

instead of the two last axioms.

---

[*] The l_i's are *not* necessarily distinct.

Notice that even if *exceptions* propagate, *labels* must not (implicitly) propagate. For instance, *pred(0)* is exceptional and *NEGATIVE*, thus the term *succ(pred(0))* is also exceptional (implicit propagation of standard exceptions), but is not a *NEGATIVE* value.

The semantics of **Lbl-Ax** works directly on the *values* of A, in a straightforward manner.

Definition 11 : An exception algebra A=($A, \{A_l\}$) validates **Lbl-Ax** iff for each axiom of **Lbl-Ax**

$$[t_1 \in l_1 \wedge \cdots \wedge t_n \in l_n \ \wedge \ v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Longrightarrow \quad t \in l$$

and for each substitution, σ, with range in A, the following holds :
if $\sigma(t_i)$ belongs to A_{l_i} for all i, and $\sigma(v_j) = \sigma(w_j)$ for all j, then $\sigma(t)$ belongs to A_l.

Although the specifier is free to include whatever axioms (s)he wants in **Lbl-Ax**, it should be noted that labelling axioms have been designed in order to formalize *preconditions* (introduced by Guttag in [Gut 79]).

Remark 1 : **Lbl-Ax** does not create exceptions. The subsets A_l are not necessarily disjointed from A_{Ok}. For instance, even if **Lbl-Ax** contains an axiom of the form "*0∈ANY-LABEL*", 0 is still an okay standard form (and thus an okay value). In other words, okay values labeled by **Lbl-Ax** are automatically recovered. More precisely, *erroneous values* are defined as follows :

Definition 12 : We denote by A_{err} the smallest subset of A=($A, \{A_l\}$) such that :

- □ A_{err} contains $[A_l - A_{Ok}]$ for all labels $l \in \mathbf{L}$.
- □ for each operation $op \in \mathbf{\Sigma}$ and for all values $v_1 \cdots v_n$ (according to the arity of *op*), if (at least) one of the v_i belongs to A_{err} and if $op(v_1, \ldots, v_n)$ is not a member of A_{Ok}, then $op(v_1, \ldots, v_n)$ belongs to A_{err}.

The intuitive meaning of this definition is the following : the first condition states that exception labels generate errors except if they are recovered ; the second condition means that errors propagate except if they are recovered. The second condition is called the *implicit error propagation rule*.

Notice that "*err*" is not a label. It is not compatible with exception morphisms ($\mu(A_{err})$ is not always included in B_{err}).

7. GENERALIZED AXIOMS

Definition 13 : We denote by **Gen-Ax** a finite set of *generalized axioms* as follows :

$$[t_1 \in l_1 \wedge \cdots \wedge t_n \in l_n \ \wedge \ v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Longrightarrow \quad v_{m+1} = w_{m+1}$$

where t_i, v_j and w_j are **Σ**-terms with variables, and l_i are members of $\mathbf{L} \cup \{Ok\}$ [*].

Example 7 : Terms such as *(Maxint+3)–4* can be recovered into their final value ; and *at the same time* we can amalgamate all terms that contain a negative subterm over an additional constant *CRASH* :

$n \in NEGATIVE$	$\Rightarrow$	n	=	$CRASH$
		$succ(CRASH)$	=	$CRASH$
		$pred(CRASH)$	=	$CRASH$
		$CRASH - n$	=	$CRASH$
		$n - CRASH$	=	$CRASH$
		$CRASH + n$	=	$CRASH$
		$n + m$	=	$m + n$
$n \in TOO\text{-}LARGE$	$\Rightarrow$	$n+0$	=	n
$n+succ(m) \in TOO\text{-}LARGE$	$\Rightarrow$	$n+succ(m)$	=	$succ(n+m)$
$succ(n) \in TOO\text{-}LARGE$	$\Rightarrow$	$pred(succ(n))$	=	n
$n \in TOO\text{-}LARGE$	$\Rightarrow$	$n–0$	=	n
$n \in TOO\text{-}LARGE$	$\Rightarrow$	$n–succ(m)$	=	$pred(n)–m$

Each term that contains a negative value in its subterms is equal to *CRASH*. Every other term is amalgamated with its normal form ($succ^i(0)$), (even if this form is not an okay one).

