ALGEBRAIC SPECIFICATION OF PROGRAMMING LANGUAGE SUBSETS

Ilie PARPUCEA*

Received: March 1, 1993
AMS Subject Classification: 68Q05

REZUMAT. Specificarea algebrică a subseturilor unui limbaj de programare. În această lucrare autorul încearcă aplicarea unui model algebric de specificare, în definirea subseturilor unui limbaj de programare. Modelul se bazează pe ierarhia algebrelor eterogene. Cîteva rezultate teoretice cît şi un exemplu concret de specificare sînt redate amănunţit pe parcursul întregii lucrări.

1. Introduction. The algebraic modelling of programming language specification is subjected to an intense research, with significant results. Partial or "relatively total" solutions were found, having the category theory [ADJ73, ADJ77], partial or total heterogeneous universal algebras [BRO82, BRO87], algebras with operator schemes, or even context-free algebras [RUS72] as algebraic departure point. These solutions concerned many times only certain aspects of the programming language specification.

The ADJ group of authors attempts to develop a concept of programming language within the framework of the category theory. It is interesting as to the mathematical object in its own self, constituting a strong source of inspiration for subsequent results. In [WAG89] the author presents a model of algebraic specification of a language for abstract data type specification. This model of language belongs to the object-oriented language set.

The total or partial heterogeneous algebras seem to be ever

* *Universitatea "Babeş-Bolyai" Cluj-Napoca*
Facultatea de Ştiinţe Economice
P-ţa Ştefan cel Mare 1, 3400 Cluj-Napoca, România

more used to the complex approach of programming language specification. Lots of papers (e.g. [BRO82, BRO87]) constitute interesting approaches, from both theoretical and practical viewpoints. They present algebraic models for defining abstract types of partial data and hierarchical data types. Very interesting is the pure algebraic model concerning the specification of programming language semantics, using particular models of abstract data types [BRO87].

Both the heterogeneous algebras [BIR70] and those with scheme of operators [HIG63] constituted the main theoretical source for the elaboration of HAS hierarchy [RUS80] and introduction of context-free algebras [RUS72]. It is to be noticed that the HAS hierarchy concept allows a unitary approach of the programming language specification, from both semantic and syntactical viewpoint.

The specification of a programming language implies the existence of a set of data (which constitutes the data base of the language), and a set of operation defined on this data base (which constitutes the instruction set). The data base and the instruction set associated to a language constitute the so-called "computing reality". The concept of program will allow the identification of that subset from the "computing reality" which needs transformations. Within the framework of a specified programming language, a program will constitute a well defined entity from two points of view: semantic and syntactical.

We shall consider a language which gathers together all "computing realities" corresponding to a given set of languages.

This language will be called universal language (UL). Its restriction to a certain "computin reality" will constitute a subset of the universal language. Such an approach allow to consider the programming languages hierarchically as to their specification and implementation.

2. Example of simple language. In order to exemplify our model, we shall consider the following simple language. This language includes three data types: numerical, array and record. On the numerical type there are defined the arithmetic operations (+,-,*,/,**) and the relational operations (<,<=,>,>=,==,/=). One defines the instruction IF and the assignment instruction. The syntax of data types and arithmetic expressions is given in BNF by:

```
<TYPE_DECLARATION>::=type <TYPE_NUME> is <TYPE_DEFINITION>
<TYPE_DEFINITION>::=<NUMERICAL_TYPE>/<ARRAY_TYPE>/<RECORD_TYPE>
<CONSTRAINT>::= range <RANGE>
<RANGE>::=<SIMPLE_EXPRESSION>..<SIMPLE_EXPRESSION>
<SIMPLE_EXPRESSION>::=<FACTOR>{<ARITHMETICAL_OPERATOR><FACTOR>}
<FACTOR>::=<VARIABLE>/<CONSTANT>
<VARIABLE>::=<IDENTIFIER>/<INDEXED_COMPONENT>/
             <SELECTED_COMPONENT>
<RELATION>::=<SIMPLE_EXPRESSION><RELATIONAL_OPERATOR>
             <SIMPLE_EXPRESSION>
<ARITHMETICAL_OPERATOR>::=+/-/*///**
<RELATIONAL_OPERATOR>::=</<=/>/>=/==//=
<SUBTYPE_DEFINITION>::=subtype<SUBTYPE_NAME>
```

```
                          is<TYPE_DESIGNER>[<CONSTRAINT>]
<TYPE_DESIGNER>::=<TYPE_NAME>/<SUBTYPE_NAME>/
                      <PREDEFINITION_TYPE>
<NUMERICAL_TYPE>::=<CONSTRAINT>/new<PREDEFINITION_TYPE>
                     <CONSTRAINT>
<PREDEFINITION_TYPE>::=<INTEGER>/<FLOAT>
<OBJECT_DECLARATION>::=<IDENTIFIER_LIST>:<TYPE_DESIGNER>
                      [:=<SIMPLE_EXPRESSION>];
<IDENTIFIER_LIST::=<IDENTIFIER>{,<IDENTIFIER>}
<RECORD_TYPE>::=record<COMPONENT_LIST> end record.
<COMPONENT_LIST>::=<OBJECT_DECLARATION>{,<OBJECT_DECLARATION>}
<ARRAY_TYPE>::=array(<INDEX>{,<INDEX>}) of <TYPE_DESIGNER>
<INDEX>::=<RANGE>/<INTEGER>
```

