A. B. Cremers and H.-P. Kriegel, editors, Proc. 6. GI-Fachtagung für Theoretische Informatik, LNCS 145, pages 141–151, Berlin, 1983. Springer–Verlag

ALGEBRAIC AND OPERATIONAL SEMANTICS OF EXCEPTIONS AND ERRORS

M.Gogolla, K.Drosten, U.Lipeck
Abteilung Informatik, Universität Dortmund
Postfach 500500, D-4600 Dortmund 50

H.D.Ehrich
Lehrstuhl B für Informatik, TU Braunschweig
Postfach 3329, D-3300 Braunschweig

Abstract

The specification of abstract data types requires the possibility to treat exceptions and errors. We present an approach allowing all forms of error handling : error introduction, error propagation and error recovery. The algebraic semantics of our method and a new correctness criterion is given. We also introduce an operational semantics of a subclass of our specifications which coincides with the algebraic semantics.

Key Words

Specification of abstract data types, error and exception handling, algebraic semantics , correctness of specifications , operational semantics.

1. Introduction

Abstract data types offer promising tools for the specification and implementation of programs. Research in this field has been initiated by [Gu 75, LZ 74]. Pleasant features of the method are that it is well-founded algebraically [ADJ 78, Eh 79, EKTWW 81, WPPDB 80] and operationally [Ro 73, Hu 77, O'D 77, Wa 77] and that it is a sound basis for specification languages.
The problem of handling exceptions and errors in abstract data types has been studied in [GHM 77, ADJ 78, Go 78.1, Go 78.2, Ma 79, Bl 80] and an operational treatment has been given in [EPE 81].
We here modify the approach of [EPE 81] and study the algebraic and operational semantics of specifications allowing error and exception handling. We distinguish syntactically between error introducing and normal functions and allow two different types of variables for the same sort. Thus all forms of error and exception handling, i.e. error introduction, error propagation and error recovery, may be treated. We avoid the strict propagation of errors as in [ADJ 78,Go 78.1,Ma 79], the transformation of axioms via new operations as in [ADJ 78] and the introduction of a semi-lattice structure on the set of sorts as in [Go 78.2].
Section 2 gives an informal introduction to our method. In sections 3 and 4 we present the algebraic semantics of our specifications. Section 5 gives a new correctness criterion and section 6 introduces the operational semantics of a subclass of our specifications and shows that it coincides with the algebraic semantics. Because of space limitations, we omit the proofs.

2. The basic idea

The natural numbers are an example of a simple data type, which needs error and exception handling. We are going to use the natural numbers in different versions throughout the paper.

Example 2.1

To specify the natural numbers we need the following functions :

```
0 : ---> nat
pred, succ : nat ---> nat
add, times : nat x nat ---> nat
```

The axioms are given by :

```
pred(succ(n))=n                          ( 1 )
pred(0)=error                            ( 2 )
add(0,n)=n                               ( 3 )
add(succ(n),m)=succ(add(n,m))            ( 4 )
times(0,n)=0                             ( 5 )
times(succ(n),m)=add(times(n,m),n)       ( 6 )
```

But what is the semantics of an axiom like pred(0)=error ? If we treat 'error' as an extra constant in nat, we will find times(0,pred(0)) = times(0,error) = 0 and so the introduced error has been forgotten by axiom (5). This might suggest the idea that errors should propagate and so we could add the following axioms :

```
succ(error)=error      ( E1 )
pred(error)=error      ( E2 )
add(error,n)=error     ( E3 )
add(n,error)=error     ( E4 )
times(error,n)=error   ( E5 )
times(n,error)=error   ( E6 )
```

But unfortunately we didn't really specify what we had intended, because unwanted contradictions occur. 0 = times(0,error) = error holds due to equations (5) and (E6) and so $succ^n(0)$ = error for every $n \in N_0$ due to (E1).

There are other reasons which don't support the idea of strict error propagation. For example consider a straightforward specification of the factorial function on the natural numbers :

```
fac(n) = if(eq(n,0),succ(0),times(n,fac(pred(n))))
```

With the error propagation idea in mind we will find :

```
fac(0) = if(eq(0,0),succ(0),times(0,fac(pred(0))))
       = if(true,succ(0),times(0,fac(error)))
       = if(true,succ(0),times(0,error))
       = if(true,succ(0),error) = error                    ***
```

We present an approach which allows all forms of error handling, i.e. error introduction, error propagation and error recovery, to be treated in an easy way and which avoids difficulties like the above. Our main instruments are :

- the partition of the carrier sets into a normal and an error part,
- the syntactical classification of functions into those which introduce errors in normal situations and those which preserve ok states and
- the introduction of two types of variables. The first type will serve for non error situations only, the other for ok and exceptional states as well.

