STUDIA
UNIVERSITATIS BABEŞ-BOLYAI

MATHEMATICA

3

1987

CLUJ-NAPOCA

Anul XXXII

1987

STUDIA
UNIVERSITATIS BABEŞ-BOLYAI

MATHEMATICA

3

Redacţia: 3400 CLUJ NAPOCA, Str. M. Kogălniceanu 1 ● Telefon 1 61 01

SUMAR — CONTENTS

STUDIA UNIV. BABEŞ–BOLYAI, MATHEMATICA, XXXII, 3, 1987

THE *HAS* HIERARCHY AND THE EXTENSIBILITY OF THE PROGRAMMING LANGUAGES

ILIE PARPUCEA*

Received: February 25, 1987

REZUMAT. — Ierarhia HAS şi extensibilitatea limbajelor de programare. Tematica abordată în această lucrare o constituie formalizarea metematică (algebrică) a extensibilităţii limbajelor de programare. Formele semantice îmbracă o formă nouă privită dintr-un capitol modern al matematicii, teoria algebrelor universale. Abordarea de pe poziţii algebrice a acestei probleme fundamentează matematic abordarea unor probleme legate de implementarea, specificarea şi dezvoltarea limbajelor.

1. Introduction. As man-computer communication means, the programming languages have constituted and still constitute a topic tackled by the majority of the researchers in informatics, having profound implications about the use of computing equipments. This paper constitutes an attempt to point out some formal looks concerning the extensibility of the languages, using the heterogeneous algebraic structure hierarchy. The algebraic modelling of the programming language specification and especially the extensibility of these allow for the development of certain calculation models, reliable and efficient from the point of view of the user, and point out generation mechanisms of a syntax naturally associated to a given semantics, as well as connection and estimate schemes intrinsically connected to the communicators for which the programming language is specified. The *heterogeneous algebraic structures* (abbreviated *HAS*), a relatively recent concept, satisfy best the above stated characteristics.

The macro-assembler facilities pointed out in the frame of the programming language extensibility play an important part in the apparition and generalization of similar ideas for the extension of the advanced languages. The idea of the macro-generation found a fertile field, especially the last decade, due to its part played in the extensibility and portability of the programming systems.

The computer programming practice and the peculiar field of the design and implementation of operating systems have pointed out the necessity of improved, more flexible and more complete extensible type mechanisms.

The abstracting of the notion of type had initially a limited area: the standardization of a set of abstract operations for object handling. Such a standard set of abstract operations will not be sufficient for an adequate handling of any object. The selection of the operations must be put at user's disposal, the user being the only one able to factorize in his type definition the invariant features of the used abstractings.

The abstract types are used in the peculiar application field of operating system programming. With the new abstractings, the operating system of a

* *University of Cluj-Napoca, Computing Data Center, 3400 Cluj-Napoca, Romania*

computing system can be modelled as a set of types, each resource constituting an object.

2. An algebraic model concerning the representation of the semantic forms from a programming language. A programming language can be considered as being a triple of the form $LP = (M_P, F_P, f: M_P - F_P)$, where:

M_P = collection of calculus abstractions;

F_P = collection of symbols wich represent the abstractions of M_P for the communicators which use the LP language;

f = function which associates to every abstract object of M_P the representation form in F_P.

The collection M_P can be specified by means of a computing system, which is a pair $(D, 0)$ in which D is a collection of abstract types of data, while 0 is a collection of operations upon the types of D.

The set M_P corresponds to the semantics of a language, while F_P corresponds to the syntax of this one. By means of f, the existence of a rule for associating each abstraction $m \in M_P$ with its symbolic representation $f(m) \in F_P$ is ensured.

Using the heterogeneous algebraic structures, the semantics M_P of a programming language will be represented under the form of a computing systtem.

We shall refer further down to a M_P specification algebraic model, based on the concept of heterogeneous algebraic structure *(HAS)*. Since its apparition, the concept of *HAS* proved to be a strong tool, able to solve many difficult problems of the programming languages. This will allow a mathematically formalized representation of all the types of abstractions (objects) contained into a language. The *HAS* mechanism used further down is known under the denominations of *HAS* hierarchy and has been introduced by *T.* Rus in 1975. The aim of this mechanism (called the heterogeneous algebraic hierarchy or, more exactly, the *HAS* hierarchy) is to offer a development frame to some mathematical (algebraic) theories directed by certain applications. Each level in the frame of the hierarchy has a certain degree of connection with the initial application to be modelled. By passing from a level of this hierarchy to an upper level, the fidelity of the obtained theory in modelling the initial application increases; conversely, by passing from a level of the hierarchy to a lower hierarchical level, the degree of connection with the initial application decreases, i.e. the fidelity of the obtained theory in modelling the given initial application decreases. Taking into account on one hand the heterogeneous character of the real applications (from the point of view of the concrete objects implicated in such applications), and on another hand the idealization of these features of heterogenization of the respective objects by their modelling by means of the actual mathematical theories, it is easy to emphasize the necessity of such hierarchies directed by the degree of heterogenization of the abstract or concrete objects which they contain and model. The development of a mathematical tool according to the purpose of the true modelling of the real applications implies two basic principles:

a) Any homogeneous algebraic structure is a *HAS* of hierarchical zero level in a *HAS* hierarchy.

b) Any i-level HAS can be chosen as basis for an $(i+1)$-level. The basic concept used to formulating this principle has the meaning of specifier. So, if the i-level HAS is given under the form:

$$HAS = (I, \Omega),$$

where I is the support, while Ω is the collection of operations of the structure, then:

(i) The support of the i-level HAS is used as index set for specifying the support of the $(i+1)$-level HAS. Namely if A is the support of the $(i+1)$-level HAS, then A must be considered as a family of sets, each set of this family being specified by an element of the index set I. Every $i \in I$ is transformed in the $(i+1)$-level HAS into the set A_i. In this manner, the support of the $(i+1)$-level HAS becomes the family $A = (A_i)_{i \in I}$.

(ii) We denote by $R_n(i)$ the set of all n-ary relationships defined in the i-level HAS. An n-ary relationship $r \in R_n(i)$ becomes in the $(i+1)$-level HAS a relationship scheme which specifies by factorization the following family of relationships:

$$\Sigma_r = \{r_{i_1 i_2 \ldots i_n} \subseteq A_{i_1} \times A_{i_2} \times \ldots \times A_{i_n} | (i_1, i_2, \ldots, i_n) \in r\}.$$

In other words, each n-uple $(i_1, i_2, \ldots, i_n)$ belonging to the relationship r specifies a certain n-ary relationship in the $(i+1)$-level HAS. One notices than an n-ary relationship in the i-level HAS acts in the $(i+1)$-level HAS as a family of relationships. This means that the hierarchical level i factorizes the features of the hierarchical level $i+1$, each element of the support of the $(i+1)$-level HAS behaving as an equivalence class of the elements of the support of the $(i+1)$-level HAS. The equivalence relationship is given by a certain specific set of features which must be „forgotten" for reaching the elements of the support of the i-level HAS. Conversely, each element of the $(i+1)$-level HAS is considered as being a set specified by the adjunction of the features „forgotten" for obtaining the elements of the support set of the i-level HAS.

Further down, the i-level HAS and the $(i+1)$-level HAS will be simply denominated $HAS(i)$ and $HAS(i+1)$, respectively. We shall also make the following notation convention: the $HAS(i)$ is called base with respect to the $HAS(i+1)$ from the same hierarchy and is simply denoted:

$$\mathcal{B} = \{B, \Omega_B\},$$

where B is the support set of the $HAS(i)$, while Ω_B is the set of the operations which define the structure $\mathcal{B}$. So, in the same hierarchy, the $HAS(i+1)$ is specified on the base $\mathcal{B}$, according to the principle (b), under the form of the triple:

$$\mathcal{A} = (A = (A_b)_{b \in B}, \Sigma = (\Sigma_w)_{w \in \Omega_B}, F),$$

where the following notations were used:

$A = (A_b)_{b \in B}$ is a family of sets indexed by the support of the base;

$\Sigma = (\Sigma_w)_{w \in \Omega_B}$ is a family of operation schemes specified by the operations of the base. Each n-ary operation $w \in \Omega_B$ belonging to the operations of the base induces in the structure specified by the base a set of operation schemes defined as follows:

$$\Sigma_w = \{\sigma = (b_1 b_2 \ldots b_n b) \mid b = w(b_1, b_2, \ldots, b_n)\}.$$

For clearness, we shall include into the operation schemes specified by $w \in \Omega_B$ both the symbol w and its n-arity. Therefore, the operation schemes from Σ_w will be denoted by:

$$\sigma = \{n, w, b_1 b_2 \ldots b_n b\},$$

where n is the n-arity of the operation w, while w is the symbol of the operation specified by such a scheme. Since the operations specified by means of the schemes are heterogeneous operations, one considers the symbol of the operation w as being somewhat more general, namely distributed upon its operands. The operation scheme acquires the form:

$$\sigma = \{n, s_0 s_1 \ldots s_n, b_1 b_2 \ldots b_n b\},$$

where $s_0 s_1 \ldots s_n$ is considered to be the sign of the operation distributed upon the operands and called sometimes the word of state of the operation.

In specifying the $HAS(i + 1)$ by the $HAS(i)$ given under the form of a triple, we have also used the symbol F, which is considered to be the symbol of a function associating to each operation scheme $\sigma \in \Sigma_w$, $w \in \Omega_B$, a heterogeneous operation specific for the $HAS(i + 1)$. The domain of definition, the range of the operation, the n-arity and the symbol of such an operation, all these are obviously determined by the scheme σ, while the acting mode is specific for $HAS(i + 1)$, being subsequently determined by F.

If $\sigma = \{n, s_0 s_1 \ldots s_n, b_1 b_2 \ldots b_n b\}$, then F is a specific operation in the $HAS(i + 1)$ as follows:

$$F(\sigma): A_{b_1} \times A_{b_2} \times \ldots \times A_{b_n} \rightarrow A_b.$$

In this manner, one associates to every class Σ_w a family of heterogeneous operations, each of them being denoted by the same symbol $s_0 s_1 \ldots s_n$, but acting differently because their domains of definition and ranges are different. This name ambiguity is called sometimes the overload of the name $s_0 s_1 \ldots s_n$ and can be removed by either the context, or the adequate modification of the symbol $s_0 s_1 \ldots s_n$.

These few theoretical notions concerning the HAS hierarchy take into account the necessity to create a formal mechanism for specifying a concept of abstract computing system able to constitute the semantics for a programming language. A computing system, as it has been noticed, can be characterized as being a pair $HAS = \{D, 0\}$, in which D is a collection of objects, called calculation objects or data of the system, while 0 is a collection of actions upon the objects of D and the actions of 0 form the operations of the computing system. Both the objects of D and the actions of 0 can be characterized as belonging to two types: primitive and composite. In other words, every calculation object $d \in D$ can be considered as a primitive object (being called in this case primitive calculation abstraction), or can be considered as a composite object (being called in this case composite calculation abstraction). Similarly, every $o \in 0$ can be considered as a primitive operation acting upon the data of D, or can be considered as a composite operation, obtained by the composition of other operations of 0. One can perform an analysis of the objects of D taking into account the following features: the intrinsic nature, the mechanism of new abstraction specification in terms of given abstractions, the estimate scheme of a given abstraction in terms of the component abstractions. For the primitive objects, the intrinsic nature specifies a collection of primitive objects as function of their behaviour with respect to a given set of operations. The intrinsic nature refers to the so-called type of behaviour with respect to a given set of operations, depending not on the objects subjected to these operations. The behaviour types are most often defined by means of identities associated to the calculation system HAS, under the form of behaviour axioms (called sometimes definition axioms, too). The mechanism of new abstraction specification in terms of given abstractions refers to the fact that if $d_1, d_2, \ldots, d_n$ are given abstractions in D, then there exist in 0 operation schemes which allow the definition of a new abstraction, d, in terms of the abstractions $d_1, d_2, \ldots, d_n$. In other words, for an adequate choice of $o \in 0$, we have $d = o(d_1, d_2, \ldots, d_n)$. The operation scheme which provides the operation o is called the mechanism of specification of the abstraction d in terms of the abstractions $d_1, d_2, \ldots, d_n$.