---

[*] The l_i's are *not* necessarily distinct.

The semantics of **Gen-Ax** works directly on the *values* of A, in a straightforward manner.

Definition 14 : The algebra A validates **Gen-Ax** iff : for each axiom of **Gen-Ax**,

$$[t_1 \in l_1 \wedge \cdots \wedge t_n \in l_n \wedge v_1 = w_1 \wedge \cdots \wedge v_m = w_m] \quad \Rightarrow \quad v = w \ ,$$

and for each substitution σ with range in *A*, the following holds :
if $\sigma(t_i)$ belongs to A_{l_i} for all i, and $\sigma(v_j) = \sigma(w_j)$ for all j, then $\sigma(v) = \sigma(w)$ in *A*.

Example 8 : We have shown (Example 5) that the evaluation of the term *pred(succ(Maxint))* fails via the okay axioms, this term is exceptional. Nevertheless, *pred(succ(Maxint))* is recovered via our generalized axioms of Example 7, using the axiom :

$$succ(n) \in TOO\text{-}LARGE \quad \Rightarrow \quad pred(succ(n)) = n \ .$$

It suffices to show that *succ(Maxint)* is labeled with *TOO-LARGE* ; which results from the first labelling axiom of Example 6. Thus, the term *pred(succ(Maxint))* is recovered into the class of *Maxint*.

Definition 15 : Let **SPEC**=<**Σ**-*Exc*,**St-Frm**,**St-Exc**,**Ok-Ax**,**Lbl-Ax**,**Gen-Ax**> be an exception specification. A **Σ**-*Exc*-algebra, A, is a **SPEC**-algebra iff it validates all parts of **SPEC**. We denote the full subcategory of Alg(**Σ**-*Exc*) containing the **SPEC**-algebras by Alg(**SPEC**).

8. INITIALITY RESULTS

In this section, we show that Alg(**SPEC**) has an initial object. Our main result is more general ; it extends the major technical result of the classical abstract data type theory [ADJ 76].

Theorem 1 : Let **SPEC** be an exception-specification over **Σ**-*Exc*. Let A be a **Σ**-*Exc*-algebra, and let *R* be a binary relation over A compatible with the sorts of **Σ**-*Exc*. There is a least congruence over A, denoted by $\equiv_{\mathbf{SPEC},R}$, and there are least subsets of $(A/\equiv_{\mathbf{SPEC},R})$, denoted by $\{(A/\equiv_{\mathbf{SPEC},R})_{l_i}\}$, such that $(A/\equiv_{\mathbf{SPEC},R})$ is a **SPEC**-algebra and $\equiv_{\mathbf{SPEC},R}$ contains *R*.

Proof : given in appendix.

Corollary 1 : The category Alg(**SPEC**) has an initial object, denoted by $T_{\mathbf{SPEC}}$.

Proof : From the definition of exception morphisms, it is clear that the **SPEC**-algebra $T_{\mathbf{SPEC}}$, obtained by Theorem 1 with A=$T_{\mathbf{\Sigma}\text{-}Exc}$ and *R*=∅, gives the answer (since $T_{\mathbf{\Sigma}\text{-}Exc}$ is already initial in Alg(**Σ**-*Exc*)). □

Example 9 : With the specification **SPEC** of bounded natural numbers given in sections 3 through 7, the initial algebra is defined as follows :
$T_{\mathbf{SPEC}}$ = {*CRASH*}∪**N**, with operations 0 *succ pred* + and − as usual. Every negative value is amalgamated with *CRASH*, and every operation applied over *CRASH* gives *CRASH*. Moreover, $\mathbf{N}_{NEGATIVE}$ is equal to {*CRASH*}, $\mathbf{N}_{TOO\text{-}LARGE}$ is equal to *]Maxint,+∞[* and $\mathbf{N}_{Ok}$ is equal to *[0,Maxint]*.