The syntax of assignment and IF instructions is given also in BNF by:

```
<ASSIGNMENT_STATEMENT>::=<VARIABLE>:=<SIMPLE_EXPRESSION>
<EL_IF>::=<RELATION> then <SEQUENCE>
<IF_LIST>::=<EL_IF>/<IF_LIST>elseif<EL_IF>
<IF>::=if<IF_LIST>endif/if<IF_LIST>else<SEQUENCE>endif
<SEQUENCE>::=<ASSIGNMENT_STATEMENT>{;<ASSIGNMENT_STATEMENT>}
```

Remark. The symbols +, -, *, /, ** correspond to the arithmetic operations of addition, substraction, multiplication, division, powering, while the symbols <, <=, >, >=, ==, /= correspond to the relational operations of smaller than, smaller than or equal to, greater than, greater than or equal to, equal to, different from, respectively.

In the above defined language the numerical types <INTEGER>

and <FLOAT> are considered to be predefined. We renounced the definition of the following syntactical categories: <INDEXED-COMPONENT>, <SELECTED-COMPONENT>, <IDENTIFIER>, <CONSTANT>, <TYPE-NAME> and <SUBTYPE-NAME>. Neither the theoretical aspect, nor the examples we shall present will be altered by these restrictions.

A context-free grammar is considered to be a quadruple $\underline{G} = \langle V_n \cup V_t , V_t , P , V_n \rangle$, where

$$P = \{ A \rightarrow s_0 A_1 s_1 A_2 s_2 \ldots s_{n-1} A_n s_n \mid s_0, s_1, \ldots, s_n \in V_t^*, A, A_1, A_2, \ldots, A_n \in V_n \}$$

V_n is the nonterminal symbol set, while V_t is the terminal symbol set. A specification base $\underline{B} = \langle V_n \cup V_t , \Sigma S \rangle$ is made to correspond to this context-free grammar, where

$$\Sigma S = \{ \delta = \langle n, s_0 s_1 \ldots s_n, A_1 A_2 \ldots A_n A \rangle \mid A \rightarrow s_0 A_1 s_1 \ldots s_{n-1} A_n s_n \in P \}.$$

The context-free algebra [RUS83] specified by $\underline{B}$ can be written in the form of the triple

$$\underline{A} = \langle (W)_{A \in V_n} = W_A(V_t, \Sigma S)_{A \in V_n}, \Sigma S_{\underline{B}}, F \rangle,$$

where $W_A(V_t, \Sigma S) = \{x \in V_t^* \mid A \overset{*}{\rightarrow} x\}$, and for every $w_k \in W_{A_k}(V_t, \Sigma S)$,

$k=1,2,\ldots,n$, and $\delta = \langle n, s_0 s_1 \ldots s_n, A_1 A_2 \ldots A_n A \rangle$ we have $F_\delta(w_1, w_2, \ldots, w_n) = s_0 w_1 s_1 \ldots s_{n-1} w_n s_n \in W_A(V_t, \Sigma S)$.

F is the symbol of a function which associates to every operation scheme $\delta \in \Sigma S$ a heterogeneous operation specific to the context-free algebra $\underline{A}$. If $\delta = \langle n, s_0 s_1 \ldots s_n, A_1 A_2 \ldots A_n A \rangle$, then F_δ is the function

$$F_\delta : W_{A_1} \times W_{A_2} \times \ldots \times W_{A_n} \rightarrow W_A.$$

The action law of the function is the above mentioned one. We present further down the operation schemes $\delta \in \Sigma S$:

σ_1 = <2, type is,TYPE_NAME TYPE_DEFINITION TYPE_DECLARATION>

σ_2 = <1, ,NUMERICAL_TYPE TYPE_DEFINITION>

σ_3 = <1, ,ARRAY_TYPE TYPE_DEFINITION>

σ_4 = <1, ,RECORD_TYPE TYPE_DEFINITION>

σ_5 = <1, range ,RANGE CONSTANT>

σ_6 = <2, .. ,SIMPLE_EXPRESSION SIMPLE_EXPRESSION RANGE>

σ_7 = <1, ,FACTOR SIMPLE_EXPRESSION>

σ_8 = <3, ,FACTOR ARITHMETICAL_OPERATOR SIMPLE_EXPRESSION SIMPLE_EXPRESSION>

σ_9 = <1, ,VARIABLE FACTOR>

σ_{10}= <1, ,CONSTANT FACTOR>

σ_{11}= <1, ,IDENTIFIER VARIABLE>

σ_{12}= <1, ,INDEXED_COMPONENT VARIABLE>

σ_{13}= <1, ,SELECTED_COMPONENT VARIABLE>

σ_{14}= <3, ,SIMPLE_EXPRESSION RELATIONAL_OPERATOR SIMPLE_EXPRESSION RELATION>

σ_{15}= <0, + ,ARITHMETICAL_OPERATOR>

σ_{16}= <0, - ,ARITHMETICAL_OPERATOR>

σ_{17}= <0, * ,ARITHMETICAL_OPERATOR>

σ_{18}= <0, / ,ARITHMETICAL_OPERATOR>

σ_{19}= <0, ** ,ARITHMETICAL_OPERATOR>

σ_{20}= <0, < ,RELATIONAL_OPERATOR>

σ_{21}= <0, <= ,RELATIONAL_OPERATOR>

σ_{22}= <0, > ,RELATIONAL_OPERATOR>

σ_{23}= <0, >= ,RELATIONAL_OPERATOR>

σ_{24}= <0, == ,RELATIONAL_OPERATOR>

σ_{25}= <0, /= ,RELATIONAL_OPERATOR>

σ_{26}= <2,subtype is ,SUBTYPE_NAME TYPE_DESIGNER SUBTYPE_DEFINITION>

σ_{27}= <3,subtype is ,SUBTYPE_NAME TYPE_DESIGNER CONSTRAINT SUBTYPE_DEFINITION>

σ_{28}= <1, ,TYPE_NAME TYPE_DESIGNER>

σ_{29}= <1, ,SUBTYPE_NAME TYPE_DESIGNER>

σ_{30}= <1, ,PREDEFINITION_TYPE TYPE_DESIGNER>

σ_{31}= <1, ,CONSTRAINT NUMERICAL_TYPE>

σ_{32}= <2, new ,PREDEFINITION_TYPE CONSTRAINT NUMERICAL_TYPE>

σ_{33}= <1, ,INTEGER PREDEFINITION_TYPE>

σ_{34}= <1, ,FLOAT PREDEFINITION_TYPE>

σ_{35}= <2, : ,IDENTIFIER_LIST TYPE_DESIGNER OBJECT_DECLARATION>

σ_{36}= <3, : := ,IDENTIFIER_LIST TYPE_DESIGNER SIMPLE_EXPRESSION OBJECT_DECLARATION>

σ_{37}= <1, ,IDENTIFIER IDENTIFIER_LIST>

σ_{38}= <2, , ,IDENTIFIER_LIST IDENTIFIER IDENTIFIER_LIST>

σ_{39}= <1,record end record, COMPONENT_LIST RECORD_TYPE>

σ_{40}= <1, ,OBJECT_DECLARATION COMPONENT_LIST>

σ_{41}= <2, , ,COMPONENT_LIST OBJECT_DECLARATION COMPONENT_LIST>

σ_{42}= <1, ,INDEX INDEX_LIST>

σ_{43}= <2, , ,INDEX_LIST INDEX INDEX_LIST>

σ_{44}= <2,array( ) of ,INDEX_LIST TYPE_DESINGER ARRAY_TYPE>

σ_{45}= <1, ,RANGE INDEX>

σ_{46}= <1, ,INTEGER INDEX>

σ_{47}= <2, := ,VARIABLE SIMPLE_EXPRESSION ASSIGNMENT_STATEMENT>

σ_{48}= <2, then ,RELATION SEQUENCE EL_IF>

σ_{49}= <1, ,EL_IF IF_LIST>

σ_{50}= <2, elsif ,IF_LIST EL_IF IF_LIST>

σ_{51}= <1,if endif,IF_LIST IF>

σ_{52}= <2, if else endif,IF_LIST SEQUENCE IF>

σ_{53}= <1, ,ASSIGNMENT_STATEMENT ASSIGN_LIST>

σ_{54}= <2, , ,ASSIGN_LIST ASSIGNMENT_STATEMENT ASSIGN_LIST>

σ_{55}= <1, ,ASSIGN_LIST SEQUENCE>

3. Basic concepts. Let $\underline{B} = \langle I \cup S, \Sigma S, \alpha \rangle$ be the specification base for an arbitrary programming language. $\underline{B}$ is organized under the form of a homogeneous algebra. I is the set of all object names, S is the reserved word set, such that $S \cap I = \emptyset$; ΣS is the set of operation schemes represented as triples: $\Sigma S = \{\sigma = \langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle\}$; α is the set of axioms given as couples of words (w_1, w_2). These words are generated by the operations specified by the operation schemes $\sigma \in \Sigma S$. For an operation scheme $\sigma \in \Sigma S$, $\sigma = \langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$, the function