The performing (or estimate) scheme of an abstraction in terms of the component abstractions must be discussed as follows:

(a) In the case of the primitive abstractions, the performing scheme is hidden because of the primitivity of the respective abstractions.

(b) In the case of the composite abstractions, the performing or estimate scheme of an abstraction in terms of the component abstractions is an explicit datum.

The performing scheme provides the model or algorithm of obtaining a composite object (or abstraction) in terms of the component objects (or abstractions).

If we consider a class of composite objects in a calculation system, and we „forget" the performing scheme of these objects in terms of the component objects, considering them as implicit data, then we obtain an abstraction which forms a new primitive class. Therefore, the notions of primitive object and composite object in a calculation system are relative.

Concluding, as for the attributes featuring the objects of the set D as being the collection of the calculation objects for a computing system $HAS = \{D, 0\}$, we can consider D as being decomposed on two levels as follows:

(d_1) the collection $D_p \subseteq D$ of the primitive objects;

(d_2) the collection $D_c \subseteq D$ of the composite objects.

The collections D_p and D_c can also be regarded under the form of families of behaviour types as function of the operations of 0, namely:

$$D_p = (D_{pi})_{i \in I}; \quad D_c = (D_{cj})_{j \in J},$$

where I and J are index sets adequately chosen for describing all the behaviour types defined in terms of given identity systems. Let these identities be given under the form of the family $E = (E_i)_{i \in I} \cup (E_j)_{j \in J}$, where E_i and

E_j define the behaviour types i and j. The abstract computing system HAS becomes the triple:

$$HAS = \{D = (D_{pi})_{i \in I} \cup (D_{cj})_{j \in J},\ E,\ 0\}.$$

A similar study about the operations of 0 will lead us to the conclusion that these ones can also be stratified on two levels:

(o_1) the operations featuring the behaviour types of $D_p = (D_{pi})_{i \in I}$, called primitive operations;

(o_2) the operations featuring the behaviour types of $D_c = (D_{cj})_{j \in J}$, called composite operations.

So, the abstract computing system acquires the form:

$$HAS = \{D = (D_{pi})_{i \in I} \cup (D_{cj})_{j \in J},\ E = (E_{pi})_{i \in I} \cup (E_{cj})_{i \in J},$$
$$0 = (0_{pi})_{i \in I} \cup (0_{cj})_{j \in J}\},$$

in which we have:

(a) For every $i \in I$, D_{pi} is an abstract algebraic structure specified by the set of identities E_{pi} with respect to the operations 0_{pi}.

(b) For every $j \in J$, D_{cj} is a structure of a composite calculation objects, specified by a subset of the operations 0_{cj} called definition (or forming) operations of the objects of D_{cj}. These ones form an abstract algebraic structure specified by the identities E_{cj} with respect to a subset of the operations of 0_{cj} called calculation operations in D_{cj}.

In order ot handle the objects of the $HAS = \{D, E, 0\}$ with respect to the operations of 0, we shall refer to the data of D under the denomination of constants or variables. For $i \in I$, an object $d \in D_{pi}$ will be called i-type primitive constant. A symbol x which can denominate every i-type constant will be called i-type variable. Analogously, for $j \in J$, an object $d \in D_{cj}$ is called j-type composite constant, while a symbol y which can denominate every j-type constant is called j-type variable.

The calculation needs, expressible in a concrete HAS of the above described kind, are generally represented by means of sequences of operations of 0 upon calculation objects, constant or variable, precised as previously.

A determined succession of operations upon a given set of constant and variable calculation objects in the $HAS = \{D, E, 0\}$, which, performed in the given order, leads to a k-type calculation object, is a k-type expression.

If $o_1, o_2, \ldots, o_n$ are the operations to be performed in the given order upon the constants $c_1, c_2, \ldots, c_m$ and the variables $x_1, x_2, \ldots, x_p$ for defining a k-type expression, we denote this expression by:

$$E(k)[o_1, o_2, \ldots, o_n;\ (c_1, c_2, \ldots, c_m),\ (x_1, x_2, \ldots, x_p)],$$

or, shorterly, $E(k)$. Obviously, $E(k)$ defines a composite operation $o \in 0$ with the n-arity p.

An expression having the form:

$$x := E(k)[o_1, o_2, \ldots, c_n;\ (c_1, c_2, \ldots, c_m),\ (x_1, x_2, \ldots, x_p)]$$

is called assignment expression and constitutes a primitive calculation unit. A calculation process can now be regarded as an ordered sequence of primitive calculation units.

Upon the calculation processes, one can define the following types of operations: the concatenation operation, the iteration operation and the selection operation.

This has constituted an unformalized vision upon a *HAS* and upon the related concepts. We shall present further down, by using the *HAS* hierarchy model, the formal specification of the concept of *HAS*.

The formal specification of the *HAS* consists of the construction of an algebraic model of the respective system and of a symbolism by means of which we could represent the calculation concepts implicated in the *HAS*. The algebraic model is constructed by using the *HAS* hierarchy concept.

We consider I as being a finite set of symbols, called basic index, and S as being a finite set of symbols, called the set of words of state of the calculation system, such that $I \cap S = \Phi$. The elements of the set I are called primitive types.

The base $B = \{S^+, I^+, \lambda\}$ will constitute the basis on which the *HAS* will be specified. S^+ and I^+ are the free semigroups generated by the concatenation operation in the symbols of S and I, respectively, while $\lambda \subseteq S^+ \times I^+$ is a finite relationship. If $(x, y) \in \lambda$, then $\lambda(x) = \lambda(y)$. Let be $x = s_0 s_1 \ldots s_n$, $s_i \in S$, $i = \overline{0, n}$, and $y = i_1 i_2 \ldots i_n i$, i and $i_k \in I$, $k = \overline{1, n}$; then the triple $\sigma = \{n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i\}$ specifies an operation scheme in the *HAS*. We denote by ΣS the set of all the operation schemes, provided by B, namely:

$$\Sigma S = \{(n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n i) \mid (s_0 s_1 \ldots s_n; i_1 i_0 \ldots i_n i) \in \lambda,$$
$$\lambda(s_0 s_1 \ldots s_n) = \lambda(i_1 i_2 \ldots i_n i) = n + 1\}.$$

We choose a family of sets indexed by I as support for a heterogeneous algebra specified by B. We denote by $D_p = (D_{pi})_{i \in I}$ the collection of primitive data, and by $\Sigma S(i)$ the set of operation schemes for obtaining the elements belonging to the primitive type i. Then $\Sigma S_B = (\Sigma S(i))_{i \in I}$.

Now we consider the set J of all types of calculation objects constructed from the objects $D_p = (D_{pi})_{i \in I}$. J represents the set of the types of composite calculation objects of the *HAS*.

We extend the set I in the base B by means of the set J, and the set S by means of the set S_1 of the symbols of the composite operations distributed upon their operands. Assuming that $S \cap S_1 = \Phi$, then we obtain a new base, namely $B_1 = \{(S \cup S_1)^+, (I \cup J)^+, \lambda_1\}$, where $\lambda_1 = \lambda \cup \lambda'$ and:

$$\lambda' = \{(s_0' s_1' \ldots s_n', i_1 i_2 \ldots i_n j) \mid s_k' \in S_1, k = \overline{0, n},$$
$$i_k \in I, k = \overline{1, n}, j \in J\}.$$

The base B_1 defined in this manner allows the specification of a new *HAS* which we denote by:

$$A = \{D = (D_{pi})_{i \in I} \cup (D_{cj})_{j \in J}, \Sigma S_B \cup \Sigma S', F\},$$

where $\Sigma S'$ represents the set of operation schemes provided by the pairs from λ'.

By iterating the above described process for a finite number of times, one obtains a heterogeneous algebraic structure, which we denote by A_{HAS} and call heterogeneous algebra associated to the HAS, under the form:

$$A_{HAS} = \{D = (D_{pi})_{i\in I} \bigcup (D_{cj})_{j\in J}, \ \Sigma S_{HAS}, \ F_{HAS}\}.$$

For every $\sigma \in \Sigma S_{HAS}$, there exists $F_{HAS} \in 0$, which is either a calculation operation, or an operation of forming a new type of calculation object from given calculation objects (primitive or composite).

If $\sigma \in \Sigma S(j)$, then $\sigma = (n, s_0 s_1 \ldots s_n, j_1 j_2 \ldots j_n j)$ and $F_{HAS}(\sigma) : Dj_1 \times Dj_2 \times \ldots \times Dj_n \to Dj$ represents a composite operation which defines a j-type object in terms of the objects of the types $j_1, j_2, \ldots, j$.

For every type $k \in I \bigcup J$, the objects of the type k are the constants or the variables of the respective type.

The set $C_k = \{(o, s, k) \mid (o, s, k) \in \Sigma S_k\}$ can be regarded on one hand as the set of the nullary operations of the type k, and on another hand as the set of the symbols $s \in S$ which denote these operations. One assumes that there exists amongst the individual constants of the type k at least one, called the exception (or the error) of the type k, which is denoted by $Er(k)$ and plays the part of annuller of the type k with respect to every operation which contains the type k as operand. So:

(a) $\forall k \in I \bigcup J, \ \exists \ Er(k) \in D_k$, where $D_k = D_{pk}$ or $D_k = D_{ck}$.

(b) If $\sigma \in \Sigma S_k$ and $\sigma = (n, \ s_0 s_1 \ldots s_n, \ j_1 j_2 \ldots j_n k)$, and for a given i, $1 \leqslant i \leqslant n$, $d_i = Er(j_i)$, then:

$$F_{HAS}(\sigma)(d_1, d_2, \ldots, d_n) = Er(k).$$

The variables of the type k are specified by means of a set of symbols $VR(k)$, so that $\forall k, \ VR(k) \bigcap S = \Phi$ and $VR(k) \bigcap (I \bigcup J) = \Phi$. Therefore, one adds to ΣS_k the set of triples $\{(o, v, k) \mid v \in VR(k)\}$.

The set of nullary operation schemes from ΣS_k specifies the constants and the variables of the type k according to the following proceeding:

— if $(o, s, k) \in \Sigma S_k$ and $s \in S$, then s represents a k-type constant;

— if $(o, s, k) \in \Sigma S_k$ and $s \in VR(k)$, then s represents a k-type variable.

In order to define the concept of calculation process in the HAS specified by means of the algebra A_{HAS}, the concept of k-type expression, $k \in I \bigcup J$, will be formalized.

Let therefore

$$A_{HAS} = \{D = (D_{pi})_{i\in I} \bigcup (D_{cj})_{j\in J}, \ \ \Sigma S_{HAS}, \ F_{HAS}\}$$

be the algebra for specifying the HAS, and $k \in I \bigcup J$. The concept of k-type formal expression is defined as follows:

(a) If $c(k)$ is a k-type constant, then $c(k)$ is called k-type formal expression. If $\sigma = (o, s, k)$ specifies the constant $c(k)$, then $F_{HAS}(\sigma) = s$.

(b) If $v(k)$ is a k-type variable, then $v(k)$ is called k-type formal expression. For $\sigma = (o, s, k)$ which specifies the variable $v(k)$, $F_{HAS}(\sigma)$ is the k-type object denominated at the moment of applying $F_{HAS}(\sigma)$ by the symbol s.

