STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXVI, 3 1991

THEORETICAL SUPPORT FOR OBJECT-ORIENTED AND PARALLEL PROGRAMMING

ILIE PARPUCEA[*]

Received: November 13, 1991
AMC Subject Classification: 68Q05

Rezumat. Un model teoretic privind programarea paralelă şi orientată-obiect. Lucrarea prezintă unele aspecte principale ale unui model algebric pentru specificarea unor concepte de bază din limbajele de programare. Legătura cu alte lucrări din domeniul specificării algebrice este prezentată în partea introductivă. Secţiunea 2 prezintă pe scurt conceptul algebric de ierarhie *HAS*. Conceptele de obiect, metoda şi clasa, specifice programării orientate-obiect sînt expuse în secţiunea 3. Secţiunea 4 este destinată conceptelor de algoritm secvenţial, algoritm paralel, proces secvenţial şi proces paralel.

1. **Introduction.** Generally, the specification of a programming language has two purposes. The first purpose is the specification of the data types proper to the language to be specified, which include the primitive (predefined) data types and the composed data types. The second purpose involves the specification of the operations (statements) which act on the data types. The algebraic approach of a language specification constitutes the topics of a great number of papers. We shall mention further down some such works directly related to the present paper.

In [3] and [4] T. Rus presents a hierarchical and algebraic specification model. A context-free algebra is associated to a context-free grammar. The specification aiming at such a model involves a cascade of heterogeneous algebras. The last algebra from the cascade (hierarchy) corresponds to the complete specification of the language. In this model, the hierarchical

[*] *University of Cuj-Napoca, Department of Economics, 3400 Cluj-Napoca, Romania*

order (hierarchy depth) depends on the context-free grammar which specifies the syntax of the language. My paper, which uses an algebraic model [2], starts from the algebraic definition of certain primitive data types (defined in the form of homogeneous algebras). These will be organized into a specification base (signature) for the whole hierarchy of heterogeneous algebras which will constitute the specification levels of the programming language to be specified. This time the hierarchical order (the number of levels of the hierarchy) depends on the complexity of the elements of the language to be specified. The proposed model defines in a personal manner the concepts of object, method and class, proper to the object-oriented programming.

An algebraic approach of a language specification for abstract data types specification can be found in [6]. Unlike this paper, in my paper the concepts of object and method are defined on complexity levels (corresponding to the hierarchical levels), allowing in this way a relatively easy implementation of the model. The final result of a language specification will look like a multi-level tree, unlike [1] where the specification appears as a tree with a single level. At each level the data types and their corresponding operations are defined.

Lastly, on the basis of the algebraic definition of the concepts of parallel algorithm and parallel process [5] implanted on the specification levels of the proposed model, the algebraic specification becomes concurrent algebraic specification.

2. ***HAS* hierarchy concept.** The concept of *HAS* hierarchy was presented in detail by T. Rus in [3] and [4]. This concept is based on the following two principles:

p1. Every homogeneous algebraic structure is a *HAS* of zero hierarchical level in a *HAS* hierarchy;

p2. Every i-level *HAS* can be chosen as a base for an $(i+1)$-level *HAS*.

Let HAS^i be the i-level *HAS* given in the form of the pair:

$$HAS^i = < A^i \; , \; \Omega^i >,$$

where A^i is the support, while Ω^i is the collection of operations defined on the support A^i. The support of the HAS^i is used as index set for the specification of the HAS^{i+1}. Consider the n-ary operation $\omega\epsilon\Omega^i$, and $a = \omega\ (a_1,a_2,\ldots,a_n)$, where $a,a_1,a_2,\ldots,a_n \epsilon$ $\epsilon\ A^i$. The set of operation schemes Σ_ω is defined as follows:

$$\Sigma_\omega = \{\sigma = <n,\omega,a_1a_2\ldots a_na> / \ a=\omega(a_1,a_2,\ldots,a_n> \}.$$