$$F_\sigma : A_{i_1} X A_{i_2} X \ldots X A_{i_n} \to A_i$$

specifies a signature for a heterogeneous operation in the language specified by $\underline{B}$. Consider an arbitrary subset $J \subseteq I$, to which we shall refer in what follows.

DEFINITION 1. *The operation scheme* $\sigma' = \langle n, s_0 s_1 \ldots s_n,$

$j_1 j_2 \ldots j_n i>$, $j_k \in J$, $k=1,2,\ldots,n$, *is called restriction of the operation scheme* $\sigma\in\Sigma S$, $\sigma=<n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i>$ *to the subset* J *if there exists the function* $F_{\sigma'} : A_{j_1} X A_{j_2} X \ldots X A_{j_n} \to A_i$ *which is a restriction of the function* $F_\sigma : A_{i_1} X A_{i_2} X \ldots X A_{i_n} \to A_i$.

For a given subset J, an operation scheme σ can have one or several restrictions. We denote by $\sigma|_J$ the set of all restrictions of σ to J, namely:

$$\sigma|_J = \{ \sigma' | \sigma' \text{ is a restriction of the operation scheme } \sigma \text{ to the subset } J \}.$$

Remark. If $\sigma'=<n, s_0 s_1 \ldots s_n, j_1 j_2 \ldots j_n i>$ is a restriction of the operation scheme $\sigma=<n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i>$ to the subset J, then $A_{j_k} \subseteq A_{i_k}$, $\forall_{j_k}$, $k=1,2,\ldots,n$.

DEFINITION 2. *Let* $\Sigma'=\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ *be a set of operation schemes. A restriction of* Σ' *to the subset* J *is defined as follows:*

$$\sum{}'|_J = \{\sigma_i' | \sigma_i' \in \sigma_i|_J \,,\ i=\overline{1,m}\}$$

DEFINITION 3. *The restriction of the specification base* $\underline{B} = <I \cup S\, \Sigma S, \alpha>$ *to the subset* J *is the specification base* $\underline{B}|_J = <J \cup S, \Sigma S|_J, \alpha|_J>$, *where* $\alpha|_J$ *is that subset of the axiom set* α *which consists of the axioms satisfied by the operations specified by* $\Sigma S|_J$ *(restriction of the operation scheme set* ΣS *to the subset* J*).*

DEFINITION 4. *An interpretation* [RUS83] *of the specification base* $\underline{B} = <I \cup S, \Sigma S, \alpha>$ *is the triple* $<A, \rho, \Psi>$, *where* $A = (A_i)_{i\in I}$ *is a family of sets indexed by the set* I *of the supports of* $\underline{B}$; φ

is a function $\varphi : I \to A$ *such that* $\varphi(i)$ *is an element of A for every* $i \in I$; Ψ *is the function* $\Psi : \Sigma S \to OP(A)$, *where* $OP(A)$ *is the set of all operations defined on A.*

The interpretation $\underline{A} = \langle A, \varphi, \psi \rangle$ of the specification base $\underline{B} = \langle I \cup S, \Sigma S, \alpha \rangle$ fulfils the following conditions:

C1. $\varphi(i) = A_i$ for every $i \in I$; A_i is called the support set for the name of the object i.

C2. For every $\sigma \in \Sigma S$, $\sigma = \langle n, s_0, s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$, $\Psi(\sigma): \varphi(i_1) \times \varphi(i_2) \times \ldots \times \varphi(i_n) \to \varphi(i)$, or in other words $\Psi(\sigma): A_{i_1} \times A_{i_2} \times \ldots \times A_{i_n} \to A_i$, such that if $\sigma = \langle 0, s, i \rangle$ then $\Psi(\sigma): \phi \to \varphi(i)$ or $\Psi(\sigma): \phi \to A_i$, namely $\Psi(\sigma)$ is in this case the injection of the element $\Psi(\sigma)$, which will be denoted by s, in the set A_i.