9. STRUCTURED EXCEPTION SPECIFICATIONS

9.1. Forgetful functors

Definition 16 : Let **Σ**-Exc_1 =< $\mathbf{S}_1, \mathbf{\Sigma}_1, \mathbf{L}_1$ > and **Σ**-Exc_2 =< $\mathbf{S}_2, \mathbf{\Sigma}_2, \mathbf{L}_2$ > be two exception signatures such that **Σ**-$Exc_1 \subset$ **Σ**-Exc_2. We define the *forgetful functor* U from Alg(**Σ**-Exc_2) to Alg(**Σ**-Exc_1) in a similar manner as in the classical abstract data type theory :

- □ for each **Σ**-Exc_2-algebra B=$(B,\{B_l\})$, U(B) is the **Σ**-Exc_1-algebra A=$(A,\{A_l\})$ such that *A* (resp. A_l for each $l \in \mathbf{L}_1 \cup \{Ok\}$) is the subset of *B* (resp. B_l) corresponding to the sorts of $\mathbf{S}_1$ (i.e. we remove the subsets associated with the sorts of $\mathbf{S}_2 - \mathbf{S}_1$). The $\mathbf{\Sigma}_1$-operations work over A as they do over B.
- □ for each **Σ**-Exc_2-morphism μ: B → B' , $U(\mu)$ is the **Σ**-Exc_1-morphism μ, restricted to U(B) and corestricted to U(B').

Unfortunately, given two specifications $\mathbf{SPEC}_1 \subset \mathbf{SPEC}_2$, if B is a $\mathbf{SPEC}_2$-algebra, then U(B) is a **Σ**-Exc_1-algebra but is not always a $\mathbf{SPEC}_1$-algebra. This is due to the following fact : if $\mathbf{SPEC}_2$ adds some standard exceptions to the operations of $\mathbf{SPEC}_1$, then it is possible that it removes some $\mathbf{SPEC}_1$ okay standard forms. Thus, several occurrences of $\mathbf{SPEC}_1$-okay axioms are inhibited. There are then several $\mathbf{SPEC}_2$-algebras that do not validate $\mathbf{Ok\text{-}Ax}_1$.

9.2. Presentations

Definition 17 : A *presentation* over the exception specification $\mathbf{SPEC}_1$ is a tuple

$$\mathbf{PRES} = < \mathbf{S} , \mathbf{\Sigma} , \mathbf{L} , \mathbf{St\text{-}Frm} , \mathbf{St\text{-}Exc} , \mathbf{Ok\text{-}Ax} , \mathbf{Lbl\text{-}Ax} , \mathbf{Gen\text{-}Ax} >$$

such that $\mathbf{SPEC}_2 = \mathbf{SPEC}_1 + \mathbf{PRES}$ is an exception specification, $<\mathbf{S}_0, \mathbf{\Sigma}_0> \cap <\mathbf{S},\mathbf{\Sigma}>$ is empty, and for all $\mathbf{SPEC}_2$-algebras, A, the $\mathbf{\Sigma}$-Exc_1-algebra U(A) is a $\mathbf{SPEC}_1$-algebra.
The specification $\mathbf{SPEC}_1$ is called the *predefined* specification.

This definition is not a very constructive one. Nevertheless, we shall give a sufficient condition under which **PRES** is a presentation.

Proposition 2 : If, for each elementary declaration of **St-Exc** of the form

$$[t_1 \in \text{St-Exc} \wedge \cdots \wedge t_n \in \text{St-Exc}] \quad \Rightarrow \quad t \in \text{St-Exc} ,$$

the leading operator symbol of t belongs to $\mathbf{\Sigma}$, then **PRES** is a presentation over $\mathbf{SPEC}_1$.
This means that the standard exceptions added by **PRES** are only preconditions on the new operations. There must not be any new standard exceptions with a predefined operation at the top.
Proof : Standard exceptions are closed under partial evaluations, but this evaluation only concerns *strict* subterms. The leading operator is never avoided. Thus, each new standard exception contains a new operation at the top ; and the presentation cannot remove predefined standard forms. Consequently, it cannot remove any occurrence of a predefined okay axiom. □

Example 10 : We define the following presentation **PRES** over $\mathbf{SPEC}_1 = NAT + BOOL$, in order to specify bounded arrays of natural numbers :