Since the operations specified by means of the operation schemes σ are heterogeneous operations, we consider the symbol of the operation ω to be distributed upon its operands. In other words, the operation scheme σ becomes $\sigma=<n,s_0s_1\ldots s_n,a_1a_2\ldots a_na>$. For a complete specification of the HAS^{i+1} by the HAS^i , consider a function F which associates to each operation scheme $\sigma\epsilon\Sigma_\omega$, $\omega\epsilon\Omega^i$, a heterogeneous operation specific to HAS^{i+1}. If $\sigma=<n,s_0\ldots s_n$, $a_1\ldots a_na>$, then F_σ is a specific operation in HAS_{i+1}, that is , the function:

$$F_\sigma \ : \ A^i_{a_1}\times A^i_{a_2}\times\ldots\times A^i_{a_n} \dashrightarrow A^i_a \ .$$

In these conditions HAS^{i+1} is specified on the basis of HAS^i

and has the form:

$$HAS^{i+1} = \langle A^{i+1}=(A_a^{i+1})_{a\in A}i, \Sigma=(\Sigma_\omega)_{\omega\in\Omega}i, F\rangle.$$

On the basis of the concept of HAS hierarchy, in Section 3 we shall present schematically a model for the specification of a programming language. The terms of object, method, and class, specific to the object-oriented languages, are found again in our model. Section 4 presents briefly the algebraic formalization of the concepts of parallel algorithm and parallel process [5] adapted to our model.

3. Object-oriented hierarchical specification. An abstract object is defined as heterogeneus algebra as follows:

Object name	: NAME;
Supports	: $\text{NAME}_1, \text{NAME}_2, \ldots, \text{NAME}_n$;
Operation schemes	: $\sigma_1, \sigma_2, \ldots, \sigma_m$;
Variables	: LIST_1 : $\text{NAME}i_1$
	LIST_2 : $\text{NAME}i_2$
	....
	LIST_k : $\text{NAME}i_k$
Axioms	: $w_{11}=w_{12}$.
	$w_{21}=w_{22}$,
	......
	$w_{p1}=w_{p2}$;

end NAME.

We choose as zero level of the *HAS* hierarchy (HAS^0) the following partial homogeneous algebra:

$HAS^0 = <A^0, \Omega^0, H^0 : A^0 \rightarrow I>$,

where:

A^0 = support of the algebra, consisting of a set of predefined abstract objects $(A_j^0)_{j \in J}$ specified as homogeneous algebras;

Ω^0 = the set of operations defined on the objects (A_j^0) , $j \in J$;

H^0 = function which associates a measure to every $a \in A^0$;

I = the set of the measures printed out by the function H^0.

The level one of the *HAS* hierarchy (HAS^1) is defined on the basis of HAS^0 and has the form:

$$HAS^1 = < A^1 = (A_i^1)_{i \in I} , \Sigma = (\Sigma_\omega)_{\omega \in \Omega^0}, H^1 : A^1 \dashrightarrow I, F^1 > ,$$

where:

A^1 = a family of subsets with the property that the same measure $H^0(a)$, $a \in A_i^1$, $i \in I$, is associated to all elements of such a subset;

Σ = the set of operation schemes specified by the operations corresponding to the zero level;

H^1 = the same definition as in the case of H^0;

F^1 = symbol of a function $\left(F^1 : (\Sigma_\omega)_{\omega \in \Omega^0} \dashrightarrow OP(A^1)\right)$, where $OP(a^1)$ is the set of all operations defined on A^1.

For $\omega \in \Omega^0$, $a_i \in A^0$, $i=1,2,\ldots,n$, $a=\omega(a_1,a_2,\ldots,a_n)$, an operation scheme

$$\sigma=<n,s_0s_1\ldots s_n,H^0(a_1)H^0(a_2)\ldots H^0(a_n)H^0(a)>$$

is associated.

If $\sigma\in\Sigma$ is an operation scheme, then F_σ^1 , , defined as follows:

$$F_\sigma^1 : A^1_{H^0(a_1)}xA^1_{H^0(a_2)}x\ldots xA^1_{H^0(a_n)}\rightarrow A^1_{H^0(a)}$$

is a heterogeneous operation in HAS^1.