Example 2.2

We give a specification for the intended natural number algebra including an error constant.

```
0 : ---> nat
succ : nat ---> nat
pred : nat ---> nat : unsafe
add , times : nat x nat ---> nat
error : ---> nat : unsafe
```

Functions, which might introduce errors and error constants, unsafe functions as we call them, are indicated in the signature by ': unsafe' The other functions are called ok functions.
The signature gives a classification for all terms t. If an unsafe function occurs in t, it is not known whether an error might be introduced or not and so t is viewed as a possible error and called unsafe term. If only ok functions occur in t, we know t corresponds to a natural number. In this case t is called an ok term.
We mark pred as an unsafe function, because it introduces an error when applied to the ok value 0. For the functions succ, add and times we know that they will return ok values when they are applied to such ones.
The ok part of the intended carrier set are terms of the form $succ^n(0)$ with $n \in N_0$ corresponding to the natural numbers. The error part of the carrier set are terms in which the function symbol 'error' occurs. These terms can be seen as error messages informing about illegal application of pred to 0.
We now use ok variables n and m which means they serve for non error situations only, i.e. only for ok terms. The axioms are exactly (1) - (6) of example 2.1, but there are semantical differences. It is not allowed to substitute for example the term 'error' for variable n in axiom (5) and therefore the difficulties described in example 2.1 do not occur.
Now the following identities hold :

- pred(succ(succ(pred(succ(0))))) = pred(succ(succ(0))) = succ(0)
  Please note it is not allowed to substitute succ(pred(succ(0))) into the ok variable n in axiom (1), but we may substitute the semantically equivalent term succ(0). This will be clarified later.
- times(0,pred(0)) = times(0,error)
  No further simplifications can be made on the last term, because axiom (5) cannot be applied. The term can be seen as an error message within its environment. ***

3. Algebras with ok predicates

In this and the following two sections we show how the results of [ADJ 78] carry over to our notion of algebra. The syntax of our many-sorted algebras with ok predicates is defined via a signature, which gives names corresponding to the sorts of the carrier sets and to the operations on these sets. Our carrier sets are not homogeneous, because we want to distinguish between normal situations and exceptions. For this reason we already mark in the signature those function symbols, which correspond to operations introducing errors in normal situations.

Definition 3.1
A signature (with ok predicate) is a quintupel $(S,\Sigma,arity,sort,ok_\Sigma)$, where
(1) S is a set (of sorts) , Σ is a set (of function symbols) and
(2) arity , sort and ok_Σ are mappings :
arity : $\Sigma \longrightarrow S^*$, sort : $\Sigma \longrightarrow S$ and $ok_\Sigma : \Sigma \longrightarrow BOOL$.

- A signature $(S,\Sigma,arity,sort,ok_\Sigma)$ will often be denoted by Σ only.
- Given a function symbol σ, $arity(\sigma)=s1...sn$ denotes the sorts of the arguments, $sort(\sigma) = s$ gives the result sort. This is written as $\sigma : s1 \times ... \times sn \longrightarrow s$.

- $ok_\Sigma(\sigma)$=TRUE means σ is an ok function symbol, while $ok_\Sigma(\sigma)$=FALSE indicates an unsafe function symbol. The notion of ok and unsafe functions will be made clear by the following definition.

Definition 3.2
Let signature Σ be given. A Σ-algebra (with ok predicates) is a triple (A,F,ok_A), where
(1) $A=\langle A_s\rangle_{s\in S}$ is an S-indexed family of sets,
(2) $F=\langle\sigma_A\rangle_{\sigma\in\Sigma}$ is a Σ-indexed family of functions, for every function symbol σ : s1 x ... x sn ---> s we have a function
$\sigma_A : A_{s1} \times \ldots \times A_{sn} \rightarrow A_s$ and
(3) $ok_A=\langle ok_s\rangle_{s\in S}$ is an S-indexed family of predicates,
$ok_s : A_s$ ---> BOOL such that
(4) for every function symbol σ : s1 x ... x sn ---> s with $ok_\Sigma(\sigma)$= TRUE and $(a1,\ldots,an) \in A_{s1} \times \ldots \times A_{sn}$ with $ok_{si}(ai)$=TRUE for i=1,...,n, we have $ok_s(\sigma_A(a1,\ldots,an))$=TRUE.

