PUC

Monografias em Ciência da Computação

Nº 3/83

A METHODOLOGY FOR THE SOUND AND COMPLETE SPECIFICATION OF ABSTRACT DATA TYPES

by

Paulo A. S. Veloso

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453

RIO DE JANEIRO – BRASIL

Series: Monografias em Ciência da Computação

Nº 3/83

Series Editor: A.L. Furtado January 1983

A METHODOLOGY FOR THE SOUND AND COMPLETE SPECIFICATION OF ABSTRACT DATA TYPES *

by

Paulo A. S. Veloso

* Research partly sponsored by CNPq, FINEP and IBM do Brasil

ABSTRACT

This paper outlines and justifies a methodology for the sound and complete specification of abstract data types viewed as initial algebras. This methodology helps overcoming the problems of what equations to write and when to stop writing them by means of canonical term algebras and symbolic transformations. The methodology is justified by showing that the equations obtained always form a sound and complete specification of the given model. Furthermore, the methodology is shown to be non-obtrusive, in that it does not prevent finding such a specification it one exists at all.

Key words:

abstract data types, initial algebra,
specification methodology,
canonical term algebra,
algebraic specification,
symbolic manipulations

RESUMO

Este trabalho esboça e justifica uma metodologia para a especi - ficação correta e completa de tipos abstratos de dados vistos como álgebras iniciais. Esta metodologia ajuda a suplantar os problemas de que equações escrever e de quando parar de escrevê-las, o que é feito através de álgebras de termos canônicos e transformações simbólicas. A metodologia é justificada mostrando-se que as equações obtidas sempre constituem uma especificação correta e completa para o modelo dado. Além disso, a metodologia é não obtrusiva, no sentido de não impedir que se ache uma tal especificação, se é que existe.

Palavras chaves:

tipos abstratos de dados,
álgebra inicial,
metodologia para especificação,
álgebra de termos canônicos,
especificação algébrica
manipulações simbólicas

1. Introduction

Abstract data types provide a powerful and elegant tool for program construction and verification [4,7]. Such uses, however, require formal specifications of the data types. And the task of formally specifying a model given informally - sometimes even vaguely - is laborious and error-prone [6].

Within the initial-algebra approach [3,9] a data type is specified by a (finite) set of (conditional) equations, the corresponding abstract data type being the (isomorphism class of the) quotient of the algebra of terms by the congruence generated by the equations.

Here we outline a methodology to help overcoming some problems usually faced in obtaining an algebraic specification for a data type: what equations to write and when to stop writing them. We also show that this methodology has two desirable properties. First, the resulting set of equations will always be a (correct and complete) specification for the data type. Second, by following it we are not provented from finding an algebraic specification if one ideed exists. Deciding the question of existence of an algebraic specification with some prescribed properties (finite, recursive, etc.) is not the purpose of the methodology.

2. Methodology

Consider a very simple data type, the natural numbers with the operations zero, successor and predecessor. Clearly every natural number can be uniquely represented by a term involving only successor and zero. Thus, symbolic manipulations on such terms can be used to describe the operations of the naturals. An axiomatic description, say , by (conditional) equations, provides a formal specification of the data type. This is the basic idea of the methodology.

Examples illustrating the applicability and usefulness of the methodology are given elsewhere [1,2,8,10] .

The syntax of the data type is supposed to be given by a set Σ of operations. Its semantics is given (formally or informally) by some other method, for instance, by means of a model.

The methodology consists of the following steps:

1) Elect a canonical form, i.e., a set C of terms such that every element of the data type is uniquely represented in C and whenever $\sigma t_1 \ldots t_n$ is in C then so are $t_1, \ldots, t_n$ σ being an operation in Σ.

2) Translate the given specification into a specification of the operations in terms of the canonical form of 1.

3) For each operation $\sigma \in \Sigma$, write axioms to transform each term of the form $\sigma c_1 \ldots c_n$, where $c_1, \ldots, c_n$ are canonical representatives, into the appropriate canonical representative given by 2.

In many cases we can perform 3 by steps using the following heuristics

3.1) - devise a simpler transformation that "approximates" the desired transformation;

3.2) - write "candidate axioms" to perform the simpler transformation (wich often suggests some candidates);

3.3) - check that these candidate axioms

a) are correct (by using the informal specification or the one given by 2).

b) indeed perform the desired transformation.

This methodology can be formally justified for data types that can be regarded as (many-sorted) algebras in which every element is the value of a variable-free term. Some algebraic terminology will be required for a sketch of the proof. The data type to be specified can be given in several ways of varying degrees of formalization (e.g. set-theoretical models, grammars, graphs, intuitive descriptions, etc.). But any such way amounts to giving an (S,Σ) - algebra A, with sets S of sorts and Σ of operation symbols. An algebraic specification for A consists of a (finite) set of (conditional) equations ε such that $T_{\Sigma,\varepsilon}$ (the quotient of the free algebra T_Σ of terms by the congruence generated by ε) is isomorphic to the algebra A.

The methodology is based on obtaining an algebra of terms C, with $C \cong A$ by construction, and writing ε so as to have $C = T_{\Sigma,\varepsilon}$. The steps previously described can now be rephrased more precisely as follows.