(c) If $\sigma \in \Sigma S_{HAS}$ and $\sigma = (n, s_0 s_1 \ldots s_n, j_1 j_2 \ldots j_n k)$, and $F_{HAS}(\sigma): D_{j_1} \times \times D_{j_2} \times \ldots \times D_{j_n} \to D_k$, then for $d_r \in D_{j_r}$, $r = \overline{1, n}$, $F_{HAS}(\sigma)(d_1, d_2, \ldots, d_n) = s_0 d_1 s_1 \ldots d_n s_n$, and we call $s_0 d_1 s_1 \ldots d_n s_n$ k-type expression.

(d) Any k-type expression can be constructed according to the rules (a), (b), (c).

The above defined concept of k-type expression leads us to the construction of the family of symbols $W = (W(k))_{k \in I \cup J}$, where $W(k)$ is the set of the k-type expressions. The construction of this family behas with respect to the operations provided by the operation schemes from ΣS_{HAS} similarly to the behaviour of the set D with respect to the same operations.

The triple:

$$W = \{W = (W_p(i))_{i \in I} \cup (W_c(j))_{j \in J}, \Sigma S_{HAS}, F_{HAS}\},$$

in which the function F_{HAS} acts on ΣS_{HAS} according to the rules of expression construction, is an algebra similar to the heterogeneous algebra A_{HAS} associated to the HAS.

We shall not insist about the process of estimating a k-type expression from the algebra W; see T. Rus [1].

Every formal expression $w \in W_k$, $k \in I \cup J$, denotes a certain calculation object of the HAS specified by A_{HAS}. The above mentioned estimating process constructs such objects starting from $w \in W_k$.

The considered HAS is still too poor; it cannot yet be considered a true and complete model for the real calculation systems often used as semantics of the various programming languages. We shall further down enrich the calculation system constructed through the algebra HAS, by introducing new objects which we call calculation unit, simple calculation process and composite calculation process. These ones will be introduced as new types of calculation objects in A_{HAS}, to which one associates a representation mode by using the formal expressions in w.

Let $w \in W_k$, $k \in I \cup J$, be a formal expression, that is w is the formal representation of a k-type calculation object in A_{HAS}. From the construction rules of w, it results to be composed on one hand by constants and/or variables, and on another hand by symbols of operations distributed upon the operands. Therefore w appears as being of the form:

$$w(c_1, c_2, \ldots, c_p; x_1, x_2, \ldots, x_n),$$

where $c_1, c_2, \ldots, c_p$ are all the different constants from w, while $x_1, x_2, \ldots, x_n$ are all the different variables from w. If $i_1, i_2, \ldots, i_n$ represent the types of the variables $x_1, x_2, \ldots, x_n$, then w can be considered as a derived operation defined by an operation scheme of the form:

$$\{n, s_0 s_1 \ldots s_n, i_1 i_2 \ldots i_n k\},$$

where $s_0 s_1 \ldots s_n$ are determined from the structure of w, namely:

$$w: D_{i_1} \times D_{i_2} \times \ldots \times D_{i_n} \to D_k.$$

This function makes $w(c_1, c_2, \ldots, c_p; d_1, d_2, \ldots, d_n)$ to constitute, for any system of values $(d_1, d_2, \ldots, d_n)$ of the types $i_1, i_2, \ldots, i_n$, respectively, a k-type calculation object constructed according the estimation process applied to w. In other words, in the conventional programming languages, w represents a procedure whose formal parameters are $(x_r, i_r), r = \overline{1, n}$, where each pair represents a variable together with its type. Unlike these ones, the derived operation represented by w appears as a composite operation, determined by an operation scheme whom universal estimation scheme is embedded in the process of estimation of the formal expressions.

The abstractions represented by w are characterized by:

(a) The intrinsic nature is given by the type of represented object.

(b) The estimation scheme represented by the estimation process of the formal expressions w.

(c) The representation scheme of w is given by the forming rules of the formal expression represented by w.

This fact allows the definition of a concept of universal and standardized calculation unit, embedded into the formal expressions.

A primitive calculation unit means a calculation object (generally composite) of the *HAS* consisting of:

(a) A construction process of the calculation object in the *HAS*.

(b) A process of identification of the constructed object.

A primitive calculation unit can be represented by means of the symbolism $x := w$, where w is a k-type formal expression, while x is a k-type variable.

We can refer to a primitive calculation unit either anonymously, by simply writing the expression of the form $x := w$, or by a symbolic name which we call label. In this last case, the primitive calculation unit becomes denominated or labelled and appears in the conventional programming languages under the form:

$$\langle LABEL \rangle\, x := w.$$

Examples of calculation objects of the above discussed kind are offered by the conventional programming languages under the names of functions, subprograms, procedures, modules, etc.

Therefore a primitive calculation unit is a calculation object specificable by means of the anonymous or denominated derived operation, which, for certain values assigned to the variables which it contains, leads to a calculation object of a given type.

Taking into account the characteristics of the abstractions handled by a *HAS*, the objects of the type „primitive calculation unit" can be discussed as follows:

(a) The intrinsic nature of a „primitive calculation unit" — type object is the derived operation embedded into such a unit, which is, as a matter of fact, its specification mechanism, too.

(b) The estimation scheme of a calculation unit.

(c) The representation scheme of the calculation units is given through formal expressions of the form $x := w$ for the case of the anonymous derived operations, $\langle LABEL \rangle\ x := w$ for the case of the labelled derived operations, and, finally, of the form $NAME((x_1, v_1, i_1); (x_2, v_2, i_2), \ldots, (x_n, v_n, i_n))$ for the case of the derived operations. The triples (x_j, v_j, i_j), $j = \overline{1, n}$, are called association triples (*TA*) of the formal parameters with the effective ones, while the list $((x_1, v_1, i_1), (x_2, v_2, i_2), \ldots, (x_n, v_n, i_n))$ is called triple association list *(LA)* of the formal parameters with their actual values corresponding to a calculation object determined by the estimate of the implicated derived operation.

We extend the domain of the types of constructed calculation objects of the *HAS* specified by means of the algebra A_{HAS} by the new calculation objects specificable by the types implicit or anonymous primitive calculation unit (abbreviated *UPCA*) and explicit or denominated primitive calculation unit (abbreviated *UPCD*). In this manner, the set ΣS_{HAS} of the operation schemes of the *HAS* specified by A_{HAS} is extended with the following new schemes:

$$\{2,\ \varepsilon := ;,\ VAR\ EXP\ UPCA\}$$
$$\{3, (,\ ,),\ VAR\ CONST\ TIP\ TA\}$$
$$\{1,\ \varepsilon, \varepsilon,\ TA,\ LA\}$$
$$\{2,\ \varepsilon, \varepsilon,\ LA\ TA\ LA\}$$
$$\{2,\ \varepsilon(\ \ ),\ VAR\ LA\ UPCD\}$$

In these schemes we have used the following abbreviations: *VAR* for variable, *CONST* for constant, *TIP* for type, *EXP* for expression. The above presented operation schemes describe the manner of representing the objects of the respective types in the algebra *W*.

We agree to use the term of calculation unit for a *UPCA*-type or a *UPCD*-type object. Under the operation scheme form, it appears as follows:

$$\{1,\ \varepsilon\ \varepsilon,\ UPCA\ UC\}$$
$$\{1,\ \varepsilon\ \varepsilon,\ UPCD\ UC\}.$$

The concept of calculation unit constitutes the basic, defining element for the hierarchical construction of the new types of calculation objects, which constitute the calculation system specified by means of the algebra A_{HAS}. By using the composition scheme of the specification operations for calculation units, one introduces the concept of composite calculation unit *(UCC)*. In this manner, the sequential composition operation leads to the introduction of a new type of object, namely the *BLOC*, the selective composition operation leads to the definition of composite objects of the type IF and *CASE*, while the iteration composition leads to the definition of composite objects of the type *FOR*, *DO* or *LOOP*.

The specification mode of the above introduced types allows the definition of the calculation object type *PROCES*.

From the point of view of the conventional programming languages, the types *BLOC*, *IF*, *CASE*, *FOR*, *DO*, *LOOP* constitute instructions which synthesize a sequence of predefined operations, without being formalized in the mathematical meaning.

In the new vision about the semantics (M_P) of a programming language the semantic forms acquire an independent mathematical contour. The semantics itself is regarded as an algebraic structure well determined by the objects and by the operations acting upon these ones. The enrichment of the semantic (algebraic structure) with new types (objects) on the basis of the presented algebraic model is practically unlimited, this depending only on the user's grounding and ability. This new look upon the semantics of a programming language allows to establish a mathematical (algebraic) basis for the development of the programming languages. The extension of a language by new types (calculation abstractions) amounts — on the basis of the new algebraic model — to the introduction of new operations in ΣS_{HAS}, which act upon the objects from D, resulting new types (composite objects) which enrich the set D (the calculation abstractions).

REFERENCES

1. T. Rus, *Mecanisme formale pentru specificarea limbajelor*, Ed. Acad. R.S.R., București, 1983.
2. S. Niculescu, D. P. Goleșteanu, *Macroprocesoare și limbaje extensibile*, Ed. științifică și enciclopedică, București, 1982.
3. I. Purdea, G. Pic, *Tratat de algebră modernă*, Vol. 1, Ed. Acad. R.S.R., București, 1977.

STUDIA UNIV. BABEŞ–BOLYAI, MATHEMATICA, XXXII, 3, 1987

FORTRAN CAN BE IMPROVED

FL. M. BOIAN*, M. FRENŢIU*, Z. KÁSA* and L. ŢÂMBULEA*

Received: May 15, 1987

ABSTRACT. — In this note we will express our opinion **about** some statements and ideas useful for a new standard Fortran.

Fortran is one of the first programming languages and one of the most used of them. Its popularity is due to the simplicity of the language and, also, to the efficiency of the compiler. Today there are many programming languages [4], and some of them are more sophisticated than Fortran. But, also, there are a lot of Fortran programs organised in many libraries, and there is a great experience of programming in Fortran. We think that Fortran must remain a programming language, but it can be improved.

It is well known that Fortran does not have the control and data structures needed for structured programming [3]. This affects the style of programming and the clarity of the programs. The structure IF—THEN—ELSE is already present in the standard Fortran 77 [2, 5]. We suggest that a WHILE structure is also needed.

Some new operations, as declarations and working with integers represented by n binary digits ($n \geq 1$), or operations with matrices (as Basic has), may also be introduced.

Some of the programming languages (Pascal, Ada, etc) permit concurrent programming. Adding only a few new statements concurrent programming may be introduced in Fortran too.

To define a task we need:

— the line of definition, a TASK statement,

— the body of the task,

— the final line, the END statement.

Inside the body of the task a TERM statement is needed to cause the termination of the execution of the task. The execution of a task is called using the statement EXECUTE. The definition of these statements (TASK and EXECUTE), similar with the statements SUBROUTINE and CALL, may be:

$$\text{TASK name } [*c]\ (p_1, p_2, \ldots, p_n)$$

$$\text{EXECUTE name } [*ae]\ (a_1, a_2, \ldots, a_n)$$

where $p_1, p_2, \ldots, p_n$ are dummy arguments and $a_1, a_2, \ldots, a_n$ are the actual arguments, similar to the usage of the parameters of the Fortran subroutines. The positive integer constant c permits to use the task in c copies, and ae is an arithmetic expression that specifies which copy of the task is called.

* *University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania*

At any time a task may be in one of the following states:
— active, i.e. its instructions are executed,
— waiting, if the task is waiting for the appearance of an event,
— inactive, if its execution has been terminated or if it has not been activated.