C3. Every axiom (W_1, W_2) is considered to be satisfied in the interpretation $\underline{A} = \langle A, \varphi, \psi \rangle$; this means that it is satisfied by operations from $OP(A)$.

We shall hereafter denote by $\underline{I}(\underline{B})$ the set of interpretations of a specification base $\underline{B} = \langle I \cup S, \Sigma S, \alpha \rangle$.
If $\underline{A} \in \underline{I}(\underline{B})$, then it can be denoted $\underline{A} = \langle A = (A_i)_{i \in I}, \Sigma S, \psi \rangle$, where $\varphi(i) = A_i$ for every $i \in I$, and

$\psi(\sigma): \varphi(i_1) \times \varphi(i_2) \times \ldots \times \varphi(i_n) \to \varphi(i)$ or $\psi(\sigma): A_{i_1} \times A_{i_2} \times \ldots \times A_{i_n} \to$

for every $\sigma \in \Sigma S$, $\sigma = \langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$. In other words, every interpretation $\underline{A}$ of the specification base $\underline{B}$ is a heterogeneous algebra.

DEFINITION 5. *Let* $\underline{B}$ *=<I∪S, ΣS, α> be a specification base, and let* $\underline{A} = \langle A, \varphi, \psi \rangle$ *be an interpretation; then* $\underline{A}|_J = \langle A|_J, \varphi|_J, \psi|_J \rangle$ *is called restriction of the interpretation* $\underline{A} \in \underline{I}(\underline{B})$ *to the subset J, where:* $A|_J$ *is the family of sets* $(A_j)_{j \in J}$*;* $\varphi|_J$ *is the restriction of the function* φ *to the subset J;* $\Psi|_J$ *is the restriction of the function* Ψ *to the operation scheme set* $\Sigma S|_J$.

PROPERTY 1. *Let* $\underline{B}$ *=<I ∪ S, ΣS, α> be a specification base, and let* $\underline{A} \in \underline{I}(\underline{B})$ *an arbitrary interpretation . Then the restriction of* $\underline{A}$ *to the subset J,* $\underline{A}|_J = \langle A|_J, \varphi|_J, \psi|_J \rangle$ *, is an interpretation of the restriction of* $\underline{B}$ *to the subset J,* $\underline{B}|_J = \langle J \cup S, \Sigma S|_J, \alpha|_J \rangle$.

Proof. The proof results immediately from Definitions 3,4 and 5.

For a given specification base $\underline{B}$ =<*I* ∪ *S*, *ΣS*, *α*>, an important interpretation is the so-called word-algebra or the heterogeneous *W*-algebra [RUS83] specified by the base $\underline{B}$. This is obtained as follows:

(i) One constructs the family of all words specified by the base $\underline{B}$ in the form $W=(W_i)_{i \in I}$, where every W_i is defined as follows:

r1. If $\sigma=\langle 0,s,i \rangle, \sigma \in \Sigma S$, then $s \in W_i$.

r2. If $\sigma \in \Sigma S$, $\sigma=\langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$, and if $w_1, w_2, \ldots, w_n$ are words of the types $i_1, i_2, \ldots, i_n$, respectively, namely $(w_1, w_2, \ldots, w_n) \in W_{i1} x W_{i2} x \ldots x W_{in}$, then

$w = s_0 w_1 s_1, \ldots, s_{n-1} w_n s_n \in W_i$.

r3. The rules r1 and r2 describe all the words of the type i, which will sometimes be denoted by $W(i)$, too.

(ii) For the given specification base $\underline{B}$ one constructs the following interpretation:

p1. For every $i \in I$, $\varphi(i) = W(i)$.

p2. For every $\sigma \in \Sigma S$, $\sigma = \langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$, the function $\Psi(\sigma): W(i_1) x W(i_2) x \ldots x W(i_n) \to W(i)$ acts according to the law: if $w_k \in W(i_k)$, $k = 1, 2, \ldots, n$, then $\Psi(\sigma)(w_1, w_2, \ldots, w_n) = s_0 w_1 s_1 \ldots s_n w_n \in W(i)$.