1. Elect a family C of subsets $C_s \subseteq T_{\Sigma,s}$, $s \in S$, such that

   a) for each $\sigma \in \Sigma$ if $\sigma t_1 \ldots t_n \in C$ then $t_1, \ldots, t_n \in C$

   b) the restriction ρ of the unique homomorphism $h: T_\Sigma \to A$ to C gives a bijection $\rho: C \to A$

2. Define for each $\sigma \in \Sigma$ and all $t_1, \ldots, t_n \in C$

   $\sigma_C(t_1, \ldots, t_n) = \rho^{-1}(\sigma_A[\rho(t_1), \ldots, \rho(t_n)])$

3. For each $\sigma \in \Sigma$ write (conditional) equations ϵ_σ such that

   a) C satisfies ϵ_σ

   b) for all $t_1, \ldots, t_n \in C$ $\sigma t_1 \ldots t_n \equiv_{s_\sigma} \sigma_C(t_1, \ldots, t_n)$

The set $\epsilon = \bigcup_{\sigma \in \Sigma} \epsilon_\sigma$ is then a (correct and complete) specification for A i.e. $A \cong T_{\Sigma,\epsilon}$

3. Justification

In order to guarantee that the set $\mathcal{E}$ of all these ε_σ's for $\sigma \in \Sigma$, indeed specify A we must show $A \cong T_{\Sigma,\mathcal{E}}$.

First notice that steps 1(b) and 2 above ensure the Σ-algebra C isomorphic to A.

Now, recall that [3, p. 123; 9] a canonical Σ-term algebra is defined as a Σ-algebra C such that for each $s \in S$ $C_s \subseteq T_{\Sigma,s}$ and whenever $\sigma t_1 \ldots t_n \in C$ then

(i) $t_1, \ldots, t_n \in C$

(ii) $\sigma_C(t_1, \ldots, t_n) = \sigma t_1 \ldots t_n$

To see that the algebra C used in the methodology fits into this definition, notice that condition (i) is guaranteed by step 1(a); whereas (ii) follows from 1(b) and 2.

Finally, theorem 5 of [9] shows that the conditions

(I) C satisfies $\mathcal{E}$

(II) for all $\sigma \in \Sigma$, $t_1, \ldots, t_n \in C$,

$$\sigma t_1 \ldots t_n \equiv_{\mathcal{E}} \sigma_C(t_1, \ldots, t_n)$$

are necessary and sufficient for $C \cong T_{\Sigma,\mathcal{E}}$, but these conditions are guaranteed by 3(a) and 3(b).

Therefore we have altogether $A \cong C \cong T_{\Sigma,\mathcal{E}}$, thereby showing the soundness of the methodology.

Furthermore, the methodology is non-obtrusive, in that following it does not prevent finding $\mathcal{E}$ if it exists at all. In order to see this, assume the existence of a

(finite) set E of (conditional) equations such that $A \cong T_{\Sigma,E}$.

Now, theorem 4 of [9] guarantees the existence of a canonical term algebra $C \cong T_{\Sigma,E}$. For any such C, in view of the necessity of conditions (I) and (II) above, it is possible to perform steps 1(a), 3(a) and 3(b) of the methodology.

It remains to see that we can satisfy the other requirements. To this end let us call h and g, respectively, the unique homomorphisms of T_Σ onto A and C and notice that there exists an isomorphism f: C → A such that h = f.g. Now, given $t \in C$, h(t) = f[g(t)] and by lemma 9A of [3, p. 146] , g(t) = t, so h(t) = f(t). Hence the restriction of h to C gives an isomorphism of C onto A, thereby fulfilling the requirements of steps 1(b) and 2 of the methodology.

4. Conclusions

We have described and justified a methodology for the (correct and complete) specification of an abstract data type given by a model. The justification is based on properties of canonical term algebra [9], which have already been employed to prove the correctness of a given specification [3]. Here these tools are used to obtain a specification.

Some comments on the usefulness of the methodology may be in order. It is flexible enough to allow modular treatment of the various sorts. This is of importance as the crucial step is the election of a "nice" canonical form. The translation of the operations is immediate. The last step consists basically of programming a term-rewriting system [1,5].

The method has been found very helpful also in the formal specification and representation of data base applications [2, 10] .

5. References

[1] A. L. Furtado and P. A. S. Veloso, Procedural specifications and implementations for abstract data types. SIGPLAN Notices 16(3) (1981) 53-62.

[2] A. L. Furtado, P. A. S. Veloso and J. M. V. de Castilho, Verification and testing of S-ER representations, Proc. 2nd Intern. Conf. on Entity-Relationship Approach (1981).

[3] J. A. Goguen, J. W. Thatcher and E. G. Wagner, An initial algebra approach to the specification, correctness and implementation of abstract data types, in: Current Trends in Programming Methodology vol. IV, R. T. Yeh, ed. (Prentice-Hall, Englewood Cliffs, 1978) 80-149.

[4] J. V. Guttag, Abstract data types and the development of data structures, Comm. ACM 20(6) (1977) 306-404.

[5] G. Huet and D. C. Oppen, Equations and rewrite rules: a survey, Technical Report CSL-111, SRI International, Menlo Park, CA (1980)

[6] D. Kapur, Specifications of Majster's traversable stack and Veloso's traversable stack, SIGPLAN Notices 14(5) (1979) 46-53.

[7] T. H. Pequeno and C. J. Lucena, An approach for data type specification and its use in program verification, Inform. Proc. Letters 8(2) (1979) 98-103.

[8] T. H. Pequeno and P. A. S. Veloso, Do not write more axioms than you have to, Proc. Intern. Computing

Symp., vol. 1 (Academia Sinica, Taipei, 1978) 487-498.

[9] J. W.Tatcher , E. G. Wagner and J. B. Wright, Specification of abstract data types using conditional axioms (Extended abstract), Research Report RC2614, IBM Res. Ctr. (Yorktown Heights, 1976).

[10] P. A. S. Veloso, J. M. V. de Castilho and A. L. Furtado, Systematic derivation of complementary specifications, Proc. 7th Very Large Data Bases Conference (1981).