- A Σ-algebra (A,F,ok_A) will often be denoted by A only. Whenever no ambiguity arises, we will omit indices of the predicates : for a function symbol σ ok(σ) means $ok_\Sigma(\sigma)$ and for $a\in A_s$ ok(a) means $ok_s(a)$. If ok(σ)=TRUE holds, we call σ_A an ok function, otherwise an unsafe function.
- For $a\in A_s$ $ok_s(a)$=TRUE will mean a is an ok element of A_s, otherwise it is called error element. The ok predicates split the carrier sets into an ok part A_{ok} and an error part A_{err}.
$A_{ok}=\langle A_{ok,s}\rangle_{s\in S}$, $A_{ok,s}=\{ a\in A_s \mid ok_s(a)=TRUE \}$.
$A_{err}=\langle A_{err,s}\rangle_{s\in S}$, $A_{err,s}=\{ a\in A_s \mid ok_s(a)=FALSE \}$.
For the application we have in mind the ok elements correspond to normal situations, while the error elements indicate exceptional states.
- Part (4) of the definition requires that ok functions yield ok values for ok arguments, or in other words, only unsafe functions may introduce errors when applied to normal situations. Because we have to treat these unsafe functions carefully, we already distinguish them syntactically from ok functions. This garantees that whenever an expression consisting only of ok functions is applied to ok arguments, this will result in an ok element. For expressions including unsafe functions this is not known.
- Every algebra with ok predicates can also be interpreted as a conventional algebra without ok predicates by just omitting the predicates. On the other hand every conventional algebra without ok predicates can be made into an algebra with ok predicates by demanding all functions to be ok functions and all elements to be ok elements. The same holds for signatures.

Example 3.3
We describe the natural numbers together with an extra error element, which is introduced because we want to apply the predecessor function to 0. Let S = { bool,nat } be the set of sorts. Unsafe function symbols will be indicated by ': unsafe'.

```
false,true : ---> bool
0 : ---> nat
succ : nat ---> nat
pred : nat ---> nat : unsafe
negative : ---> nat : unsafe
if : bool x nat x nat ---> nat
```

The carrier sets and the ok predicates on them are given by :

$A_{bool} = \{ f , t \}$ $\quad ok_{bool}(b) = TRUE$

$A_{nat} = N_0 + \{ e_{nat} \}$ $\quad ok_{nat}(n) = (n \in N_0)$

The functions corresponding to the function symbols are defined by :

$false_A : |---> f$ $\quad 0_A : |---> 0$

$true_A : |---> t$ $\quad negative_A : |---> e_{nat}$

$$succ_A : n \ |---> \begin{cases} n+1 & \text{if } n \in N_0 \\ e_{nat} & \text{if } n = e_{nat} \end{cases}$$

$$pred_A : n \ |---> \begin{cases} n-1 & \text{if } n \in N \\ e_{nat} & \text{if } n = 0 \text{ or } n = e_{nat} \end{cases}$$

$$if_A : (b,n1,n2) \ |---> \begin{cases} n1 & \text{if } b=t \\ n2 & \text{if } b=f \end{cases}$$

$pred_A$ is an unsafe function, because only for some ok arguments it yields ok results, for the ok value 0 it returns the error element e_{nat}. ***

For every signature Σ we define the term algebra with ok predicates in the following way.

Definition 3.4

Let signature Σ be given. The Σ-term algebra $(T_\Sigma, F_\Sigma, ok_T)$ is defined by :

(1) (T_Σ, F_Σ) is the usual term algebra. $T_\Sigma = \langle T_s \rangle_{s \in S}$. $F_\Sigma = \langle \sigma_T \rangle_{\sigma \in \Sigma}$.

(2) $ok_T = \langle ok_s \rangle_{s \in S}$. For $t \in T_s$ ok_s is given by :

$$ok_s(t) = \begin{cases} FALSE & \text{if an unsafe function symbol occurs in } t \\ TRUE & \text{otherwise} \end{cases}$$

- A term is an ok term, if and only if all function symbols occurring in the term are ok function symbols.
- Our term algebras are well defined, that means T_Σ is a Σ-algebra with ok predicates satisfying part (4) of definition 3.2.