(iii) Interpreting the specification axioms as formal identities [RUS83], follows that the family $W = (W(i))_{i \in I}$ together with the above defined functions (φ, ψ) form an interpretation of the considered specification base. This interpretation is written under the form

$$\underline{W}(\underline{B}) = \langle W = (W(i))_{i \in I}, \Sigma S, \psi \rangle$$

Let $\underline{A}_1 = \langle A = (A_i)_{i \in I}, \Sigma S, \Psi_1 \rangle$, $\underline{A}_2 = \langle B = (B_i)_{i \in I}, \Sigma S, \Psi_2 \rangle$ be two similar heterogeneous algebras such that $\underline{A}_1, \underline{A}_2 \in \underline{I}(\underline{B})$.

DEFINITION 6. *A family of functions $f = (f_i)_{i \in I}$, $f_i : A_i \to B_i$, indexed by the set I is called similarity morphism or homomorphism from $\underline{A}_1$ to $\underline{A}_2$ if for every operation scheme $\sigma \in \Sigma S$, $\sigma = \langle n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i \rangle$ and for every*

$$(a_1, a_2, \ldots, a_n) \in A_{i_1} x A_{i_2} x \ldots x A_{i_n}$$

we have

$$f_i(\psi_1(\sigma)(a_1, a_2, \ldots, a_n)) = \psi_2(\sigma)(f_{i_1}(a_1), f_{i_2}(a_2), \ldots, f_{i_n}(a_n)).$$

DEFINITION 7. *Let $\underline{B} = \langle I \cup S, \Sigma S, \alpha \rangle$ be a given specification base. A model of $\underline{B}$ is an interpretation $\underline{W} \in \underline{I}(\underline{B})$ with the* *property that for every other interpretation $\underline{A} \in \underline{I}(\underline{B})$ there exists exactly one homomorphism $f:\underline{W} \to \underline{A}$.*

In [RUS83] there is shown that for every specification base $\underline{B} = \langle I\cup S,\ \Sigma S,\ \alpha\rangle$ there exists a model of this base. This model is the word algebra $\underline{W}\ (\underline{B}) = \langle W=(Wi))_{i\in I},\ \Sigma S,\ \Psi\rangle$. If $\underline{W}_1(\underline{B})$ and $\underline{W}_2(\underline{B})$ are two models of the base $\underline{B}$, then they are isomorphic.

Since the model $\underline{W}(\underline{B})$ belongs to the set of interpretations of $\underline{B}$, according to Definition 5 one can consider the restriction of the model $\underline{W}(\underline{B})$ to the subset J, written as

$$\underline{W}(\underline{B})|_J = \langle W|_J = (W(j))_{j\in J},\ \Sigma S|_J,\ \Psi|_J\rangle .$$

Property-1 also holds for the model $\underline{W}(\underline{B})$, but may be reformulated as follows:

PROPERTY 2. *Let $\underline{B}=\langle I\cup S,\ \Sigma S,\ \alpha\rangle$ be a specification base. Then the restriction $\underline{W}(\underline{B})|_J$ of the model is equal to the model of the restriction of $\underline{B}$, $\underline{W}(\underline{B})|_J)$.*

THEOREM 1. *Let $\underline{B}=\langle I\cup S,\ \Sigma S,\ \alpha\rangle$ be a specification base, and let $\underline{W}(\underline{B})$ be a model of this base. For every $J\in\mathcal{P}(I)$, $J\neq\emptyset$, there exists a restriction $\underline{B}|_J$ of $\underline{B}$ whose model is $\underline{W}(\underline{B})|_J$ (here $\mathcal{P}(I)$ stands for the set of the subsets of I).*

Proof. Consider $J\in\mathcal{P}(I)$, $J\neq\emptyset$, and construct the restriction $\underline{B}|_J$. Let $\underline{W}(\underline{B})$ be the model of the specification base $\underline{B}$, and let $\underline{W}(\underline{B})|_J$ be the restriction of the model $\underline{W}(\underline{B})$ to the subset J. According to Property 2, follows that $\underline{W}(\underline{B})|_J = \underline{W}(\underline{B}|_J)$.

For a given specification base $\underline{B} = \langle I\cup S,\ \Sigma S,\ \alpha\ \rangle$, the construction of an interpretation $\underline{A} \in \underline{I}(\underline{B})$ supposes the choice of both the family of sets A and the operation set OP(A) defined on this family. The construction of an interpretation for a specification base corresponds to the construction of the level one (HAS_1) from the HAS (heterogeneous algebraic structure) hierarchy [RUS80] of a language specification. According to the principles of construction of a HAS hierarchy, the level one (HAS_1) will constitute the specification base for the level two (HAS_2). The transit from level one to level two is performed by constructing a new interpretation of the specification base HAS_1. This process can continue until the specified language satisfies the requests of the specifier. The new theoretical concepts presented in this paper are valid for every level HAS_i from a language specification. For simplicity reasons we discussed only the level one from the HAS hierarchy.

