ALGEBRAIC SPECIFICATION WITH SUBSORTS USING DECLARATIONS

M.Gogolla
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

Conventional algebraic specifications describe data types some times only implicit with hidden sorts and functions. A more powerful and elegant technique can be achieved by putting more structure into the specification. Our approach distinguishes between basis sorts and subsorts. The subsorts, whose carriers are included in the basis sort carriers, are described by declarations, i.e. a list of terms possibly with variables of basis as well as subsorts. We study the initial algebra semantics of our specifications, which include declarations and equations.

Key words

Abstract data type, algebraic specification, algebraic semantics, subsort, declaration, equation, hidden sorts and functions.

1. Introduction

Conventional algebraic specifications describe data types some times only indirectly with hidden sorts and functions. Therefore the structure of the type may be concealed in the hidden part. For example a specification of the integer numbers could have a hidden sort for the natural numbers and an explicit conversion from natural to integer numbers. But a natural number is an integer number and therefore a term denoting a natural number should also denote an integer number without any explicit conversion operator. This is possible in our approach.

We distinguish between basis sorts and subsorts. The carriers of the subsorts are included in the basis sort carriers and the subsorts are described by declarations, this means that terms of the corresponding basis sort are given. A basis sort can include several subsorts having equal rights, but more complex structures can be build up by using declarations. Our main results are initiality theorems in the spirit of [ADJ 78] and it is proved that our specifications can be expressed in the conventional algebraic framework using hidden sorts and functions. A later paper will generalize the approach to arbitrary partial orderings on sorts.

Martin Gogolla, Hans-Dieter Ehrich: Algebraic Specifications with Subsorts Using Declarations. Bulletin of the EATCS, No. 21 (1983), pp. 31-38.

The general idea of declarations has been presented in [Go 78], where a semi-latice of sorts is taken as a basis, but there are some difficulties with that approach. Declarations are also introduced in [Wa 82], but in that paper there is only one basis sort (in our terminology) and a very liberal notion of signature. Similar problems of handling coercions and overloaded operators are also discussed in [Re 80].

The approach presented here generalizes our results according error and exception handling in [GDLE 83], where we introduced for each sort a subsort representing the ok part of the data type. This is quite different from the view of errors and exceptions taken in [Go 78], where disjoint ok and error subsorts for each sort are wanted.

2. The basic idea

It is known from the literature that it is not possible to specify the equality predicate on integer numbers finetely equational without the use of hidden functions. But in our approach, which allows equations as well as declarations, we can give - in our opinion - a straightforward and short specification without hidden functions for the equality predicate.

Example

```
spec integer numbers with equality
srts int , bool
opns 0 : ---> int ;
     _-1 , _+1 : int x int ---> int ;
     false , true : ---> bool ;
     _≡_ : int x int ---> bool
subs int : nat
vars i , j : int ; n : nat
decs 0 : nat ;
     n+1 : nat
eqns (i-1)+1 = (i+1)-1 = i ;
     0≡0 = true ;
     n+1≡0 = 0≡n+1 = false ;
     i+1≡j+1 = i≡j
ceps
```

Examples of computations are :

$$3\equiv 3 = 2\equiv 2 = 1\equiv 1 = 0\equiv 0 = \text{true}$$

$$-1\equiv -3 = 0\equiv -2 = 1\equiv -1 = 2\equiv 0 = \text{false}$$ ***

Under the keyword srts (for sorts) a list of basis sorts is given, the operations are defined on these basis sorts. For each basis sort a

list of its formal subsorts can appear under the keyword subs (for subsorts). Variables may be defined for basis sorts as well as subsorts. Under the keyword decs (for declarations) a list of declarations is allowed. A declaration consists of a term of basis sort s (possibly with variables) and a subsort r of the basis sort s. This guarantees that whenever the term is evaluated the result will be an element belonging not only to the basis sort but to the subsort as well.

In the example, where the integers have the natural numbers as a subsort, the two declarations state the following :

- 0 is not only an integer number, but also a natural number.
- Whenever there is a natural number n and the successor function +1 is applied, then this will yield a natural number again.

In the equations the subsort variables are actualized only by those terms, which the declarations forced to be of subsort nature. By this the range of variables (and therefore the range of equations) can be restricted to subsets of the carriers.

To summon up the differences between our approach and the usual pure equational specification technique, we state that

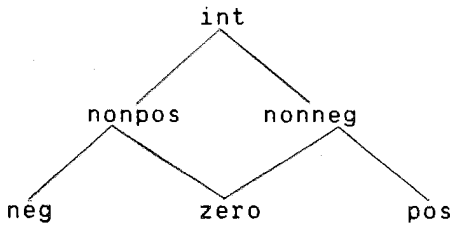
- we distinguish between basis sorts and subsorts. The carriers of the subsorts are included in the basis sort carriers.
- we describe the subsorts by declarations. This means that we give terms of the corresponding basis sort.
- we allow variables for the basis sorts and subsorts in the declarations and equations.