$\mathbf{S} = \{ ARRAY \}$

$\mathbf{\Sigma} = \{ create , \_ [\_]:=\_ , \_ [\_] \}$ (with usual arities)

$\mathbf{L} = \{ OUT\text{-}OF\text{-}RANGE , NOT\text{-}INITIALIZED \}$

St-Frm :

$$\left.\begin{array}{r} t \in \text{St-Frm} \wedge n \in \text{St-Frm} \\ \wedge\ i \in \text{St-Frm} \wedge Maxrange < i = False \end{array}\right\} \Rightarrow \quad t[i]:=n \in \text{St-Frm}$$

$$create \in \text{St-Frm}$$

St-Exc = ∅ [because **St-Frm** already contains "$Maxrange<i = False$" in the premisses]

Ok-Ax :

$$eq?(i,j) = False \Rightarrow (t[i]:=n)[j]:=m = (t[j]:=m)[i]:=n$$
$$i = j \Rightarrow (t[i]:=n)[j]:=m = t[j]:=m$$

Lbl-Ax :

$$create[i] \in NOT\text{-}INITIALIZED$$
$$t[i] \in NOT\text{-}INITIALIZED \wedge eq?(i,j) = False \Rightarrow (t[j]:=n)[i] \in NOT\text{-}INITIALIZED$$
$$succ^{Maxrange-1}0 < i = True \Rightarrow t[i] \in OUT\text{-}OF\text{-}RANGE$$
$$i \in NEGAT \Rightarrow t[i] \in OUT\text{-}OF\text{-}RANGE$$
$$succ^{Maxrange-1}0 < i = True \Rightarrow t[i]:=n \in OUT\text{-}OF\text{-}RANGE$$
$$i \in NEGAT \Rightarrow t[i]:=n \in OUT\text{-}OF\text{-}RANGE$$

Gen-Ax : ∅ [... for simplicity, but we can specify recoveries, ad libidum]

Proposition 2 ensures that **PRES** is a presentation over *NAT* and *BOOL*. Notice that this specification is an example where standard forms are not normal forms.

9.3. Synthesis functors

Definition 18 : The *synthesis functor* associated with the presentation **PRES** is the functor, F, from Alg($\mathbf{SPEC}_1$) to Alg($\mathbf{SPEC}_2$), defined by means of Theorem 1 as follows :

- for each $\mathbf{SPEC}_1$-algebra, A, the morphism $eval$: $T_{\mathbf{\Sigma}_1(A)} \to$ A defines a binary relation in $T_{\mathbf{\Sigma}_2(A)}$ by :

$$x R y \iff eval(x)=eval(y) \text{ for all } x \text{ and } y \text{ in } T_{\mathbf{\Sigma}_1(A)} \quad [*].$$

From Theorem 1, we know that there is a least congruence over $T_{\Sigma_2(A)}$, $\equiv_{\mathbf{SPEC}_2,R}$, generated by R, such that $F(A)=(T_{\Sigma_2(A)}/\equiv_{\mathbf{SPEC}_2,R})$ [together with smallest subsets, $F(A)_l$, containing A_l] is a $\mathbf{SPEC}_2$-algebra. The $\mathbf{SPEC}_2$-algebra $F(A)$ is called the *synthesis* of A.

□ for each $\mathbf{SPEC}_1$-morphism, μ: A → A', $F(\mu)$ is the $\mathbf{SPEC}_2$-morphism from $F(A)$ to $F(A')$ deduced in a unique way from the Σ_2-morphism $\bar{\mu}$: $T_{\Sigma_2(A)} \rightarrow T_{\Sigma_2(A')}$.

Example 11 : Let A = {*CRASH*}∪**N** ∪ {*True,False*} be a (*NAT+BOOL*)-algebra, as in Example 9. Let **PRES** be the presentation of bounded arrays from Example 10. The synthesized algebra $F(A)$ associated with **PRES** is described as follows :

Every array that contains only okay natural numbers in the range *0..Maxrange* is an okay one. Every array that contains an operation using an index in {*CRASH*}∪]*Maxrange*,+∞[is erroneous (*OUT-OF-RANGE*). Every array that contains an erroneous natural number (∈ {*CRASH*}∪]*Maxint*,+∞[) is erroneous (by implicit error propagation rule).