A function is needed to verify the state of a task. Such a function may be:

$$\text{STATE (name } [*ac]) = \begin{cases} -1, & \text{if the task is inactive,} \\ 0, & \text{if the task is waiting,} \\ 1, & \text{if the task is active.} \end{cases}$$

The concurrent programming must solve the following problems:
— the syncronization of the execution of tasks,
— the mutual exclusion of the tasks,
— the communication between tasks.

In [1] we suggest two modes of solving these problems:
— using semaphores,
— using the concept of „rendezvous".

At last a thought about implementation. Usually standards do not say how are the concepts of a language implemented by a compiler. We think, however, that the next standard must have some features regarding implementation.

A much more detailed presentation of our opinion about improving the Fortran programming language will appear in [1].

REFERENCES

1. Boian, F., Frenţiu, M., Kása, Z., Ţâmbulea, L., *Towards a new standard Fortran*, Research Seminars on Computer Science, University of Cluj-Napoca Preprint no. 2, 1986.
2. Brainerd, W., *Fortran 77*, „Comm. A.C.M.", 21, 10 (1978) 806–810.
3. Dahl, O. — J., Dijkstra, E. W., Hoare. C.A.R., *„Structured Programming"*, Academic Press, 1972.
4. Sammet. J. E., *Roster of Programming Languages for 1974–1975*, „Comm. A.C.M.", 19, 2 (1976) 655–669.
5. * * * *American National Standard Programming Language Fortran (ANSI X3. 9 – 1978, ISO 1539–1980 (E))*, New York, 1978.

STUDIA UNIV BABEŞ-BOLYAI, MATHEMATICA, XXXII, 3, 1987

CELLULAR AUTOMATA IN GRAPHS

Z. KÁSA*

Received: May 20, 1987

Abstract. — In this paper we give a necessary and sufficient condition for a cellular automata to be cyclic, using the number of independent edge-sets in graph.

0. Introduction. A cellular automaton in a graph is a triplet $A = A(G, S, f)$, where S is a set of numbers which are associated to the vertices of the graph, f a transition function. The number associated to a vertex is called the state of this vertex. Let us denote it by $s(x)$. If $x_1, x_2, \ldots, x_n$ are the vertices of the graph then $(s(x_1), s(x_2), \ldots, s(x_n))$ is a configuration of the automaton. At discrete time steps all vertices change their state simultaneously, giving a new configuration of the automaton from the previous one.

In this paper we consider only L i n d e n m a y e r's automata [1, 4] in which $S = \{0, 1\}$ and f gives, in every vertex x, the sum modulo 2 of the numbers (states) associated to x and to its neighbours.

A cellular automaton is cyclic if every configuration of it has a predecessor. The problem is to decide if for a given graph, the corresponding automaton is or not cyclic. In [1] A n d r á s f a i has given a theorem and an algorithm to resolve this problem in the case of trees. Our approach allow us to resolve this proposed problem in some classes of graphs.

1. Definitions. Let $G = (V(G), E(G))$ be a graph without loops and multiple edges, with the vertex-set $V(G)$ and edge-set $E(G)$. Let $N(u)$ denote the set of vertex u and its neighbours:

$$N(u) = \{u\} \cup \{v \in V(G) \mid \{u, v\} \in E(G)\} = \{u\} \cup \Gamma(u), \text{ for any } u \in V(G).$$

DEFINITION 1. A cellular automaton or cellular space is a triplet $A = A(G, S, f)$, where G is a graph, S a finite set (the state alphabet), $f: C \to C$, with $C = \{s \mid s: V(G) \to S)\}$, a transition function, which gives a configuration of the automaton from the previous one.

DEFINITION 2. We say an automaton $A(G, S, f)$ Lindenmayer's if $S = \{0, 1\}$ and

$$f(s)(x) = \sum_{y \in N(x)} s(y) \pmod 2$$

Such an automaton will be denoted by $A(G)$.

DEFINITION 3. A cellular automaton $A(G)$ is cyclic if its transition function is a surjection.

* University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania

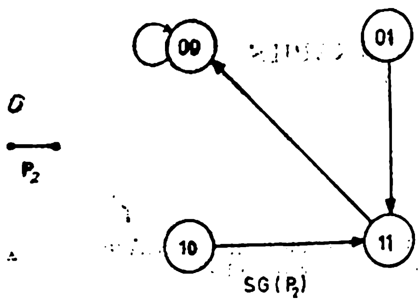

Fig. 1.

The state-graph of the automaton $A(G)$ is a directed graph $S(G)$, with configuration of the $A(G)$ as vertices, and transitions of the $A(G)$ as arcs. For $S(P_2)$, where P_2 is a path with 2 vertices, see Fig. 1. The cyclicity of an automaton may be expressed also as in:

DEFINITION 3'. A cellular automaton $A(G)$ is cyclic if its state-graph has only edges in circuits.

DEFINITION 4. A set of edges in a graph is an independent edge-set if no two edges of it are adjacents. Let us denote the number of all independent edge-sets of a given graph G by $h(G)$.

2. The main result.

THEOREM. [3] *A cellular automaton $A(G)$ is cyclic if and only if the number $h(G)$ of the independent edge-sets in the graph G is even.*

Proof. The cellular automaton $A(G)$ is cyclic in the determinant of the $A+U$ (where A is the adjacency matrix of G, U the unit matrix) is $\not\equiv 0$ (mod 2). We may compute det $(A+U)$ by the Coates' method [2]. This value is equal to the number of covers of the vertices of G by independent circuits of the oriented graph G' which has $A+U$ as adjacency matrix. The automaton is cyclic if this number is old. For every circuit in G' with more than two arcs, there exists a circuit with the same vertices, but all arcs with the opposite orientation. To every circuit in G' with only two arcs corresponds in G an edge The loops in G' have not correspondent in G. There exists a single cover of the vertices of the G' which has loops only. Therefore the number of covers by independent circuits of the graph G' has the opposite parity of the number of independent edge-set in G. This proves the theorem.

3. The number of independent edge-sets in some classes of graphs.

All following formulae may be proved by induction on n, the number of vertices in G.

3.1. Let K_n be a complete graph with n vertices ($n > 1$), then

$$h(K_{n+1}) = h(K_n) + n\{1 + h(K_{n-1})\}$$

$$h(K_2) = 1,\ h(K_3) = 3$$

It is obviously that $h(K_n)$ is odd for any n, thus $A(K_n)$ is acyclic.

3.2. Let P_n be a path with n vertices, then

$$h(P_n) = 2h(P_{n-1}) - h(P_{n-3}) \text{ or}$$

$$h(P_n) = h(P_{n-1}) + h(P_{n-2}) + 1$$

$$h(P_2) = 1,\ h(P_3) = 2,\ h(P_4) = 4$$

The former formula may be derived by computing the number of all n-digit sequences of 0 and 1, in which no two are adjacents, for the latter formula see 3.9.

$$h(P_n) = \begin{cases} \text{even, if } n \not\equiv 2 \pmod 3 \\ \text{odd, if } n \equiv 2 \pmod 3 \end{cases}$$

3.3. Let C_n be a circuit with n vertices, then

$$h(C_n) = h(P_n) + h(P_{n-2}) + 1$$

$h(C_3) = 3$, $h(C_4) = 6$ and

$$h(C_n) = \begin{cases} \text{odd, if } n \equiv 0 \pmod 3 \\ \text{even, if } n \not\equiv 0 \pmod 3 \end{cases}$$

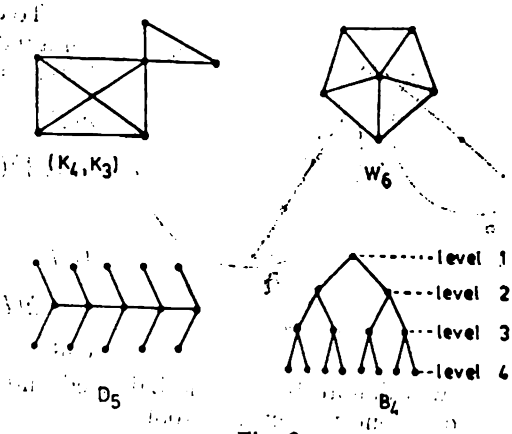

Fig. 2.

3.4. Pet S_n be a star with n vertices, then

$$h(S_n) = n - 1$$

If we complete this star-graph with edges between terminal vertices, then we obtain an wheel-graph (Fig. 2) with n vertices. For $n > 3$ we have

$$h(W_n) = (n-1)\{[h(P_{n-2}) + 1] \cdot 2 - 1\} + h(C_{n-1}) = (n - 1)\{h(P_{n-2}) + 1\} + h(C_{n-1})$$

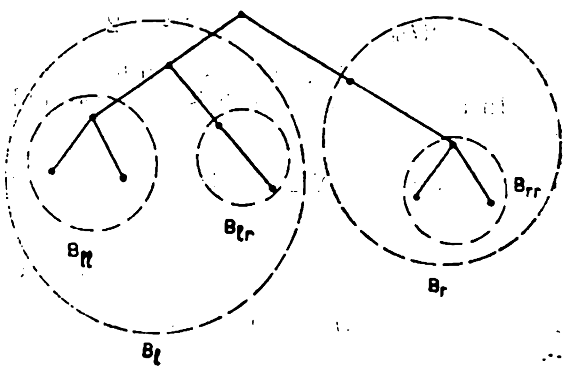

Fig. 3.

It is easy to prove that

$$h(W_n) = \begin{cases} \text{even, if } n \not\equiv 1 \pmod 3 \text{ and } n \text{ is odd} \\ \text{odd, otherwise} \end{cases}$$

If n is odd, then $h(W_n)$ and $h(C_{n-1})$ have the same parity.

3.5. Let B be a binary tree with the right subtree B_r and the left subtree B_l (Fig. 3). We denote by B_{lr} the right subtree of the left subtree, by B_{ll} the left subtree of the left subtree, and so on. Then we have:

$$h(B) = \{h(B_l) + 1\}\{h(B_r) + 1\} + \{h(B_r) + 1\}\{h(B_{ll}) + 1\}\{h(B_{lr}) + 1\} + \\ + \{h(B_l) + 1\}\{h(B_{rl}) + 1\}\{h(B_{rr}) + 1\} - 1$$

If $h(B_r)$ and $h(B_l)$ are odds, then $h(B)$ is odd too.

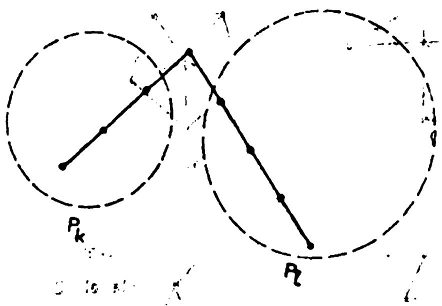

Fig. 4.

To exemplify this formula let us consider a path P_m with m vertices (fig. 4), and $k + l = m-1$. Then

$$h(P_m) = \{h(P_k)+1\}\{h(P_l)+1\} + \{h(P_l)+1\}\{h(P_{k-1})+1\} + \{h(P_k)+1\}\{h(P_{l-1})+1\} - 1$$

If $l = 2$, $k = m-3$ we obtain

$$h(P_m) = 3h(P_{m-3}) + 2h(P_{m-4}) + 4$$

which is a valid formula (see 3.2).

We will denote by B_n the balanced and complete binary tree with n levels. B_1 has only one vertex, the root, B_2 has a root and two sons, B_3 a root with two sons, each of which has exactly two sons, and so on. (Fig. 2). In this case we have

$$h(B_n) = \{h(B_{n-1})+1\}^2 + 2\{h(B_{n-1})+1\}\{h(B_{n-2})+1\}^2 - 1$$

It is easy to see that $h(B_n)$ is even for every n, thus $A(B_n)$ is cyclic.