Example 3.5

Looking at the signature in example 3.3 we find if(true,0,succ(0)) is an ok term and pred(succ(0)), pred(0) or if(true,0,negative) are error elements in the term algebra. ***

Σ-algebras with ok predicates may be compared by structure preserving mappings called Σ-algebra morphisms.

Definition 3.6

Let Σ-algebras $(A1, F1, ok_{A1}), (A2, F2, ok_{A2})$ be given. An S-indexed family of functions $h = \langle h_s \rangle_{s \in S}$, $h_s : A1_s \dashrightarrow A2_s$ is called Σ-algebra morphism, if

(1) h is a morphism from A1 to A2 taken without ok predicates and

(2) for $s \in S$ and $a \in A1_s$ $ok_{A1}(a)$ implies $ok_{A2}(h_s(a))$.

A morphism is called injective respectively surjective, if every h_s is injective respectively surjective.

A morphism is called strict, if for $s \in S$ and $a \in A1_s$ $ok_{A1}(a) = ok_{A2}(h_s(a))$.

- Because of the additional predicate structure on signatures and algebras we require in part (2) that no ok element may be mapped onto an error element or in other words that the ok property for elements is preserved by our morphisms.
- The isomorphisms are the injective, surjective and strict morphisms. The strictness property is necessary, because we do not only want the operational structure but also the predicate structure to be respected by isomorphisms.

- h_{ok} and h_{err} denote the restrictions of h to ok and error elements.
  $h_{ok} = \langle h_{ok,s} \rangle_{s \in S}$, $h_{ok,s} : A1_{ok,s} \dashrightarrow A2_{ok,s}$.
  $h_{err} = \langle h_{err,s} \rangle_{s \in S}$, $h_{err,s} : A1_{err,s} \dashrightarrow A2_{s}$.
  For strict morphisms we can denote $h_{err,s}$ of course by :
  $h_{err,s} : A1_{err,s} \dashrightarrow A2_{err,s}$.

Example 3.7

The following is an example of a morphism between algebras with ok predicates and a motivation for the freedom of allowing error elements to be mapped to ok elements. We give a morphism from the term algebra of example 3.5 into the algebra of example 3.3. It is the uniquely determined morphism between these algebras taken without ok predicates.

$h_{bool} : T_{bool} \dashrightarrow A_{bool}$
$\quad$ false $\mapsto$ f
$\quad$ true $\mapsto$ t

$h_{nat} : T_{nat} \dashrightarrow A_{nat}$
$\quad$ 0 $\mapsto$ 0
$\quad$ negative $\mapsto$ e_{nat}
$\quad$ succ(t) $\mapsto$ $succ_A(h_{nat}(t))$
$\quad$ pred(t) $\mapsto$ $pred_A(h_{nat}(t))$
$\quad$ if(b,t1,t2) $\mapsto$ $if_A(h_{bool}(b), h_{nat}(t1), h_{nat}(t2))$

Obviously, h respects the operations and h preserves ok elements. It sends each term to the result of the corresponding evaluation in A. So succ(succ(0)) will naturally be mapped to 2 and of course the error (or unsafe) terms pred(succ(succ(0))), if(true,0,negative) and pred(0) will result in 1, 0 and e_{nat}, respectively.
The next theorem confirms that this morphism is the only morphism between such algebras, i.e. our term algebras are initial. ***

Theorem 3.8

Let signature Σ, term algebra T_Σ and Σ-algebra A be given. Then there exists a unique morphism $h : T_\Sigma \dashrightarrow A$, or in other words, T_Σ is initial in the class of all Σ-algebras.

4. Specifications

An important difference between our specification technique and the usual algebraic specification without error handling is that we introduce two different types of variables for the same sort. Variables of the first type will serve for the ok part of the corresponding carrier set only, variables of the second type for the whole carrier set.

Definition 4.1

Let signature Σ be given. A pair (V, ok_V) is called a set of variables (with ok predicates) for Σ, if

(1) $V = \langle V_s \rangle_{s \in S}$ is an S-indexed, pairwise disjoint family of sets (of variables) , each V_s disjoint from Σ and

(2) $ok_V = \langle ok_{V,s} \rangle_{s \in S}$, $ok_{V,s} : V_s \dashrightarrow$ BOOL is an S-indexed family of predicates.