Let $LB_1=\langle A_1,\varphi_1,\psi_1\rangle$, $LB_2=\langle A_2,\varphi_2,\psi_2\rangle$ be two programming languages specified by the same base $\underline{B}$.

DEFINITION 8. *The language LB_1 is subset of LB_2 if $A_1 \subseteq A_2$, and, if $w_1 \in OP(A_1)$, then either $w_1 \in OP(A_2)$ or w_1 is a restriction of an operation $w_2 \in OP(A_2)$.*

COROLLARY. *Let $=\underline{B}=\langle I\cup S, \Sigma S, \alpha\rangle$ be a specification base. The language $LB|_J$ specified by the restriction $\underline{B}|_J$ of the base $\underline{B}$ is a subset of the language LB specified by the base $\underline{B}$.*

Proof. This is evident by virtue of Theorem 1.

One can construct in this way a large specification base $\underline{B}$ able to generate a subset of the object specified by $\underline{B}$, by means of the operation of restriction to a given subset J. Denoting by UL the language (object) specified by the base $\underline{B}$, a subset of UL can be obtained as an object specified by a restriction of $\underline{B}$.

4. Examples. The examples which follow concern the simple language specified from a syntactical point of view in Section 2. The set I consists of all words written in capitals from the context-free grammar of the language, while the set S consists of all words written in small letters from the same grammar.

Example 1. Consider J_1 = I\{PREDEFINITION_TYPE}. Remove from ΣS the operation schemes σ_{33}, σ_{34}, which are PREDEFINITION_TYPE. Replace the operation scheme σ_{30} by a restriction of this one to the subset J_1, denoted σ_{30}'=<1, ,INTEGER TYPE_DESIGNER>, and replace the operation scheme σ_{32} by a restriction of this one to the same subset J_1, denoted σ_{32}'=<2, new , INTEGER CONSTRAINT NUMERICAL_TYPE>. The restriction of the specification base $\underline{B}$ to the subset J_1 is $\underline{B}|_{J1}$ = $<J_1 \cup S,\ \Sigma S|_{J1}, \alpha|_{J1}>$, where $\Sigma S|_{J1}$ = $(\Sigma S \backslash \{\sigma_{30}, \sigma_{32}, \sigma_{33}, \sigma_{34}\}) \cup \{\sigma_{30}', \sigma_{32}'\}$. The following numerical-type declaration

type REAL_MIC is new FLOAT range -10.0..10.0

is correct in the language LB specified by $\underline{B}$, but not in the language $LB|_{J1}$ specified by $\underline{B}|_{J1}$. This last language admits the following numerical-type declaration

type INTEGER_MIC is new INTEGER range -10 .. 10

which is a restriction of the first declaration admitted by *LB*. Neither the operation scheme σ_{30} nor its restriction σ_{30}' contribute directly to the specification of objects of *LB* or $LB|_{J1}$, respectively. The objects of the type TYPE_DESIGNER specified by σ_{30}' will contribute to the specification of objects of the type ARRAY_TYPE, as we shall see in the next example. The axiom set $\alpha|_{J1}$ remains the same as the axiom set α. In these conditions the language $LB|_{J1}$ is a subset of the language LB.

Example 2. Consider $J_2 = J_1 \setminus \{TYPE\_DESIGNER\}$. Remove from $\Sigma S|_{J1}$ the operation schemes σ_{28}, σ_{29}, which are TYPE_DESIGNER. We choose the following restrictions for σ_{26}

σ_{26}'=<2, subtype is ,subtype_NAME INTEGER SUBTYPE_DEFINITION>,

and for σ_{27}

σ_{27}' =<3, subtype is ,subtype_NAME INTEGER CONSTRAINT SUBTYPE_DEFINITION>.

For σ_{35} and σ_{36} we choose respectively the restrictions

σ_{35}'=<2, : ,INTEGER_LIST INTEGER OBJECT_DECLARATION>,

σ_{36}'=<3, : := ,IDENTIFIER_LIST INTEGER SIMPLE_EXPRESSION OBJECT_DECLARATION>.

Lastly, one chooses only one restriction for σ_{44}

σ_{44}'=<2, array( ) of ,INDEX_LIST INTEGER ARRAY_TYPE>.