Moreover, the predefined sorts contain new erroneous values : those obtained by taking a value from outside of the range *0..Maxrange* ; those obtained by taking a value from a non initialized index ; and those obtained by taking a value from an erroneous array (implicit error propagation rule). These new values are not predefined ones, except if the generalized axioms of **PRES** amalgamates them with *CRASH*, or recovers them.

Notice that the labeled subset $F(A)_{OUT\text{-}OF\text{-}RANGE}$ contains both numbers and arrays. This is an example of an exception-algebra where a labeled subset intersects several sorts.

Theorem 2 : The synthesis functor F is a left adjoint for the forgetful functor U. This means that for each $\mathbf{SPEC}_1$-algebra, A, and for each $\mathbf{SPEC}_2$-algebra, B, $\mathrm{Hom}_{\mathbf{SPEC}_2}(F(A), B)$ is isomorphic to $\mathrm{Hom}_{\mathbf{SPEC}_1}(A, U(B))$.

Proof : Let I_A be the $\mathbf{SPEC}_1$-morphism from A to $U(F(A))$ deduced from the identity over A in a unique way. The pair (A,I_A) is a universal arrow from A to U, resulting from the definition of $F(A)$ (Theorem 1). Thus, the Yoneda lemma ([McL 71], III.2) proves our theorem. Notice that the family I_A, for A in Alg($\mathbf{SPEC}_1$), is then the unit of adjunction. □

9.4. Hierarchical consistency

In the classical abstract data type theory, hierarchical consistency means that **PRES** does not amalgamate predefined values. This means that the unit of adjunction is *injective*. With exception handling, we must also verify that **PRES** do not add predefined labels to some predefined values :

Definition 19 : Let I be the unit of adjunction I: $T_{\mathbf{SPEC}_1} \rightarrow U(F(T_{\mathbf{SPEC}_1}) = U(T_{\mathbf{SPEC}_2})$. The presentation **PRES** is *hierarchically consistent* iff I is injective and for all predefined labels $l \in \mathbf{L}_1$, we have : $I(T_{\mathbf{SPEC}_1,l}) = U(T_{\mathbf{SPEC}_2})_l \cap I(T_{\mathbf{SPEC}_1})$.

In the categorical framework, this means that I is *partially retractable* [**].

Example 12 : The *ARRAY* presentation specified in Example 10 is hierarchically consistent. But if we add the axiom : *0 ∈ TOO-LARGE* , **PRES** is not hierarchically consistent any more, since the predefined value 0 becomes labeled with the predefined label *TOO-LARGE*.

9.5. Sufficient completeness

In the classical abstract data type theory, sufficient completeness means that **PRES** does not add new values to the predefined sorts. This means that the unit of adjunction is *surjective*. In exception handling, such a definition is too restrictive. Sufficient completeness should allow presentations to add erroneous values into the predefined sorts. For instance, each value of the form *t[i]*, with *i>Maxrange*, is a new predefined value ; but the presentation is sufficiently complete, since *t[i]* is erroneous (labeled with *OUT-OF-RANGE*).

Definition 20 : The presentation **PRES** is *sufficiently complete* iff the unit of adjunction I satisfies :

$$U(\ T_{\mathbf{SPEC}_2} - T_{\mathbf{SPEC}_2,err}\ ) \quad \subset \quad I(\ T_{\mathbf{SPEC}_1}\ )$$

This means that the presentation **PRES** must not add new non erroneous values to the predefined sorts.

Example 13 : The *ARRAY* presentation specified in Example 10 is sufficiently complete. But if we remove the axiom : *create[i] ∈ NOT-INITIALIZED* , then **PRES** is no longer sufficiently complete, since *create[i]* is not erroneous any more and is not amalgamated with a predefined value (*create[i]* is then incompletely specified).

---

[*] recall that $T_{\Sigma_1(A)} \subset T_{\Sigma_2(A)}$

[**] In the classical abstract data type theory, injective morphisms, monomorphisms and partially retractable morphisms are the same. In our exception handling formalism, monomorphisms are injective morphisms, but are not always partially retractable.