- Again, a set of variables (V, ok_V) is often denoted by V only.
- When no ambiguity arises, ok(v) means $ok_{V,s}(v)$ for $v \in V_s$. In analogy to ok and unsafe functions we use the notions of ok and unsafe variables.

Definition 4.2

Let signature Σ, Σ-algebra A and variables V be given. An assignment to (or interpretation of) the variables is an S-indexed family of functions $I=\langle I_s\rangle_{s\in S}$, $I_s : V_s \dashrightarrow A_s$ such that $ok_{V,s}(v)$ implies $ok_s(I_s(v))$ for $s\in S$ and $v\in V_s$.

- If ok(v)=TRUE holds, it is not allowed to assign an error element to v, ok(v)=FALSE indicates that v may hold ok or error values.

Also, for our notions of algebra and morphism there always exist free algebras.

Lemma 4.3

Let signature Σ, variables V, Σ-algebra A and assignment $I : V \dashrightarrow A$ be given. Then there exists a Σ-algebra $T_\Sigma(V)$, such that there is a unique Σ-algebra morphism $\underline{I} : T_\Sigma(V) \dashrightarrow A$, extending I in the sense that $I_s(v)=\underline{I}_s(v)$ for $s\in S$ and $v\in V_s$.

The notions of Σ-equation, of equations satisfied by a Σ-algebra and of congruence relation on an algebra are defined as usual.
But please note, our definition of assignment implies that there is a restriction to the substitution of variables. An equation may be valid although it does not hold for error elements substituted for ok variables.

Example 4.4

Let n, n1+ and n2+ be variables of sort nat with ok(n)=TRUE and ok(n1+)=ok(n2+)=FALSE. Then the algebra of example 3.3 satisfies the following equations (among others).

pred(succ(n))=n (1) pred(0)=negative (2)
if(false,n1+,n2+)=n2+ (3) if(true,n1+,n2+)=n1+ (4)
succ(negative)=negative (5) pred(negative)=negative (6)

But, for example the equation

succ(pred(n))=n

does not hold, because $succ_A(pred_A(0))=succ_A(e_{nat})=e_{nat}$. ***

Given a Σ-algebra A and a congruence relation $\equiv$ on it, the quotient $A/\equiv$ of A by $\equiv$ can be made into a Σ-algebra with ok predicates by defining the carrier sets and operations in the usual way and by letting a class be ok, if and only if there is an ok element of the algebra in it. In this sense TRUE dominates FALSE with respect to the ok predicate of a class.

Defintion 4.5

Let signature Σ, Σ-algebra A and congruence $\equiv = \langle\equiv_s\rangle_{s\in S}$ be given.
(1) ($A/\equiv$, $F_{A/\equiv}$) denotes the usual quotient of an algebra by a congruence relation on it.
(2) $ok_{A/\equiv}=\langle ok_{\equiv,s}\rangle_{s\in S}$ is an S-indexed family of predicates.

$$ok_{\equiv,s}([a])=\begin{cases}\text{TRUE} & \text{if there is a } b\in[a] \text{ with } ok_{A,s}(b)=\text{TRUE}\\ \text{FALSE} & \text{otherwise}\end{cases}$$

- $(A/\equiv, F_{A/\equiv}, ok_{A/\equiv})$ is a Σ-algebra with ok predicates satisfying part (4) of our definition for algebra.

For a given set of equations E with variables the induced set of constant equations $E(T_\Sigma)$ and the generated least congruence relation denoted by $\equiv_E = \langle\equiv_{E,s}\rangle_{s\in S}$ are defined in the usual way. There always exists such a $\equiv_E$, since we know that there always is a least congruence generated by a given relation, if we deal only with algebras without ok predicates and our congruence definition didn't

involve the predicates. For brevity we often denote $\equiv_E$ by $\equiv$ and $a \equiv_{E,s} b$ by $a \equiv b$, if no ambiguities arise.