In these conditions, $B|_{J_2} = \langle J_2 \cup S, \sum S|_{J_2}, \alpha|_{J_2} \rangle$, *where*

$$\sum S|_{J_2} = (\sum S|_{J_1} \setminus \{\sigma_{26}, \sigma_{27}, \sigma_{28}, \sigma_{29}, \sigma_{35}, \sigma_{36}, \sigma_{44}\} \cup \{\sigma_{26}', \sigma_{27}', \sigma_{35}', \sigma_{36}', \sigma_{44}'\}.$$

The following array-type declaration

type ARR_TY is array (1..10, 1..10) of FLOAT

is correct in the languages LB and $LB|_{J1}$, but not in $LB|_{J2}$. The language $LB|_{J2}$ admits the definition of an array-type restriction (reduced from) whose elements are of integer-type. So, above array-type definition reduces in the language $LB|_{J2}$ to

type ARR_TY is array (1..10, 1..10) of INTEGER.

The following record-type declaration

type COMPLEX is

record

```
        REAL : FLOAT;
        IMAG : FLOAT;
    end record.
```

is correct in the languages *LB* and $LB|_{J1}$. The language specified by J_2 ,$LB|_{J2}$, admits only a restriction of the object COMPLEX, namely

```
    type COMPLEX is
      record
        REAL : INTEGER;
        IMAG : INTEGER;
    end record.
```

Also in this example the axiom sets $\alpha|_{J2}$ and α coincide. In these conditions the language $LB|_{J2}$ is a subset of the language $LB|_{J1}$.

Example 3. Consider $J_3 = J_2 \backslash \{\text{IF\_LIST}\}$. Remove from $\Sigma S|_{J2}$ the operation schemes σ_{49} and σ_{50}, which are of the type IF_LIST. We choose respectively for σ_{51} and σ_{52} the restrictions

σ_{51}' = <1, if endif, EL_IF IF>,

σ_{52}' = <2, if else endif, EL_IF SEQUENCE IF>.

In these conditions $B|_{J_3} = \langle J_3 \cup S, \sum S|_{J_3}, \alpha|_{J_3} \rangle$, where $\sum S|_{J_3} = (\sum S|_{J_2} \backslash \{ \sigma_{49}, \sigma_{50}, \sigma_{51}, \sigma_{52} \}) \cup \{ \sigma_{51}', \sigma_{52}' \}$. An instruction of the form

```
if RELATION then
          SEQUENCE
{elsif RELATION then
             SEQUENCE}
```

[else

SEQUENCE]

endif.

is admitted in the languages LB, $LB|_{J1}$, $LB|_{J2}$. The language $LB|_{J3}$ admits the following reduced forms

if RELATION then SEQUENCE endif,

if RELATION then SEQUENCE

else SEQUENCE

endif.

The axiom set $\alpha|_{J3}$ coincides with α, too. In these conditions the language $LB|_{J3}$ is a subset of the language $LB|_{J2}$.

R E F E R E N C E S

[ADJ73] J.A. Goguen, J.W. Thatcher, G.Wagner, J.B. Wright: *A junction between computer science and category theory, Basic definitions and examples, Part 1,* IBM Research Report RC-4526, 1973.

[ADJ77] J.A. Goguen, J.W. Thatcher, E.G. Wagner, J.B. Wright: *Initial Algebra for Computing Machinery,* vol. 24, no.1, 1977.

[BRO82] M. Broy, M. Wirsing,: *Partial Abstract Types,* Acta Informatica, 18, 47-64 (1982).

[BRO87] M. Broy, M.Wirsing, P.Pepper: *On the Algebraic Definition of Programming Languages,* ACM Transactions on Programming Languages and Systems, vol 9., no.1, 1987.

[RUS72] T. Rus: *ΣS-Algebra of a Formal Language,* Bul. Math. de la Soc. de Science, Bucharest, 15(63):2, 227-235, 1972.

[RUS80] T. Rus: HAS-Hierarchy: *A natural tool for Languag specification,* Ann. Soc. Math. Polonae, Series IV. Fundamenta Informaticae, 3:3,269-274, 1980.

[RUS83] T. Rus: *Mecanisme formale pentru specificarea limbajelor,* Editura Academiei R.S.R., 1983, Bucuresti.

[HIG63] P.J. Higgins: *Algebras with a scheme of operators,* Math. Nachr., 27, 115-132, 1963/64.

[BIR70] G. Birkoff, J.D. Lipson: *Heterogeneous algebras,* Journal of Combinatorial Theory, 8, 115-132, 1970.

[WAG89] E.G. Wagner: *An Algebraically 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.