10. CONCLUSION

In this paper, we have shown how exception handling can be integrated into algebraic specifications without losing the use of congruences, the existence of least congruences and the existence of initial models. We must point out that to guarantee the existence of least congruences, we do not need to introduce any restriction on exception specifications. For instance, even if in most examples, axioms can be transformed into canonical term rewriting systems, this condition is never required. We do not introduce any restrictions on the class of models taken into account, i.e. we do not restrict ourselves to finitely generated algebras or to the ground term algebra. This allows our results to hold in a very general framework. It should be noted that the key idea is to distinguish *exceptions* and *errors*, and this is made possible by working at the level of $T_{\Sigma(A)}$. Indeed the formalism described in this paper relies on this simple but powerful idea.

What is especially important is that once the initiality results are guaranteed for exception specifications, the classical specification-building primitives are easily extended to our framework. We have carefully detailed how enrichment carries over to our exception specifications, and how hierarchical consistency and sufficient completeness can be suitably redefined. In the same way, parameterization may be extended to exception specifications, since it mainly relies on initiality, synthesis functors and pushouts (see [ADJ 80]). As a last remark, we want to emphasize the fact that the concepts of abstract implementation developed in [EKMP 80], and [BBC 86], may also be extended to exception specifications (cf. [Ber 84] and [Ber 86] respectively). This fact is especially important since realistic examples of abstract implementations can hardly be designed without exception handling (e.g. the implementation of bounded queues by means of bounded arrays).

ACKNOWLEDGEMENTS
This work is partially supported by CNRS GRECO de Programmation, and by the ESPRIT Projects METEOR and FOR-ME-TOO.

11. APPENDIX

This appendix contains the technical proofs omitted in the body of the article.

Proof of Proposition 1 : We follow the usual method for minimality proofs :
□ The set, **C**, of all congruences satisfying the IF..THEN condition is not empty : it contains at least the trivial congruence.
□ Now, we show that the congruence, $\equiv_{Ok}$, equal to the conjunction of all the congruences in **C**, is still in **C**. Thus, we want to prove that $\equiv_{Ok}$ satisfies the IF..THEN condition.
Assume that $\equiv_{Ok}$, σ and the okay axiom $([v_1 = w_1 \wedge \cdots v_n = w_n] \Rightarrow v{=}w)$ satisfy the three conditions of the IF statement. We want to prove that $\sigma(v) \equiv \sigma(w)$ for all "≡" in **C**. Thus, since each ≡ in **C** satisfies the IF..THEN condition, it suffices to prove that each ≡ in **C** satisfies these three conditions. The first condition is clear, since it is independent of ≡ (*eval* is intrinsic to A). The second condition results from the fact that $\equiv_{Ok}$ is the conjunction of all ≡ in **C** : this implies that $t_i \equiv \alpha_i$ for all i and all ≡. The same reasoning applies to the third condition, which ends our proof. □

Proof of Theorem 1 :

We will prove a more general result. Theorem 1 means that there is a least **SPEC**-algebra, B=(A/ $\equiv_{\mathbf{SPEC},R}$), finitely generated over A and compatible with *R*. We will prove that this result can be extended to the non finitely generated algebras :

Theorem 1*b* **:** Let **SPEC** be an exception-specification over the exception-signature **Σ**-*Exc*. Let A be a **Σ**-*Exc*-algebra, and let *R* be a binary relation over A compatible with the sorts of **Σ**-*Exc*. There is a least **SPEC**-algebra, B, and an exception morphism $\mu: A \to B$ such that : if xRy then $\mu(x) = \mu(y)$.

Proof : Let **F** be the family of all **SPEC**-morphisms, ν: A → Z , where Z is a **SPEC**-algebra and ν is compatible with *R*. **F** is not empty : it contains at least the trivial morphism τ: A → **S**.