Example 4.6

If we look at the equations of example 4.4, we find the following pairs are in $E(T_\Sigma)_{nat}$ due to the first equation :

<pred(succ(0)),0>
<pred(succ(succ(0))),succ(0)>

But the following pairs are not in $E(T_\Sigma)_{nat}$:

<pred(succ(negative)),negative>
<pred(succ(pred(succ(0)))),pred(succ(0))>

On the other hand, the last pair is in the congruence relation generated by $E(T_\Sigma)$:

<pred(succ(0)),0> ε $E(T_\Sigma)_{nat}$ ===> pred(succ(0)) ≡ 0 ===>
succ(pred(succ(0))) ≡ succ(0) ===>
pred(succ(pred(succ(0)))) ≡ pred(succ(0)) ≡ 0 ***

The pleasant thing about our approach to error and exception handling is that the fundamental initiality result of [ADJ 78] is still valid.

Theorem 4.7

$T_\Sigma/\equiv_E$ is initial in the class of all Σ-algebras satisfying E, i.e. given a Σ-algebra A, which satisfies E, we have a unique morphism $g : T_\Sigma/\equiv_E \dashrightarrow A$.

Remark

$T_\Sigma/\equiv_E$ is denoted by $T_{\Sigma,E}$ and called the quotient term algebra.

Example 4.8

The quotient of T_Σ in example 3.5 by the equations in example 4.4 is isomorphic to the algebra of natural numbers in example 3.3. ***

We now know that for given signature Σ, variables V and equations E, there always exists an initial Σ-algebra which can be chosen as a standard semantics. So we put together signatures, variables and equations as usual, getting a specification.

Definition 4.9

A specification is a triple (Σ,V,E), where Σ is a signature with ok predicate, V is a set of variables with ok predicates and E is a set of Σ-equations.

5. Correctness of specifications

The usual notion of correctness of specifications - the isomorphism between the specified algebra and the given model - is somewhat too strong for our purpose. Our main interest lies in the ok part of the carrier sets. The cruical point is that terms like succ(error) and pred(error) in example 2.2 are error elements, but it is not important here that they are different. So we allow different error elements of the specified algebra to be identified in our model.

Definition 5.1

Let specification (Σ,V,E) and Σ-algebra A be given. (Σ,V,E) is called correct with respect to A, if

(1) there is a strict morphism $h : T_{\Sigma,E} \dashrightarrow A$ such that
(2) $h_{ok} : T_{\Sigma,E,ok} \dashrightarrow A_{ok}$ is bijective and
(3) $h_{err} : T_{\Sigma,E,err} \dashrightarrow A_{err}$ is surjective.

(Σ,V,E) is strongly correct with respect to A, if it is correct with respect to A and the morphism h is an isomorphism.

- The conventional notion of correctness of specifications for algebras without ok predicates may be embedded into this correctness criterion, because then we only deal with ok functions and ok elements and so $T_{\Sigma,E,err} = A_{err} = \emptyset$.

Example 5.2

We give a correct specification for the algebra A defined in example 3.3. The axioms E are identical to equations (1) - (4) of example 4.4. $T_{\Sigma,E}$ is described by a canonical term algebra using the context-free languages defined by the following productions.

```
<bool>     ::= false | true
<nat>      ::= <nat-ok> | <nat-err>
<nat-ok>   ::= 0 | succ( <nat-ok> )
<nat-err>  ::= negative | succ( <nat-err> ) | pred( <nat-err> )
```

$T_{\Sigma,E,bool} = L(\langle bool \rangle)$ $\quad ok_{bool}(b)=TRUE$

$T_{\Sigma,E,nat} = L(\langle nat \rangle)$ $\quad ok_{nat}(n)=(n \in L(\langle nat\text{-}ok \rangle))$

The operations in $T_{\Sigma,E}$ are defined in the usual way, e.g.

$$pred_{\Sigma,E} : t \longmapsto \begin{cases} succ^{n-1}(0) & \text{if } t=succ^{n}(0) \text{ and } n>0 \\ negative & \text{if } t=0 \\ pred(t) & \text{if } t \in L(\langle nat\text{-}err \rangle) \end{cases}$$

We now define a mapping $h : T_{\Sigma,E} \longrightarrow A$.

$$h_{bool} : T_{\Sigma,E,bool} \longrightarrow A_{bool}$$

$$true \mapsto t$$

$$false \mapsto f$$

$$h_{nat} : T_{\Sigma,E,nat} \longrightarrow A_{nat}$$

$$t \longmapsto \begin{cases} n & \text{if } t \in L(\langle nat\text{-}ok \rangle),\ t=succ^{n}(0) \\ e_{nat} & \text{if } t \in L(\langle nat\text{-}err \rangle) \end{cases}$$

To prove that h is a strict morphism, we have to show :

(1) $h(\sigma_{\Sigma,E}(a1,\ldots,an))=\sigma_A(h(a1),\ldots,h(an))$ holds for every $\sigma \in \Sigma$.