In a t-nary tree, which is balanced and complete with n levels (each internal vertex has degree $t+1$, the root t), we have:

$$h(X_n^t) = \{h(X_{n-1}^t)+1\}^t + t\cdot\{h(X_{n-1}^t)+1\}^{t-1}\{h(X_{n-2}^t)+1\}^t - 1$$

$$h(X_n^t) = \begin{cases} \text{even}, & \text{if } t \text{ is even} \\ \text{odd}, & \text{if } t \text{ is odd} \end{cases}$$

for any n. thus $A(X_n^{2k})$ is cyclic, and $A(X_n^{2k+1})$ acyclic.

3.6. Let $K_{m,n}$ be a complete bipartite graph, then

$$h(K_{m,n}) = h(K_{m,n-1}) + m\{1 + h(K_{m-1,n-1})\}$$

$$h(K_{1,1}) = 1,\ h(K_{1,2}) = 2,\ h(K_{m,n}) = h(K_{n,m})$$

$$h(K_{m,n}) = \begin{cases} \text{odd}, & \text{if } m \text{ and } n \text{ are odds} \\ \text{even}, & \text{in other cases} \end{cases}$$

3.7. Let (K_m, K_n) be a graph formed of two complete graphs, which have in common a single vertex (Fig. 2) or more generally (G_1, G_2) a graph formed of G_1 and G_2 with a single vertex in common. Then we have

$$h(K_m, K_n) = h(K_m)\{h(K_{n-1})+1\} + h(K_n)\{h(K_{m-1})+1\} - h(K_{m-1})h(K_{n-1})$$

$h(K_m, K_n)$ is odd for all m, n except the case $m = n = 2$, $h(K_2, K_2) = h(P_3) = 2$. For two circuits in this situation we have

$$h(C_m, C_n) = h(C_m)\{h(P_{n-2})+1\} + h(C_n)\{h(P_{m-2})+1\} - h(P_{m-2})h(P_{n-2})$$

from which results that $h(C_m, C_n)$ is even if and only if $m \not\equiv 0 \pmod 3$ and $n \not\equiv 0 \pmod 3$.

3.8. Let D_n be a tree with $3n$ vertices, in which the root has three sons, from which one has three sons, each of others are terminals, and so on, except the last internal vertex, which has only two sons (Fig. 2).

$$h(D_n) = 2\, h(D_{n-1}) + h(D_{n-2}) + 3, \quad h(D_1) = 2, \; h(D_2) = 9$$

which is derived from

$$h(D_n) = 2\{h(D_{n-1}) + 1\} + h(D_{n-2}) + 1 .$$

We have

$$h(D_n) = \begin{cases} \text{even} & \text{if } n \equiv 0 \pmod 4 \text{ or } n \equiv 1 \pmod 4 \\ \text{odd} & \text{if } n \equiv 2 \pmod 4 \text{ or } n \equiv 3 \pmod 4 \end{cases}$$

3.9. Let G be a graph, $G \backslash A$ the graph obtained from G by eliminating all vertices of A with all adjacent edges, then

$$h(G) = h(G \backslash \{x\}) + \sum_{y \in \Gamma(x)} \{h(G \backslash \{x, y\}) + 1\}$$

E.g. for a path P_m $(k + l = m - 1)$, we have

$$h(P_m) = \{h(P_k) + 1\} \cdot \{h(P_l) + 1\} - 1 + \{h(P_{k-1}) + 1\} \cdot \{h(P_l) + 1\} + \\ + \{h(P_k) + 1\} \cdot \{h(P_{l-1} + 1\}$$

For $l = 2$ amd $k = m - 3$ we have

$$h(P_m) = 3h(P_{m-3}) + 2h(P_{m-4}) + 4$$

which is a valid formula for P_m.

Let $G \backslash [x, y]$ denote the graph G without the edge $\{x, y\}$, then:

$$h(G) = h(G \backslash [x, y[) + h(G \backslash \{x, y\}) + 1$$

E.g. if G is a path P , with $k + l = m$, then

$$h(P_m) = \{h(P_k) + 1\} \cdot \{h(P_l) + 1\} + \{h(P_{k-1}) + 1\} \cdot \{h(P_{l-1}) + 1\} - 1 + 1$$

For $k = m - 2$, $l = 2$ we obtain

$$h(P_m) = h(P_{m-3}) + 2\, h(P_{m-2}) + 2$$

which is anew a valid formula.

If $h\,(G \backslash [x, y])$ and $h(G \backslash \{x, y\})$ have the same parity, then $h(G)$ is odd, else is even.

A special case of the latter formula of $h(G)$ is the following. Let $(G_a - G_b)$ a graph formed of the graphs G_a and G_b which are joined by an edge $\{a, b\}$. If G_a and G_b are the trees T_a and T_b such that $d(a) = m + 1$ and $d(b) = n + 1$ ($d(x)$ is the degree of the vertex x), T_{ai} is the subtree of the T_a which is obtained from T_a by deleting the edge $\{a, i\}$ for $i \neq b$. Let us denote by $\Gamma^b(a)$ the set of adjacent vertices to a, except the vertex b.

$$h(T_a - T_b) = \{h(T_a) + 1\} \cdot \{h(T_b) + 1\} + \prod_{i \in \Gamma^b(a)} \{h(T_{ai}) + 1\} \cdot \prod_{i \in \Gamma^b(a)} \{h(T_{bi}) + 1\} - 1$$

If T_a is a path P_m with m vertices, and T_b a path P_n with n vertices, then $(T_a - T_b) = P_{m+n}$, and

$$h(P_{m+n}) = \{h(P_m) + 1\} \cdot \{h(P_n) + 1\} + \{h(P_{m-1}) + 1\} \cdot \{h(P_{n-1}) + 1\} - 1$$

For $m = 1$ we have

$$h(P_{n+1}) = h(P_n) + h(P_{n-1}) + 1$$

By adding 1 to each member of this equality, we obtain

$$h(P_{n+1}) + 1 = \{h(P_n) + 1\} + \{h(P_{n-1}) + 1\}$$

and from this we can see that $h(P_n) + 1$ is a Fibonacci number, but $h(P_2) = 1$, $h(P_3) = 2$, $F_1 = 1$, $F_2 = 2$, $F_3 = 3$, and thus $h(P_n) = F_n - 1$.

From this results that

$$h(C_n) = F_{n+1} + F_{n-1} - 1$$

3.10. Let us consider a graph as in Fig. 5. Then we have

$$h(G) = \{h(G_1) + 1\} \cdot \{h(G_2) + 1\} \cdot \{h(G_3) + 1\} - 1 + \{h(G_1') + 1\} \cdot \{h(G_2') + 1\} \cdot \{h(G_3) + 1\} + \{h(G_1') + 1\} \cdot \{h(G_3') + 1\} \cdot \{h(G_2) + 1\}$$

where G_i' for $i = 1, 2, 3$ is the graph G_i without the vertex x_i and its neighbours We can see that if $h(G_2)$ and $h(G_3)$ are odds, then $h(G)$ is odd too. If G_1 is a path P_m, G_2 a path P_n with $m = n = 2 \pmod 2$, then $h(G)$ is odd.

Let us consider a graph G as in fig. 6., then we have

$$h(G) = \{h(G_2) + 1\} \cdot \{h(P_{n-1}) + 1\} - 1 + \{h(G_1) + 1\} \cdot \{h(P_{n-2}) + 1\}.$$

If $n \equiv 0 \pmod 3$ then $h(P_{n-1})$ is odd, $h(P_{n-2})$ even, and

$$h(G) = \{h(G_2) + 1\} \cdot even - 1 + \{h(G_1) + 1\} \cdot odd$$

Therefore $h(G)$ has the same parity as $h(G_1)$.

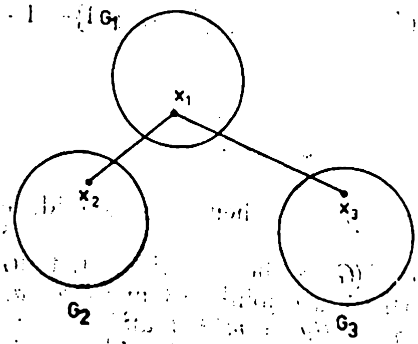

Fig. 5.

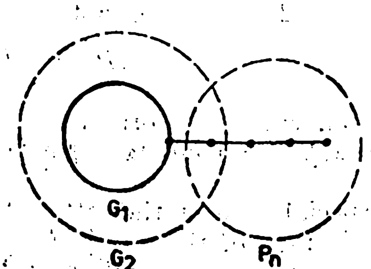

Fig. 6.

REFERENCES

1. Andrásfai, B. *Cellulular automata in trees*, Colloquia Mathematica Societatis János Bolyai, 37. „Finite and infinite sets", Eger (Hungary), North Holland, Amsterdam, 1981, pp. 35–45.
2. Coates, L. C, *Flow-graph solution of linear algebraic equations*, IRE Trans., CT–6 (1959) 170–187.
3. Kása, Z., *Cyclic cellular automata in graphs*, Research Seminars, Univ, of Cluj-Napoca, 1985.
4. Lindenmayer, A., *Mathematical models for cellular interactions in development*, „J. Theoret. Biol.", 18 (1968) Part I. 280–299, Part II. 300–315.
5. Szwerinski, H., *Symmetrical one-dimensional cellular spaces*, „Information and control", 67 (1985) 163–172.

STUDIA UNIV. BABEŞ—BOLYAI, MATHEMATICA, XXXII, 3, 1987

PRINCIPAL COMPONENTS OF A FUZZY CLASS

D. DUMITRESCU* and T. TOADERE*

Received: May 25, 1987

REZUMAT. — Componentele principale ale unei clase nuanţate. În lucrare se definesc componentele principale ale unei clase nuanţate ce descrie un nor de puncte. Se dă un algoritm pentru detectarea grupărilor liniare de puncte ce constituie substructura clasei nuanţate. Este propus un algoritm de clasificare ierarhică divizivă. Algoritmul foloseşte informaţia relativă la structura claselor nuanţate furnizată de analiza componentelor principale.

Introduction. The aim of this paper is to extend the principal component analysis for a fuzzy class and to design a hierarchical classifier. In Section 1 the principal components of a fuzzy class are defined. In Section 2 an algorithm to detect the linear cluster substructure of a fuzzy class in given. In Section 3 using principal component analysis a hierarchical classification procedure is developed. Definitions and notations from [3] and [5] are used.

1. **Principal component analysis.** Let $X = \{x_1 \ldots, x_p\}$, $x_i \in \mathbf{R}^d$, be a data set. Every $x_i \in X$ is a pattern vector. Let C be a fuzzy class which describes a cluster or cloud of points from X. We consider the shape of this cluster to be linear. Our aim is to detect the most important directions along the cluster is spread out. These directions will be called the *principal components* of the fuzzy class C. In this paper we admit the cloud may be approximated by straight lines in $\mathbf{R}^d$, but there is no dificulty to consider more general linear varieties.

Let us consider the line L through the point y_0 with the direction v:

$$L = \{y \in \mathbf{R}^d \mid y = y_0 + tv, \ t \in \mathbf{R}\}.$$

Distance $d_C(x, L)$ of $x \in X$ to L in the fuzzy class C is given by

$$d_C(x, L) = \min_{y \in L} d_C(x, y).$$

We remember that the distance in C between $x \in X$ and $y \in L$ is

$$d_C(x, y) = C(x)d(x, y) \tag{1}$$

This result may be obtained using either the local distance introduced in [3, 5] or the fuzzy points distance considered by Gerla and Volpe [8].