The function H^0 associates to every $a\in A^1$ a characteristic called measure. Another characteristic associated to an $a\in A^1$ is the interpretation mode. For a given measure $H^0(a)$ associated to an element $a\in A^1$ there can exist several interpretation modes (integer, real, boolean, character, etc.). Hence the interpretation mode for an $a\in A^1$ is the significance (integer, real, boolean, character, etc.) assigned to the representation (encoding) of a. We particularize the function H^0 as follows: for every $a\in A^1$, $H^0(a)=1(a)$, where $1(a)$ is the representation length, representing a in the computer storage. The conection between the interpretation mode and the measure (representation length) of the unstructured type data can be performed by a bi-dimensional matrix. One considers $a_{ij}=1$ if for the representation length i there exists the interpretation mode j, and $a_{ij}=0$ in the opposite case. The row index of the matrix ($i=1,2,\ldots,n$) signifies the possible representation length of the data from A^1, while the column index ($j=1,2,\ldots,m$) is the index associated to the elements of the set of the interpretation modes. Having these elements, we define the level two of the HAS hierarchy (HAS^2) on the basis of the preceding level:

$$HAS^2=\langle A^2=(A^2_{ij})_{i\in N,j\in N},\Sigma=(\Sigma_\omega)_{\omega\in OP(A^1)},H^2:A^1xM\to NxM,F^2\rangle$$

where:

A^2_{ij} = the support of the data type i interpreted in the mode j;

$\Sigma\omega$ = the set of the operation schemes generated by the operation $\omega\epsilon OP(A^1)$;

N = a set containing all possible representation length;

M = a set containing all possible interpretation modes;

H^2 = function which establishes by its values the index set for the family of sets A^2;

$\forall\ (a,j)\epsilon A^1xM,H^2(a,j)=(H^1(a),j)a_{H^1(a)j}$

F^2 = the symbol of a function which associates to an operation scheme

$\sigma=<n,s_0s_1\ldots s_n,H^2(a_1,j_1)H^2(a_2,j_2)\ldots H^2(a_n,j_n)H^2(b,j_{n+1})>$

a heterogeneous operation on the data set A^2.

The zero level HAS^0 points out the primitive (predefined) data types together with the operations defined on them. The level one HAS^1 points out the data division in subsets, the criterion of differentiation being the representation length. The level two HAS^2 differentiates the data according to a characteristic supplementary to the preceding level, that is, the interpretation mode. This allows the specification of data types (short integer, long integer, real on different length, character, etc.). The composed data types are specified at this level, too.

Let us denote by $OP(HAS^i)_{i=0,1,2}$ the set of the heterogeneous operations defined on the three hierarchical levels. An equivalence relation, denoted HAS^e, is defined on this set. Two operations ω_1, ω_2 $\in (HAS^i)_{i=0,1,2}$ are, by definition, called equivalent if they have the same definition domain. Consider ω_1: $A_1 x A_2 x \ldots A_n \to A_{n+1}, \omega_2 : B_1 x B_2 x \ldots x B_n \to B_{n+1}$, $\omega_1 HAS^e \omega_2$, where

$$A_i = \begin{cases} A_i^0 \,, & \text{if } \omega_i \in \Omega^0 \\ A^1_{H^0(a_i)} \,, & \text{if } \omega_1 = F^1_\sigma, \sigma \in \sum, \sigma = \langle n, s_0 s_1 \ldots s_n, \\ & \quad H^0(a_1) \ldots H^0(a_n) H^0(a) \rangle \\ A^2_{H^2(a_i, j_i)}, & \text{if } \omega_1 = F^2_\sigma, \ \sigma \in \sum, \sigma = \langle n, s_0 s_1 \ldots s_n, \\ & \quad H^2(a_1, j_1) \ldots H^2(a_n, j_n) \ H^2(a, j_{n+1}) \rangle \end{cases}$$

$i=1,2,\ldots n+1$, and B_i is obtained in a similar manner. Follows that $A_1=B_1, A_2=B_2, \ldots, A_n=B_n$ (equality of sets). Let $E =$ $= OP(HAS^i)_{i=0,1,2}/HAS^e$ be the set of the equivalence classes. An equivalence class $e \in E$ consists of all operations with the same definition domain. Let $C_1 x C_2 x \ldots x C_n$ be the common definition domain for the operations belonging to the equivalence class e. We call object an n-uple $(c_1, c_2, \ldots, c_n) \in C_1 x C_2 x \ldots x C_n$, while the set of all objects of this form will be called the class of objects associated to the equivalence class e. Let us denote by K the set of all classes of objects. We add to each class of objects $k \in K$ the object nil$_k$ which constitutes the nil object of the class k. An object of the class k is either the object nil$_k$ or an instance of the class k. The specification of a class k consists firstly of the specification of the form of the objects belonging to the class k. The form of a class k specifies the n-

uples which can appear as values of the instances of the objects belonging to the class *k*. On the other hand, the specification of the class *k* consists of the specification of a colection of methods belonging to the class *k*, too. An operation *wee* acts on the class of objects *k* according to a law well defined during the stage of hierarchical level construction. Such an operation on a class of objects *k* will be called method. The set of methods is classified on hierarchical levels, but there also exists hybrid methods whose definition domains originate in several hierarchical levels.