Example

As a second example we show how subsorts again can be structured and can have others, not necessarily disjoint subsorts.

```
spec integers with negative and positive numbers as subsorts
srts int
opns 0 : ---> int ;
     _-1 , _+1 : int ---> int
subs int : neg , zero , pos , nonpos , nonneg
vars i : int ; n : neg ; p : pos
decs 0 : zero ;
     0-1 , n-1 : neg ;
     0+1 , p+1 : pos ;
     0 , n : nonpos ;
     0 , p : nonneg
eqns (i-1)+1 = (i+1)-1 = i
ceps
```

A declaration of the form v : s with v a variable of sort r expresses that $A_r \subseteq A_s$. So we have structured the integers in the following way :



***

3. Signatures, algebras and morphisms

Definition

A signature (for an algebra with subsorts) is given by a six-tuple $(\overline{S},\Sigma,arity,sort,S',sub)$, where

(1) $\overline{S}$ is a set (of basis sorts) , Σ is a set (of function symbols),

(2) $arity : \Sigma \dashrightarrow \overline{S}^*$ and $sort : \Sigma \dashrightarrow \overline{S}$ are mappings and

(3) S' is a set (of subsorts) disjoint from $\overline{S}$ and sub is a mapping, $sub : \overline{S} \dashrightarrow P(S')$ such that $sub(s) \cap sub(r) = \emptyset$ if $s \neq r$.

- We will use the abbreviations Σ for $(\overline{S},\Sigma,arity,sort,S',sub)$, $\overline{\Sigma}$ for the corresponding conventional signature $(\overline{S},\Sigma,arity,sort)$ and S for the union of $\overline{S}$ and S'.
- Condition (3) means that sub is a partition of S'. But it only requires the disjointness of the subsort sets belonging to different basis sorts. It is not demanded that the subsorts of a given basis sort have to be disjoint. Such union structures can be build up by declarations.
- This definition of signature does not require that there exist elements belonging to subsorts. This will be guaranteed again by declarations.

Definition

Let signature Σ be given. A Σ-algebra (with subsorts) is a tuple (A,F), where

(1) $A=\langle A_s \rangle_{s \in S}$ is an S-indexed family of sets, such that $A_r \subseteq A_s$ if r is a subsort of s and

(2) $F=\langle \sigma_A \rangle_{\sigma \in \Sigma}$ is a Σ-indexed family of functions, every function symbol $\sigma : s1 \times \ldots \times sn \dashrightarrow s$ in Σ corresponds to a function $\sigma_A : A_{s_1} \times \ldots \times A_{s_n} \dashrightarrow A_s$.

- We will use A as an abbreviation for (A,F) and $\overline{A}$ for the corresponding conventional algebra without the subsort structure.
- Condition (1) requires that the subsort carriers are included in the corresponding basis sort carriers.
- Due to our definition of signature domain and codomain of functions are carriers of basis sorts.

Definition

Let signature Σ be given. The Σ-term algebra (T_Σ, F_Σ) with $T_\Sigma = \langle T_s \rangle_{s \in S}$ and $F_\Sigma = \langle \sigma_T \rangle_{\sigma \in \Sigma}$ is the $\overline{\Sigma}$-term algebra $\overline{T_\Sigma}$ with $T_s = \emptyset$ if s is a subsort.

Definition

Let signature Σ and Σ-algebras A and B be given. An $\overline{S}$-indexed family of mappings $f = \langle f_s \rangle_{s \in \overline{S}}$, $f_s : A_s \dashrightarrow B_s$ is called Σ-algebra morphism from A to B, if

(1) f is a morphism between $\overline{A}$ and $\overline{B}$ and

(2) $f_s(A_r) \subseteq B_r$ if r is a subsort of s.

- Condition (1) means that f respects the operational structure and clause (2) claims that f respects the subsort structure as well.
- It suffices to view a morphism as an $\overline{S}$-indexed family of mappings, because the subsort carriers are included in the corresponding basis sort carriers.

Theorem

Let signature Σ and term algebra T_Σ be given . Then T_Σ is initial in the category of all Σ-algebras.

Definition

Let signature Σ and Σ-algebra A be given. An S-indexed, pairwise disjoint family of sets $V = \langle V_s \rangle_{s \in S}$ denotes variables for Σ.