We may consider $d_C^2(x, L)$ as a measure of the proximity between x and the variety L. Thus the proximity $W(L, C)$ between the fuzzy class C and L may expressed as

$$W(L, C) = \sum_{j=1}^{p} d_C^2(x_j, L).$$

* University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania

The scalar product in $\mathbf{R}^n$ is

$$(x, y) = x^T M y,$$

where M is a symmetric positive definite matrix and T denotes the transposition.
We have

$$d^2(x_j, L) = \|x_j - y_0\|^2 - (x_i - y_0, v)^2. \tag{2}$$

We may thus write

$$W(L, C) = \sum_{j=1}^{p} (C(x_j))^2 (\|x_j - y_0\|^2 - (x_j - y_0, v)^2). \tag{3}$$

The search for the closest line to the cloud C leads to the minimization of $W(L\cdot, C)$. It can be proved (see [1, 2]) that the best line passes through the center of gravity, given by the mean value

$$m_C = \frac{\sum_{j=1}^{p} C(x_j) x_j}{\sum_{j=1}^{p} C(x_j)},$$

of the class C. The coordinate transformation

$$x'_j = x_j - m$$

preserves the shape of C and the new mean of the class is zero. We may thus assume that the fuzzy class C has zero mean. Therefore we may put $y_0 = 0$ and (3) becomes

$$W(L, C) = \sum_{j=1}^{p} (C(x_j))^2 \|x_j\|^2 - \sum_{j=1}^{p} (C(x_j))^2 (x_j, v)^2. \tag{4}$$

Since the first term in (4) is constant, minimizing $W(L, C)$ means to maximize the second term

$$I(v) = \sum_{j=1}^{p} (C(x_j))^2 (x_j, v)^2. \tag{5}$$

Without loos of generality we may assume $\|v\| = 1$. $I(v)$ may be written

$$I(v) = \sum_{j=1}^{p} (C(x_j))^2 v^T M x_j x_j^T M v = v^T \left(\sum_{j=1}^{p} (C(x_j))^2 M x_j x_j^T M \right) v = v^T S v, \tag{6}$$

where

$$S = \sum_{j=1}^{p} (C(x_j))^2 M x_j x_j^T M \tag{7}$$

The matrix S may be interpreted as the scatter matrix of the fuzzy class C.

Detection of the principal directions reduces to the problem

$$\begin{cases} \text{maximize} & I(v) \\ \|v\| = 1 \end{cases} \tag{8}$$

It is well known that the extremal values of the quadratic form $I(v)$ on the unit sphere are the eigenvalues λ_i corresponding to the eigenvectors of the matrix S, i.e.

$$Su_i = \lambda_i u_i, \quad i = 1 \ldots, d. \tag{9}$$

The unit eigenvectors $u_1 \ldots, u_d$ are the principal directions of the cloud C. We may assume

$$\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_d.$$

The eigenvector u_1 gives the most important direction along the cluster C is spread out. The ratio

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

represents a measure of the goodness of fit to the cloud of the first k components.

The k-dimensional subspace spanned by the first k components represents a skeleton of the cluster. This skeleton may be considered as a representation of the class. Let us denote this reprezentation by PC^k, i.e.

$$PC^k = \{u_1 \ldots, u_k\}.$$

The hierarchical classification method presented in [3, 4] and principal component analysis of a fuzzy class may cross-fertilize. In Section 3 we'll give a hierarchical classification method based on principal component analysis of a fuzzy class.

2. **Linear cluster substructure of a fuzzy class.** The cluster substructure of a fuzzy class C has been investigated in the papers [3—5]. We have considered this substructure as given by a *fuzzy partition* (see for instance [3]) $P = \{A_1 \ldots, A_n\}$ of C. Every member of P describes a cluster. In this section the case of linear cluster substructure is investigated.

We admit the fuzzy classes from P describe linear shape clusters. The prototype of every class is a line in $\mathbf{R}^d$. For the class A_i the prototype is

$$V_i(y_i, u_i) = \{y \in \mathbf{R}^d \mid y = y_i + tu_i, \ t \in \mathbf{R}\}. \tag{10}$$

The *inadequacy* $I(A_i, V_i)$ between the fuzzy class A_i and its prototype V_i is defined as

$$I(A_i, V_i) = \sum_{j=1}^{p} d^2_{A_i}(x_j, V_i) = \sum_{j=1}^{p} (A_i(x_j))^2 d^2(x_j, V_i). \tag{11}$$

The inadequacy $J(P, V)$ between the fuzzy partition P and its representation $V = \{V_1 \ldots, V_n\}$ is

$$J(P, V) = \sum_{i=1}^{n} I(A_i, V_i) = \sum_{i=1}^{n} \sum_{j=1}^{p} (A_i(x_j))^2 (\|x_j - y_i\|^2 - (x_j - y_i, u)^2). \quad (12)$$

A local minimum of the criterion funtion J may be obtained [4] by the iterative procedure with

$$A_i(x_j) = \frac{C(x_j)}{\sum_{k=1}^{n} \frac{d^2(x_j, V_i)}{d^2(x_j, V_k)}}, \quad 1 \leqslant i \leqslant n, \quad 1 \leqslant j \leqslant p, \quad (13)$$

$$y_i = \frac{\sum_{j=1}^{p} (A_i(x_j))^2 x_j}{\sum_{j=1}^{p} (A_i(x_j))^2}, \quad 1 \leqslant i \leqslant n, \quad (14)$$

u_i = the unit eigenvector corresponding to the largest eigenvalue of the matrix

$$S_i = \sum_{j=1}^{p} (A_i(x_j))^2 M(x_j - v_i)(x_j - v_i)^T M. \quad (15)$$

This iterative procedure may be called Generalized Fuzzy Lines (G F L) algorithm. For $C = X$, G F L reduces to Fuzzy n-lines algoritm [1].

G F L represents an useful tool to detect the linear cluster substructure of a fuzzy class. This procedure may be used to desing a hierarchical binary divisive classifier [3]. In this paper we propose another hierarchical classification method in which principal component analysis of a fuzzy class is used.

3. Principal components and hierarchical classification. In this section a divisive hierarchical clustering procedure is proposed. The information given by the principal component analysis is used to obtain a fuzzy partition of a fuzzy class. The procedure may be viewed as a classification method as well as a dimensionality reduction techique.

Let k_1, $k_1 \geqslant 2$, be the number of selected principal components of X. Since the best components are choosen we may assume that there are k_1 linear shape clusters X. In order to obtain the fuzzy classes describing these clusters the G F L algorithm for $C = X$ and $n = k_1$ is used. A fuzzy partition P^1 of X with k_1 classes and the corresponding line prototypes $V_1 \ldots, V_{k_1}$ are thus obtained.

Let us remark that the directions $u_1, \ldots, u_k$ of the line protoypes of a fuzzy class C are not necessarily orthogonal. For this resson $V = \{V_1 \ldots, V_k\}$ is a representation of C more suitable than the representation PC^k of the orthogonal principal components.

The degree in which the class prototypes are closed to the principal components represents the certainty degree that exactly k_1 linear clusters are present in the data set. In order to estimate the distance between prototypes directions u_i' s and principal components v_i' s; the classes may be renumbered such that

$$\cos(u_i, v_i) = \min_{j=1,\ldots,k_1} \cos(u_j, v_i).$$

We may consider 1-$\cos(u_i, v_i)$ as the distance between directions u_i and v_i. "Distance" between the representation V and principal component representation PC^{k_1} may be expressed as

$$d(V, PC^{k_1}) = \sum_{i=1}^{k_1} (1\text{-}\cos(u_i, v_i)).$$

If $d(V, PC^{k_1})$ excedes an approriate prescribed thereshold D, then we may search for a more satisfactory value of the cluster number. We may conjecture that the optimal cluster number is in the neighbourhood of k_1. If a modification of k_1 improves d, the new value of k_1 will be selected. Ideally, we search for a number k_1 such that $d(V, PC^k)$ becomes minimum for $k = k_1$, i.e.

$$d(V, PC^{k_1}) = \min_k d(V, PC^k).$$

In order to construct a fuzzy hierarchy [3, 4] we consider the fuzzy partition $P^1 = \{A_1 \ldots, A_{k_1}\}$ as the first level classification partition. For every class C of P^1 we apply the same procedure as for X. The principal components and the corresponding fuzzy partition of C are therefore computed. The obtained fuzzy classes will be atoms in the fuzzy partition P^2 at the second classification level. If for an atom C only one principal component is selected then C remains unchanged. It is an atom in all subsequent fuzzy partitions P^l, $l \geqslant 2$, and represents a terminal cluster.

This decomposition process continues until the level l for which P contains only terminal undivisible clusters. If a new level is considered then $P^{l+1} = P^l$. This procedure induces a chain of fuzzy partitions ordered by the refinement relation [3, 5]. This chain generates a fuzzy hierarchy [5].

REFERENCES

1. Bezdek, J. C., Coray, C., Gunderson, R., Watson, J., *Detection and characterization of cluster substructure*, SIAM J. Appl. Math. 40 (1981), 339–371.
2. Duda, R. O., Hart, P.E., *Pattern classification and scene analysis*, Wiley — Interscience, New York, 1974.
3. Dumitrescu, D., *Hierarchical pattern classification*, Int. J. Fuzzy Sets and Systems, to appear.
4. Dumitrescu, D., *Hierarchical detection of linear cluster substructure*, Univ. of Cluj-Napoca, Fac. Math. Res. Seminars 2 (1986) 21–28.
5. Dumitrescu, D., *Numerical methods in fuzzy hierarchical pattern recognition*, Studia Univ. Babeş–Bolyai, I, 4 (1986), 31–36, II, 1(1987), 26–30.
6. Dumitrescu, D., *Hierarchical classification with fuzzy sets*, Univ. of Cluj-Napoca, Fac. Math. Res. Seminars, 4 (1984) 36–55.
7. Dumitrescu, D., Toadere, T., *Asupra unei metode ierarhice în recunoaşterea formelor*, Simp. Naţional de Teoria Sistemelor, Craiova, 1986, I, 358–362.
8. Gerla, G., Volpe, R., *The definition of distance and diameter in fuzzy set theory*, Studia Babeş–Bolyai, 3 (1986), 21–26.

STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXII, 3, 1987

REVERSIBLE EXECUTION WITH LOOP-EXIT SCHEMES

FLORIAN MIRCEA BOIAN*

Received: May 30, 1987

REZUMAT. — Execuţia reversibilă prin scheme Loop-Exit. În lucrare se definesc mai intii noţiunile de ramură şi secţiune asociate unei scheme Loop-Exit [3, 5]. Apoi se prezintă modul în care sint ele folosite la execuţia reversibilă [9, 10, 1, 8, 6].

1. **Introduction.** Reversible execution is an attractive method for run-time debugging programs. It was introduced by Zelkovitz [9, 10]. A relative efficient implementation was done by Davis [6]. Many systems using the reversible execution method are presented by *Johnson* [8].

All these systems use a stack called the HISTORY stack. During the execution, all modification over the variables are pushed in HISTORY. Hence a very large amount of memory is necessary for this stack.

In [1] a mathematical model for reversible execution is presented. There, Aguzzi uses also a stack for its model. The Aguzzi's model is theoretically applied to all recursive and bijective functions.

In this paper, a model for reversible execution to all flowchartable algorithms [7] is presented. The principal advantage of this model is a drastic reduction of the amount of memory for HISTORY stack.