The number of levels in the *HAS* hierarchy is arbitrary. It depends on the needs of defining certain objects of high complexity degree. We stopped at the level two of the hierarchy, considering it to be sufficient for a concise exposition of the model.

4. Concurrent hierarchical specification. Let *<A,OP>* a homogeneous algebra, where *A* is the support and *OP* is the set of the operations defined on *A*. If the set *OP* also contains relations, then *<A,OP>* will be called algebraic system. The concept of heterogeneous algebraic system is obtained analogously.

Let us consider the heterogeneous algebra *HA=<KL,ME>*, where *KL* is the set of the classes of objects, while *ME* is the set of the methods specified in Section 3. We define the concept of algorithm over the given heterogeneous algebra HA as being the heterogeneous algebraic system *AL=<K*1*,Me,R>*, where:

$K1$ = a finite set of classes belonging to the support HA;

Me = a finite set of methods belonging to the set Me;

R = a relation indicating the order of execution of the methods from Me on the objects of $K1$.

If the ordering relation R is linear (or total) on Me, then the algorithm is called sequential algorithm.

If the ordering relation R is partial on Me, then the algorithm is called parallel (or concurrent) algorithm.

Since the set Me of the methods specifying an algorithm is finite, this one can always be decomposed into the subsets Me_1, $Me_2,\dots,Me_k$, such that every Me_i, $i=1,2,\dots,k$, is linearly ordered by the execution of the methods. Let us denote by R_i the linear relation defined by the order of execution of the methods in Me_i. If for every $i,j,i\neq j,i,j=1,2,\dots,k$, the subset $K1_i\subseteq K1$ on which the methods from Me_i are acting and the subset $K1_j\subseteq K1$ on which the methods from Me_j are acting are disjoint each other, then the algorithm $<K1,Me,R>$ gives rise to a family of sequential algorithms $<K1_i,Me_i,R_i>$, $i=1,2,\dots,k$. These sequential algorithms can be parallelly executed and keep the consistence of the computations specified by the original algorithm.

In this context a processor is identified with an abstract agent able to execute any method featuring the HA specification. The concept of process over the HA specification is defined through the couple Process (HA)=<Processor,AL>.

A process P=<Processor,AL> will be called sequential if the defined algorithm AL is sequential.

A process P=<Processor,AL> will be called parallel (or concurrent) if the defined algorithm *AL* is a parallel algorithm.

5. **Conclusions**. The basic theoretical concepts (belonging to the programming languages) specified by means of the proposed model constitutes a theoretical nucleus for the simultaneous approach of parallel programming and object-oriented programming. The nucleus, semantically and syntactically defined, could constitute a reference basis for any other semantic construction reductible to one of the semantic forms from the nucleus by established transformation rules.

R E F E R E N C E S

1. Bergstra,J.A., Heering,J., Klop, J.W., *Object-Oriented Algebraic Specification: Proposal for a Notation and 12 Examples*, Report CS-R8411, June 1984, Centre for Mathematics and Computer Science.
2. Parpucea,I., *Dynamic Extensions of Programming Language Semantics*, Proceedings of the First International Conference "Algebraic Methodology and Software Technology", May, 22-24, 1989, Iowa, U.S.A.
3. Rus,T., *HAS-Hierarchy: A Natural Tool for Language Specification*, Ann. Soc. Math., Polonae, Series IV: Fundamenta Informaticae, 3.
4. Rus T., (in Romanian) *Mecanisme Formale pentru Specificarea Limbajelor (English translation of the title Formal Tools for Language Specification)*, Ed.Acad.R.S.R.1983, Romania.
5. Rus,T., *Language Support for Parallel Programming*, private comunication.
6. Wagner, E.G. *An Algebralically Specified Language for Data Directed Design*, Proceedings of the First International Conference, "Algebraic Methodology and Software Technology", May, 22-24, 1989, Iowa, U.S.A.