(2) $ok_{\Sigma,E}(a)=ok_A(h(a))$ for $s \in S$ and $a \in A_s$.

It is easy to see that the mapping h respects the operations. The strictness of h, its bijectivity on the ok part and its surjectivity on the error part can be seen directly from its definition. The specified algebra is correct with respect to the algebra A of example 3.3 , although all error elements are mapped to the one error element in A. If we want to get a strongly correct specification, we have to add equations (5) and (6) of example 4.4. ***

6. Operational semantics of specifications

A set of equations can be viewed as a set of rewrite rules interpreting equations from left to right. By substituting constant terms for the variables we get a set of constant rewrite rules. These rules determine a reduction process on terms which stops if none of the axioms can be applied further. In this way we give an operational semantics for specifications which is well-defined if the set of constant rewrite rules has the finite church-rosser property.

Definition 6.1

Let specification (Σ,V,E) and the set of constant equations $E(T_\Sigma)$ be given.

$\longrightarrow_E = \langle \longrightarrow_s \rangle_{s \in S}$ is the family of relations on T_Σ defined by :

(1) If $\langle t,t' \rangle \in E(T_\Sigma)_s$, then $t \longrightarrow_s t'$.

(2) If $\sigma : s1 \times \ldots \times sn \longrightarrow s$, $ti \in T_{si}$ for $i = 1,\ldots,n$ and $j \in \{ 1,\ldots,n \}$ with $tj \longrightarrow_{sj} tj'$ are given, then

$\sigma(t1,\ldots,tj,\ldots,tn) \longrightarrow_s \sigma(t1,\ldots,tj',\ldots,tn)$

$\rightarrow_E^* = \langle \rightarrow_s^* \rangle_{s \in S}$ is the reflexive and transitive closure of $\rightarrow_E$ and called the family of subterm replacements induced by E.
A term t of sort s has the normal form t', if $t \rightarrow_s^* t'$ and there is no $t' \rightarrow_s \overline{t}$. This is denoted by nf(t)=t'.

Example 6.2
The normal forms of example 5.2 are identical to the elements of the carrier set of the given canonical term algebra. ***

Definition 6.3
$\rightarrow_E^*$ is called finite church-rosser, if every term t of sort s has a normal form, and if $t \rightarrow_s^* \overline{t}$ and $t \rightarrow_s^* \overline{t}'$, then there is a t' with $\overline{t} \rightarrow_s^* t'$ and $\overline{t}' \rightarrow_s^* t'$.

- If $\rightarrow_E^*$ is finite church-rosser, each term has a unique normal form.

Example 6.4
For the specifications in examples 2.2 and 5.2 the families of subterm replacements $\rightarrow_E^*$ are finite church-rosser. ***

Definition 6.5
Let specification (Σ,V,E) with finite church-rosser $\rightarrow_E^*$ be given. The normal form algebra (NF,F_{NF},ok_{NF}) is defined by :
(1) $NF = \langle NF_s \rangle_{s \in S}$. NF_s are the normal forms of sort s.
(2) $F_{NF} = \langle \sigma_{NF} \rangle_{\sigma \in \Sigma}$. For σ : s1 x ... x sn ---> s and normal forms ti of sort si for i=1,...,n , the function σ_{NF} is given by : $\sigma_{NF}(t1,...,tn) = nf(\sigma(t1,...,tn))$.
(3) $ok_{NF} = \langle ok_{NF,s} \rangle_{s \in S}$. For a normal form t of sort s $ok_{NF,s}$ is defined by :

$$ok_{NF,s}(t) = \begin{cases} \text{TRUE} & \text{if there is an ok term } t' \text{ with } nf(t')=t \\ \text{FALSE} & \text{otherwise} \end{cases}$$

- (NF,F_{NF},ok_{NF}) is a Σ-algebra with ok predicates satisfying part (4) of our definition for algebra.
- A normal form t is ok in the normal form algebra if and only if there is an ok term t' which has t as its normal from. In this sense the ok terms dominate the error terms, or in other words if an error term is equivalent to an ok term this 'heals' the error term. If the rules are ok term preserving, which means there is no $\langle t,t' \rangle \in E(T_\Sigma)$ with ok(t)=TRUE and ok(t')=FALSE, the ok predicates in the normal form algebra are determined by the normal forms themselves.