Suppose that the algorithms are described using the Loop-Exit Schemes [5], and that these schemes are reduced [3]. Also, we suppose that the formal definitions for **AM** (the Assignment Marks), **TM** (the Test Marks), the Loop-Exit Scheme (for short *LES*) are known. See the definition 2 from [5] for details. If S is a *LES*, then we suppose that the definition of G_S (the context-free grammar associated to S), $L(S)$ (the language generated from G_S), and the static word associated to S, are known too. For details, see the definitions 5 and 6 from [5].

Now, we illustrate these notions by an example.

EXAMPLE 1. Let S be the following *LES*:

LOOP$_1$
  a_1 ; a_2 ;
  IF$_1$ a_3 THEN$_1$ EXIT$_1$; ENDIF$_1$;
  a_4 ;
ENDLOOP$_1$;
LOOP$_2$
  IF$_2$ a_5 THEN$_2$ a_6 ; ELSE$_2$ a_7 ; EXIT$_2$; ENDIF$_2$;
  IF$_3$ a_8 THEN$_3$ a_9 ; ELSE$_3$ a_{10} ; ENDIF$_3$;
  a_{11} ;
ENDLOOP$_2$;

* *University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania*

For S, we have:

$$\mathbf{AM} = \{a_1, a_2, a_4, a_6, a_7, a_9, a_{10}, a_{11}\};$$
$$\mathbf{TM} = \{a_3, a_5, a_8\};$$

The static word is: „$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}$".
The productions of G_S are:

$$\nabla \to L_1L_2$$
$$L_1 \to a_1a_2I_1a_4L_1 \mid a_1a_2a_3 + \quad B_1 \to a_1a_2I_1a_4B_1 \mid \varepsilon$$
$$I_1 \to a_3 -$$
$$L_2 \to I_2I_3a_{11}L_2 \mid a_5 - a_7 \quad B_2 \to I_2I_3a_{11}B_2 \mid \varepsilon$$
$$I_2 \to a_5 + a_6$$
$$I_3 \to a_8 + a_9 \mid a_8 - a_{10}$$

The words from $L(S)$ are the following;

$$(a_1a_2a_3 - a_4)^*a_1a_2a_3 + (a_5 + a_6(a_8 + a_9 \mid a_8 - a_{10})a_{11})^*a_5 - a_7$$

REMARKS: a) We use the notation from [2]: $(\alpha)^*$, $(\alpha)^+$ and $(\alpha \mid \beta)$ for the sets of words:

$$(\alpha)^* = \{\alpha\alpha \dots \alpha = \alpha^n \mid n \geqslant 0\};$$
$$(\alpha)^+ = \{\alpha\alpha \dots \alpha = \alpha^n \mid n \geqslant 1\};$$
$$(\alpha \mid \beta) = \{\alpha, \beta\}.$$

b) We use the notation „a_iX_i", where $X_i \in \{+, -\}$ and $a_i \in \mathbf{TM}$, and „a_i" if $a_i \in \mathbf{AM}$. The „+" is used for a THEN alternative and „−" for an ELSE alternative.

2. **Branches and sections.** In the following, we suppose that S is a *LES* having the static word „$a_1a_2 \dots a_n$".

DEFINITION 1. A word $z = a_{i_1}X_{i_1}a_{i_2}X_{i_2} \dots a_{i_s}X_{i_s}$ is a section for S iff there is $w \in L(S)$ such that:

a) $w = xyz$;
b) $i_j < i_{j+1}$ for $j = 1, 2, \dots s - 1$;
c) if $x \neq \varepsilon$ then $x = x'a_{i_0}X_{i_0}$ with $i_0 \geqslant i_1$;
d) if $y \neq \varepsilon$ then $y = a_{i_{s+1}}X_{i_{s+1}}y'$ with $i_s \geqslant i_{s+1}$.

We denote by SEC(S) the set of all sections from S.

DEFINITION 2. A word $z \in$ SEC(S) is a branch for S iff there is $w \in L(S)$ such that $w = zy$.
We denote by BRA (S) the set of all branches from S.

The following theorems establish some simple properties of $BRA(S)$ and SEC (S).

THEOREM 1. *BRA* (S) $\subset$ SEC(S), *and* $|\text{SEC}(S)| < 2^n$.

Proof. From the definitions 1 and 2 it is obvious that BRA (S) $\subset$ SEC (S). Now, for each k, $1 \leqslant k \leqslant n$ there are at most $\binom{n}{k}$ sections having k symbols from $\mathbf{AM} \cup \mathbf{TM}$. Hence,

$$|\text{SEC}(S)| < \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{n} = 2^n - 1 < 2^n \text{ Q.E.D.}$$

THEOREM 2. $L(S) \subset (\mathrm{SEC}(S))^+$

Proof: Let $w \in L(S)$, $w = a_{i_1}X_{i_1} \ldots a_{i_p}X_{i_p}$. If $i_j < i_{j+1}$ for each j, $1 \leq j \leq p-1$, then $w \in \mathrm{SEC}(S)$ and the theorem holds. Otherwise, there are r integers j_k, $1 \leq j_1 \leq j_2 \leq \ldots \leq j_r \leq p$ such that $i_{j_k} \geq i_{j_k+1}$, $k = 1, r$. If r is maximal, we denote $y_1 = a_{i_1}X_{i_1} \ldots a_{i_{j_1}}X_{i_{j_1}}$,

$$y_2 = a_{i_{j_1+1}}X_{i_{j_1+1}} \ldots a_{i_{j_2}}X_{i_{j_2}},$$

$$- - - - - - - - - - - - - - - - -$$

$$y_r = a_{i_{j_{r-1}+1}}X_{i_{j_{r-1}+1}} \ldots a_{i_{j_r}}X_{i_{j_r}} \text{ and}$$

$$y_{r+1} = a_{i_{j_r+1}}X_{i_{j_r+1}} \ldots a_{i_{j_p}}X_{i_{j_p}}.$$

It is obvious that $w = y_1y_2 \ldots y_ry_{r+1}$ and $y_k \in \mathrm{SEC}(S)$, $k = 1, 2, \ldots r+1$. Therefore $w \in (\mathrm{SEC}(S))^+$, and the theorem holds. Q.E.D.

Now, using the definitions 1, 2, and the theorems 1, 2, we can easily prove the following two theorems:

THEOREM 3. *For each* $w \in L(S)$ *there is* $z \in \mathrm{BRA}(S)$ *such that* $w = zy$.

THEOREM 4. *For each* $z \in \mathrm{SEC}(S)$ *there is* $y \in \mathrm{BRA}(S)$ *such that* $y = xz$.

Intuitively, a section is a maximal sequence of statements of S such that their order of execution is the same with their order in the text of program. In [4] we have shown how to use the branches to uncover the unitialized variables, and how to use the sections for testing and correcting programs.

3. **Algorithms** for obtaining the BRA(S) and SEC(S). Let S be a *LES*. Suppose that $a_1a_2 \ldots a_n$ is its static word.

ALGORITHM 1. Construction of the BRA (S).

Input: The context-free grammar G_S.

Output: The set BRA (S).

Step 1: A grammar G_1 is constructed from G_S as follows: all the $B_k \rightarrow \alpha B_k \mid \varepsilon$ productions from G_S are erased, and in the other productions B_k is replaced by ε.

Step 2: Using the algorithms from [2] for the elimination of the inaccessible and useless symbols, the G_2 grammar is constructed from G_1.

Step 3: A grammar G_3 is constructed from G_2 as follows: all the $L_k \rightarrow \alpha L_k$ productions from G_2 are replaced by productions $L'_k \rightarrow \alpha$, where L'_k is a new symbol, associated to L_k.

Step 4: We construct a grammar G_4 from G_3 adding to the productions of G_3 new productions. For each production $A \rightarrow \alpha L_k \beta$ of G_3, with L_k recursive in G_2, a production $A \rightarrow \alpha L'_k$ is added, where L'_k is the symbol associated to L_k in the step 3.

Step 5: Put $\mathrm{BRA}(S) = L(G_4)$.

EXAMPLE 2. Let us consider S from the example 1. After applying the steps 1—4 from the algorithm 1, G_4 has the productions:

$\nabla \rightarrow L_1L_2 \mid L'_1 \mid L_1L'_2$	$L_2 \rightarrow a_5 - a_7$
$L_1 \rightarrow a_1a_2a_3 +$	$L'_2 \rightarrow I_2I_3a_{11}$.
$L'_1 \rightarrow a_1a_2I_1a_4$	$I_2 \rightarrow a_5 + a_6$
$I_1 \rightarrow a_3 -$	$I_3 \rightarrow a_8 + a_9 \mid a_8 - a_{10}$

After applying the step 5, we obtan:

$$\text{BRA}(S) = \{a_1a_2a_3 + a_5 - a_7, \quad a_1a_2a_3 - a_4, \quad a_1a_2a_3 + a_5 + a_6a_8 + a_9a_{11}, \quad a_1a_2a_3 + a_5 + a_6a_8 - a_{10}a_{11}\}$$

THEOREM 5. *Using the algorithm* 1, *the set* BRA(S) *is obtained.*

Proof: From the definition 5 of [5] it results that in $L(S)$ the static order is modified only by $L_k \to \alpha L_k$ or $B_k \to \alpha B_k$ productions. Therefore, if it exists a derivation

$$\nabla \overset{*}{\underset{G_S}{\Longrightarrow}} u a_i X_i \, a_j X_j \delta \quad \text{with} \quad i \geqslant j,$$

then there exists a production $R \to \alpha R$ such that:

$$\nabla \overset{*}{\underset{G_S}{\Longrightarrow}} \beta_1\beta_2 R\delta_2\delta_1 \underset{G_S}{\Longrightarrow} \beta_1\beta_2 \alpha R\delta_2\delta_1, \text{ and}$$

$$\beta_1\beta_2\alpha \overset{*}{\underset{G_S}{\Longrightarrow}} u a_i X_i \text{ and } R\delta_2\delta_1 \overset{*}{\underset{G_S}{\Longrightarrow}} a_j X_j \delta.$$

Let $w \in L(S)$ and $z \in \text{BRA}(S)$ obtained using the theorem 3. If $w = z$ then in $\nabla \overset{*}{\underset{G_S}{\Longrightarrow}} w$ is not used any production $R \to \alpha R$, hence $\nabla \overset{*}{\underset{G_S}{\Longrightarrow}} w = z \quad z \in L(S_4)$.

Now, if $w = zy$, with $y \neq \varepsilon$, then we have:

$$\nabla \overset{*}{\underset{G_S}{\Longrightarrow}} u_1 R\delta \underset{G_S}{\Longrightarrow} u_1 \alpha R\delta \overset{*}{\underset{G_S}{\Longrightarrow}} zy = w$$

We can suppose that only left-derivations [2] are used and $R \to \alpha R$ is the first recursive production applied. Therefore we have:

$$u_1\alpha \overset{*}{\underset{G_S}{\Longrightarrow}} u_1u_2a_iX_i = z \quad \text{and} \quad R\delta \overset{*}{\underset{G_S}{\Longrightarrow}} a_jX_jy_1 = y \quad \text{and} \quad i \geqslant j.$$

Hence, $z \in L(G_4)$ too, and $\text{BRA}(S) \subset L(G_4)$.

Analogously, going backwards, one may show that $L(G_4) \subset \text{BRA}(S)$ Q.E.D.

Let S be a *LES* and L_k a recursive symbol of G'_S.

DEFINITION 3. The grammar G_S^k associated to L_k is obtained from G_S as follows:

a) We erase all the $\nabla \to \gamma$ productions from G_S if the L_k symbol does not appear in γ.

b) Each $A \to \alpha L_k \beta$ with $A \neq L_k$ from G_S is replaced by the productions $A \to \alpha L_k \beta$ and $\nabla \to A$.

We denote by BRA(k) the results of applying the algorithm 1 to the G_S^k grammar.