An assignment to (or interpretation of) the variables is an S-indexed family of mappings $I = \langle I_s \rangle_{s \in S}$, $I_s : V_s \dashrightarrow A_s$.

Definition

Let signature Σ and variables V be given. The extended signature $(\overline{S}, \Sigma(V), arity_{\Sigma(V)}, sort_{\Sigma(V)}, S', sub)$ is given by :

(1) $\Sigma(V) = \Sigma \cup \bigcup_{s \in S} V_s$

(2) $arity_{\Sigma(V)}(\sigma) = \begin{cases} \lambda & \text{if } \sigma \in V \\ arity(\sigma) & \text{if } \sigma \in \Sigma \end{cases}$

(3) $sort_{\Sigma(V)}(\sigma) = \begin{cases} r & \text{if } \sigma \in V_s \text{ and s is a subsort of r} \\ s & \text{if } \sigma \in V_s \text{ and s is a basis sort} \\ sort(\sigma) & \text{if } \sigma \in \Sigma \end{cases}$

- The result of applying the forgetful functor $U : ALG_{\Sigma(V)} \dashrightarrow ALG_\Sigma$ to $T_{\Sigma(V)}$ is denoted by $T_\Sigma(V)$. In $T_\Sigma(V)$ the operations corresponding to variables are forgotten and $T_\Sigma(V)_s = \emptyset$ if s is a subsort.
- A single variable v of subsort r is viewed as a term of sort s in $T_\Sigma(V)$, where s is the basis sort of r. But variables of subsort type only hold subsort values due to the definition of assignment.

Lemma

Let signature Σ, variables V, Σ-algebra A and assignment $I : V \dashrightarrow A$ be given. Then there is a unique Σ-algebra morphism $I^\# : T_\Sigma(V) \dashrightarrow A$ that extends I in the sense that $I(v) = I^\#(v)$ if $v \in V$.

4. Declarations

Definition

Let signature Σ, variables V, $T_\Sigma(V)$ and Σ-algebra A be given. A declaration is a tuple $t : s$ with $t \in T_\Sigma(V)_r$ and s a subsort of r. A declaration $t : s$ is true in A, iff for all assignments $I : V \dashrightarrow A$ $I^\#(t) \in A_s$.

- A declaration $t : s$ assures that for all interpretations I of the variables the resulting evaluation $I^\#(t)$ will not only be in the basis carrier but in the subsort carrier A_s as well.

Definition

Let signature Σ, variables V and declarations D be given. $(T_{\Sigma,D}, F_{\Sigma,D})$ denotes the Σ-term algebra generated by the declarations.

$T_{\Sigma,D} = \langle T_{D,s} \rangle_{s \in S}$. $F_{\Sigma,D} = F_\Sigma$.

$T^1_{D,s} = T_s$

$T^{j+1}_{D,s} = T^j_{D,s} \cup \{ t_c(t_1,\ldots,t_n) \mid t_c(v_1,\ldots,v_n) : s \in D \ , \ v_i : s_i \ , \ t_i \in T^j_{D,s_i} \}$

$T_{D,s} = \bigcup_{j \in N} T^j_{D,s}$

- The inductive definition of the term algebra generated by a set of declarations D starts with all terms of basis sorts and then successively adds all terms, which can be produced by a declaration, if the variables are replaced by already generated terms.
- By $t_c(v_1,\ldots,v_n) : s$ we mean that the declaration uses the set $\{v_1,\ldots,v_n\}$ of variables and has a constant part t_c consisting only of operation symbols. Of course t_c may be empty, if the declaration has the form $v : s$ with v a variable of sort r. This expresses $A_r \subseteq A_s$. $t_c(t_1,\ldots,t_n)$ results from $t_c(v_1,\ldots,v_n)$ by substituting all occurences of the variable v_i by a term t_i with $i=1,\ldots,n$.

Theorem

Let signature Σ, variables V and declarations D be given. Then $T_{\Sigma,D}$ is initial in the category of all Σ-algebras satisfying D.

5. Equations

Definition

Let signature Σ and Σ-algebra A be given. An $\overline{S}$-indexed family of relations $\equiv = \langle \equiv_s \rangle_{s \in \overline{S}}$ is called a congruence on A, iff it is a congruence on the corresponding conventional algebra $\overline{A}$.

$A/\equiv = \langle A/\equiv_s \rangle_{s \in S}$.

$$A/\equiv_s = \begin{cases} \overline{A}/\equiv_s & \text{if } s \text{ is a basis sort} \\ \{[a] \in \overline{A}/\equiv_r \mid a \in A_s\} & \text{if } s \text{ is a subsort of basis sort } r \end{cases}$$

$F/\equiv = \overline{F}/\equiv$.