Example 6.6
If we compare the normal form algebra and the quotient term algebra of example 5.2, we find they are isomorphic. ***

Theorem 6.7
Let specification (Σ,V,E) with finite church-rosser $\rightarrow_E^*$ be given. Then the quotient term algebra $T_{\Sigma,E}$ and the normal form algebra NF are isomorphic.

Acknowledgements

We thank Udo Pletat and especially Gregor Engels for their earlier work in the field and many fruitful discussions.

References

ADJ 78 Goguen,J.A./Thatcher,J.W./Wagner,E.G. : An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types. Current Trends in Programming Methodology, Vol. IV (R.T.Yeh, ed.). Prentice Hall, Englewood Cliffs, 1978, pp. 80-149.

Bl 80 Black,A.P.: Exception Handling and Data Abstraction. IBM Research Report RC 8059, 1980.

Eh 79 Ehrich,H.-D.: On the Theory of Specification, Implementation and Parametrisation of Abstract Data Types. Journal ACM, Vol.29, 1982, pp. 206 - 227.

EKTWW 81 Ehrig,H./Kreowski,H.-J./Thatcher,J.W./Wagner,E.G./Wright,J.B.: Parameter Passing in Algebraic Specification Languages. Proc. Workshop on Algebraic Specification, Aarhus, 1981.

EPE 81 Engels,G./Pletat,U./Ehrich,H.-D.: Handling Errors and Exceptions in the Algebraic Specification of Data Types. Osnabrücker Schriften zur Mathematik, Reihe Informatik, Heft 3, Univ. Osnabrück, 1981.

GDLE 82 Gogolla,M./Drosten,K./Lipeck,U./Ehrich,H.D. : Algebraic and Operational Semantics of Specifications Allowing Exceptions and Errors. Forschungsbericht Nr. 140, Abteilung Informatik, Univ. Dortmund, 1982. [Long Version of this Paper including the Proofs].

GHM 77 Guttag,J.V./Horowitz,E./Musser,D.R.: Some Extensions to Algebraic Specifications. SIGPLAN Notices, Vol. 12, No. 3, March 1977, pp. 63-67.

Go 78.1 Goguen,J.A.: Abstract Errors for Abstract Data Types. Proc. Conf. on Formal Description of Programming Concepts (E.J. Neuhold, ed.), North-Holland, Amsterdam, 1978.

Go 78.2 Goguen,J.A.: Order Sorted Algebras : Exception and Error sorts, Coercions and Overloaded Operators. Semantics and Theory of Computation Report No. 14, University of California, Los Angeles, Dec. 1978.

Gu 75 Guttag,J.V.: The Specification and Application to Programming of Abstract Data Types. Techn. Report CSRG-59, Univ. of Toronto, 1975.

Hu 77 Huet,G.: Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. Proc. 18th IEEE Symp. on Foundations of Computer Science, 1977, pp. 30-45.

LZ 74 Liskov,B./Zilles,S.: Programming with Abstract Data Types. SIGPLAN Notices Vol. 9, No. 4, April 1974, pp. 50-59.

Ma 79 Majster,M.E.: Treatment of Partial Operations in the Algebraic Specification Technique. Proc. Specifications of Reliable Software, IEEE, 1979, pp. 190-197.

O'D 77 O'Donnell,M.J.: Computing in Systems Described by Equations. LNCS 58, Springer Verlag, New York, 1977.

Ro 73 Rosen, B.K. : Tree-Manipulating Systems and Church-Rosser Theorems. Journal ACM, Vol. 20, 1973, pp. 160-187.

Wa 77 Wand,M.: Algebraic Theories and Tree Rewriting Systems. Technical Report No. 66, Indiana Univ., Bloomington, Indiana, July 1977.

WPPDB 80 Wirsing,M. / Pepper,P. / Partsch,H. / Dosch,W./ Broy,M. : On Hierachies of Abstract Data Types. Bericht TUM-I8007,Institut für Informatik, Technische Univ. München, Mai 1980.