Let B be the quotient of A such that the surjective **Σ**-*Exc*-morphism, μ: A → B , is defined by : $\mu(x) = \mu(y)$ iff $\nu(x) = \nu(y)$ for all ν in **F** ; and $\mu(x) \in B_l$ iff $\nu(x) \in Z_l$ for all ν in **F**.
μ is clearly an exception morphism since all ν in **F** are exception morphisms. Thus, it suffices to prove that B is a **SPEC**-algebra ; i.e. that B validates **St-Frm**, **St-Exc**, **Ok-Ax**, **Lbl-Ax** and **Gen-Ax**. Two lemmas are needed. Notice that, from the definition of B, there is an exception morphism, ν': B → Z , for each algebra Z in **F**. In the following, for each algebra Z in **F**, $\bar{\nu}$: $T_{\mathbf{\Sigma}(B)} \to T_{\mathbf{\Sigma}(Z)}$ denotes the **Σ**-morphism deduced in a unique way from ν'.

Lemma 1 : $\bar{\nu}(Ok\text{-}Frm_B)$ is included in $Ok\text{-}Frm_Z$.

Lemma 2 : The congruence $\bar{\nu}^{-1}(\equiv_{Z,Ok})$ contains the congruence $\equiv_{B,Ok}$ (in $T_{\Sigma(B)} \times T_{\Sigma(B)}$).

For lack of space, we do not prove these lemmas (proved in [Ber 86]). The first lemma results from minimality properties of $St\text{-}Frm_B$ and $St\text{-}Exc_Z$. The second one results from minimality properties of $\equiv_{B,Ok}$.

□ The validation of **St-Frm** and **St-Exc** means that $eval(Ok\text{-}Frm_B) \subset B_{Ok}$. This results from Lemma 1, from the fact that $eval(Ok\text{-}Frm_Z) \subset Z_{Ok}$ for all Z in **F**, and from the definition of B_{Ok} : $x \in B_{Ok}$ iff $\nu'(x) \in Z_{Ok}$ for all Z in **F**.
□ The validation of **Ok-Ax** means that if $t \equiv_{B,Ok} t'$ then *eval(t)=eval(t')* in B. This results from Lemma 2, from the fact that each Z in **F** validates **Ok-Ax**, and from the definition of B.
□ For the same reasons, the validation of **Lbl-Ax** and **Gen-Ax** results directly from the definition of B, since the semantics of **Lbl-Ax** and **Gen-Ax** is directly defined in B (not via $T_{\Sigma(B)}$).
This ends our proof. □

12. 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 79] Thatcher J., Wagner W., Wright J. : "Data type specification: parameterization and the power of specification techniques", Proc. of SIGACT 10th Annual Symposium on Theory of Computing, 1979.

[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] Bernot G., Bidoit M., Choppy C. : "Abstract implementations and correctness proofs", Proc. 3rd STACS, January 1986, Springer-Verlag LNCS.

[Ber 84] Bernot G. : "Implémentations de types abstraits algébriques en présence d'exceptions", DEA Report, LRI, Orsay, Sept. 1984.

[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, Université de Paris-Sud, 1986.

[Bid 82] Bidoit M. : "Algebraic data types: structured specifications and fair presentations", Proc. of 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.

[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.

[EPE 81] Engels G., Pletat V., Ehrich H. : "Handling errors and exceptions in the algebraic specification of data types", Osnabruecker Schriften zur Mathematik, July 1981.

[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.

[Gog 77] Goguen J.A. : "Abstract errors for abstract data types", Formal Description of Programming Concepts E.J. NEUHOLD Ed., North Holland, New York

[Gog 78] Goguen J.A. : "Exceptions and error sorts, coercion and overloading operators", SRI Research Report, 1978.

[Gut 79] Guttag J.V. : "Notes on type abstraction (Version 2)", IEEE Transactions on Software Engineering, 1979.

[Loe 81] Loeckx J. : "Algorithmic specifications of abstract data types", ICALP 1981.

[McL 71] Mac Lane S. : "Categories for the working mathematician", Graduate texts in mathematics, 5, Springer-Verlag, 1971.

[Pla 82] Plaisted D. : "An initial algebra semantics for error presentations", Unpublished Draft, 1982.

[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.

[Wir 83] Wirsing M. : “Structured algebraic specifications: a kernel language”, Habilitation thesis, Technische Universitat Munchen, 1983.