- In the factorisation of an algebra by a congruence relation we take the usual congruence classes, if we have a basis sort. If we consider a subsort, we take all congruence classes of the corresponding basis sort that are generated by the elements of the subsort.
- It suffices to view a congruence as an $\overline{S}$-indexed family of relations. An S-indexed family can be obtained by defining $a \equiv_s a'$ for $a, a' \in A_s$ and s a subsort, iff $a \equiv_r a'$ with r the basis sort of s.

Lemma

Let signature Σ, Σ-algebra A and congruence $\equiv$ on A be given. Then $(A/\equiv, F/\equiv)$ is a Σ-algebra and there is an epimorphism $f_{\equiv} : A \dashrightarrow A/\equiv$.

Remark

If A satisfies a set of declarations D, then the factorisation $A/\equiv$ satisfies D as well.

Definition

Let signature Σ, variables V, $T_\Sigma(V)$ and Σ-algebra A be given. An equation is a tuple $l = r$ with $l, r \in T_\Sigma(V)_s$ and $s \in \overline{S}$. An equation $l = r$ is true in A, iff for all assignments $I : V \dashrightarrow A$ $I^\#(l) = I^\#(r)$.

Definition

Let signature Σ, variables V, declarations D and equations E be given.
$E(T_{\Sigma,D}) = \{ \langle I^\#(l), I^\#(r) \rangle \mid l=r \in E \text{ and } I : V \dashrightarrow T_{\Sigma,D} \text{ assignment} \}$
$\equiv$ denotes the congruence generated by $E(T_{\Sigma,D})$.

- The constant equations induced by a set of declarations and equations are all pairs of terms that can be produced if the variables are substituted by elements of the term algebra generated by the declarations.

Theorem

Let signature Σ, variables V, declarations D and equations E be given and let $T_{\Sigma,D,E}$ denote the factorisation of $T_{\Sigma,D}$ by $\equiv$. Then $T_{\Sigma,D,E}$ is initial in the category of all Σ-algebras satisfying D and E.

6. Declarations versus equations

Theorem

For every specification with declarations (Σ, D, E) there is an equational specification (Σ', E') with hidden sorts and functions, such that $\overline{T_{\Sigma,D,E}} \cong T_{\Sigma',E'}$.

Remark

Let us shortly describe how the equational specification (Σ', E') looks like :

- The non hidden part of Σ' is identical to $\overline{\Sigma}$. This means that we have
  - all basis sorts and
  - all operation symbols.

- For every subsort s of basis sort r we introduce
  - a hidden sort s and
  - a hidden function $mk_s : s \dashrightarrow r$.
- For every declaration $d = t_c(v_1,\ldots,v_n,v_{n+1},\ldots,v_m) : s$ with
  - $v_1:s_1,\ldots,v_n:s_n$ the distinct variables of subsorts in d and
  - $v_{n+1}:s_{n+1},\ldots,v_m:s_m$ the distinct variables of basis sorts in d

  introduce
  - a hidden function $f_d : s_1 \times \ldots \times s_m \dashrightarrow s$ and
  - an equation

    $mk_s(f_d(v_1,\ldots,v_m)) = t_c(mk_{s_1}(v_1),\ldots,mk_{s_n}(v_n),v_{n+1},\ldots,v_m)$.
- Transform each equation

  $l_c(v_1,\ldots,v_n,v_{n+1},\ldots,v_m) = r_c(v_1,\ldots,v_n,v_{n+1},\ldots,v_m)$

  (with the same conventions for the variables as above) into

  $l_c(mk_{s_1}(v_1),\ldots,mk_{s_n}(v_n),v_{n+1},\ldots,v_m) = r_c(mk_{s_1}(v_1),\ldots,mk_{s_n}(v_n),v_{n+1},\ldots,v_m)$.

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.

GDLE 83 Gogolla,M./Drosten,K./Lipeck,U./Ehrich,H.D.: Algebraic and Operational Semantics of Exceptions and Errors. Proceedings 6th GI-Conference Theoretical Computer Science, Dortmund 1983, LNCS 145, pp. 141-151. Also : Forschungsbericht Nr. 140, 1982, Abteilung Informatik, Universität Dortmund.

Go 78 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.

Re 80 Reynolds,J.C. : Using Category Theory to Design Implicit Conversions and Generic Operators. Semantics Directed Compiler Generation (Neil D. Jones, ed.). Springer Verlag, 1980, LNCS 94, pp. 211-258.

Wa 82 Wadge,W.W.: Classified Algebras. University of Warwick, Theory of Computation, Report No. 46, October 1982.