EXAMPLE 3. Let us consider G_S from the example 1. The symbols L_1 and L_2 are recursive. Then $G_S^1 = G_S$ and G_S^2 has the following productions:

$\nabla \to L_2$
$L_1 \to a_1a_2I_1a_4L_1 \mid a_1a_2a_3 + \quad B_1 \to a_1a_2I_1a_4B_1 \mid \varepsilon$
$I_1 \to a_3 -$
$L_2 \to I_2I_3a_{11}L_2 \mid a_5 - a_7 \quad B_2 \to I_2I_3a_{11}B_2 \mid \varepsilon$
$I_2 \to a_5 + a_6$
$I_3 \to a_8 + a_9 \mid a_8 - a_{10}$

We observe that L_1, B_1, B_2, a_1, a_2, a_3 and a_4 are inaccessible symbols. After applying the algorithm 1 to G_S^2, we obtain:

$$\text{BRA}(2) = \{a_5 + a_6a_8 + a_9a_{11},\ a_5 + a_6a_8 - a_{10}a_{11},\ a_5 - a_7\}.$$

Now, the following theorem offers a method for obtaining SEC(S).

THEOREM 6. *The following relation holds*:

$$\text{SEC}(S) = \text{BRA}(S) \cup \{\text{BRA}(k) \mid L_k \text{ is recursive in } G_S'\}.$$

Proof: If G_S' does not contain any recursive symbol then the theorem obviosuly holds.

Suppose that there exists a recursive symbol L_k in G_S'. Then in G_S' there are the productions $A \to \alpha L_k \beta$ and $L_k \to \gamma L_k$. In G_S' we have $\nabla \overset{*}{\Longrightarrow} a_iX_i \ldots$ $\ldots a_jX_j$ with $j \geqslant i$ and

$$\nabla \overset{*}{\Longrightarrow} \alpha' A \beta' \Longrightarrow \alpha' \alpha L_k \beta\beta' \overset{3}{\Longrightarrow} \alpha' \alpha \gamma^3 L_k \beta\beta' \overset{*}{\Longrightarrow}$$

$$\overset{*}{\Longrightarrow} \alpha' \alpha a_iX_i \ldots a_jX_j a_iX_i \ldots a_jX_j a_iX_i \ldots a_jX_j L_k \beta\beta'.$$

It results that $a_iX_i \ldots a_jX_j \in \text{SEC}(S)$.

But in G_S^k we have

$$\nabla \overset{*}{\Longrightarrow} A\beta' \Longrightarrow L_k\beta\beta' \overset{2}{\Longrightarrow} \gamma^2 L_k \beta\beta' \overset{*}{\Longrightarrow} a_iX_i \ldots a_jX_j a_iX_i \ldots a_jX_j L_k \beta\beta'.$$

It results that $a_iX_i \ldots a_jX_j \in \text{BRA}(k)$.

For each $z \in \text{BRA}(k)$ these derivations hold in both G_S' and G_S^k.

Conversely, for each $z \in \text{SEC}(S) - \text{BRA}(S)$ it results that there is a recursive symbol L_k in G_S' such that these derivation hold in both G_S' and G_S^k.

By induction on the number of the L_k recursive symbols from G_S', it can be proved that the theorem holds. Q.E.D.

The theorem 6 offers a simple method for constructing the set SEC(S) for any *LES* S. At the Computer Center of Cluj-Napoca University, a PASCAL program for constructing SEC(S) using this method was designed.

The theorem 4 offers a simple method to memorize the set SEC(S) using only BRA(S). The BRA(S) can be memorized using a binary tree, as in the following example.

EXAMPLE 4. For the *LES* of the example 1 we have:

$$\text{SEC}(S) = \{a_1a_2a_3 + a_5 - a_7,\ a_1a_2a_3 - a_4,\ a_1a_2a_3 + a_5 + a_6a_8 + a_9a_{11},\ a_1a_2a_3 + a_5a_6a_8 - a_{10}a_{11},\ a_5 + a_6a_8 + a_9a_{11},\ a_5 + a_6a_8 - a_{10}a_{11},\ a_5 - a_7\}.$$

The BRA(S) is shown in the fig. 1. The numbers associated to leafs are the numbers of branches. For memorizing $z \in \text{SEC}(S)$ it is sufficient to memorize the first symbol from z and the number of its branches. In our example, the following seven pairs define SEC(S):

$$(a_1, 2),\ (a_1, 1),\ (a_1, 4),\ (a_1, 3),\ (a_5, 4),\ (a_5, 3),\ (a_5, 2).$$

4. **Using the SEC(S) for reversibile execution.** Analogously with [7], any *LES* can be extended to a program schemata as follows:

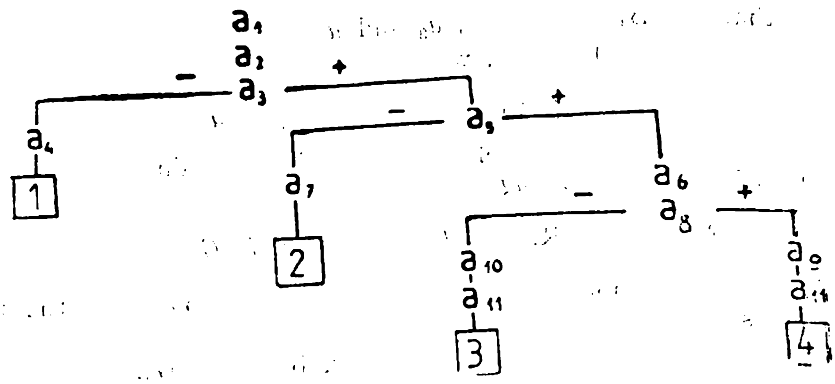

Fig. 1.

Let $\mathcal{V} = \{v_1, v_2, \ldots, v_q\}$ be a set of „variables". For a particular programming language, simple variables, items of arrays, fields of records and so on, are in $\mathcal{V}$. Let $\mathcal{F} = \{f_1, \ldots, f_r\}$ be a set of functional symbols and $\mathcal{T}$ be a set of test symbols.

For each $a_i \in \mathbf{TM}$, we have $a_i = "t(v_{i_1}, \ldots, v_{i_{n_t}})"$, with $t \in \mathcal{T}$, $n_t \geq 0$, $v_{i_j} \in \mathcal{V}$, $j = \overline{1, n_t}$. Semantically, a_i means the application of the test t to the variables v_{i_j}.

For each $a_i \in \mathbf{AM}$, we have $a_i = "v := f(v_{i_1}, \ldots, v_{i_{n_f}})"$ with $f \in \mathcal{F}$, $n_f \geq 0$, $v_{i_j} \in \mathcal{V}$, $j = \overline{1, n_f}$. Semanticaly, a_i is a assignement statement.

For reversible execution, it is sufficient to push in the HISTORY stack, at run time, the execution order of the symbols $a_i \in \mathbf{AM} \cup \mathbf{TM}$ and the changed values of the variables.

Our method replaces the order of the symbols by the order of sections from the program. More, if in a section a variable changes its values many times, in the HISTORY only a change is pushed.

During the execution, for each $S \in \mathrm{SEC}(S)$, a record as that of the fig. 2 is pushed in HISTORY.

Suppose that $z \in \mathrm{SEC}(S)$, $y \in \mathrm{BRA}(S)$, $y = xz$. If $z = a_i z'$ and b is the number of the branch y, then in the fig. 2 we have:

I contains the value i;
B contains the value b;
N contains the size of z $(N = |z|)$;
P contains the number of the pair ADDRESS — VALUE from the record;

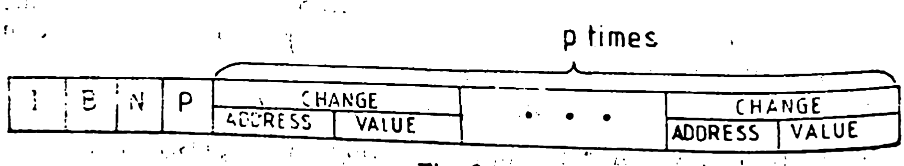

Fig. 2.

ADDRESS contains the address of the variable if this variable changes its value during the execution of z;

VALUE contains the value of the variable before the execution of z.

The following algorithm writes in the HISTORY stack.

ALGORITHM 2. (run-time)

CASE moment OF

WHEN start a section

$I := i;\ N := 0;\ P := 0;$

WHEN after execution of a test

$N := N + 1;$

WHEN before the execution of an assignment $v := f(\ldots)$

$N := N + 1;$

IF address of v is not in the record AND the old value of v is not equal to $f(\ldots)$

THEN

$P := P + 1;$

ADDRESS := the address of v;

VALUE := the old value of v;

END IF;

WHEN finish the section

$B := b;$

push the record in the HISTORY;

END CASE;

As compared with the methods from [6] and [9] our method for memorizing in HISTORY has the following two advantages:

a) Are memorized only start and finish of the section; the order of execution of statements is the order of the statements in the section.

b) Each variable appears in HISTORY at most once, and only if its value is changed in the section.

To answer the question: „how much to apply reversible execution?", two practical possibilities may be used:

a) Apply reversible execution until the latest changed value for a certain variable;

b) Apply reversible execution for the latest „n" statements. In the following algorithm, we choose this criterion for stoping the reversible execution.

ALGORITHM 3. (reversible execution)

WHILE $n > 0$ LOOP

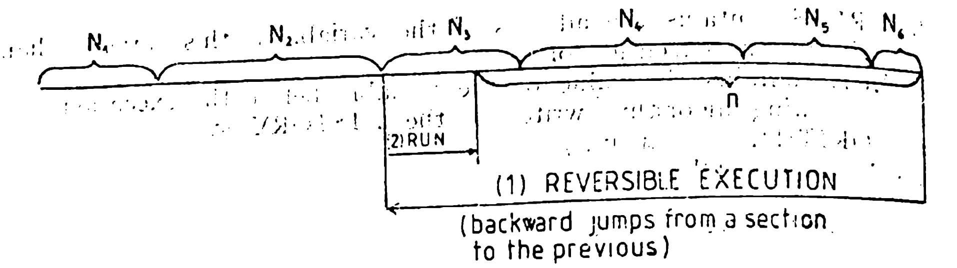

Fig. 3.

FOR each P pair ADDRESS—VALUE from the current HISTORY record LOOP

put the VALUE into the ADDRESS location;

END LOOP;
$n := n - N$;

pop the next record from the HISTORY;

END LOOP;
Execute $|n|$ statements, strating with the statement a_l;

EXAMPLE 5. În fig. 3, an example of reversible execution is presented. By $N_1, N_2, \ldots, N_6$ we denote the sizes (the number of statements and tests) for the first six execution sections. If the latest statement is in the third section, then $N_3 + N_4 + N_5 + N_6 - n$ statements from the third section must be executed after four jumps at the begining of sections.

This method is applied to the INTADA system [4], a system having six languages, INTADA/*k*, for teaching computer programming.

REFERENCES

1. Aguzzi G., *The Theory of Invertible Algorithms*. RAIRO Inform. theor., 15, no. 3 (1981) pp. 253—279.
2. Aho A. V., Ullman J. D., *The Theory of Parsing, Translation and Compiling*. Prentice—Hall, Englewood—Cliffs, New—Jersey, 1972.
3. Boian F. M., *Reducing the Loop—Exit Schemes*. Mathematica (Cluj) 28 (51), no 1 (1986), pp. 1—7.
4. Boian F.M., *Sisteme conversaționale pentru instruire în programare*. Doct. thesis, Cluj-Napoca Univ., Faculty of Mathematics, 1986.
5. Boian F.M., *Loop—Exit Schemes and Grammars; Properties, Flowchartablies*, Studia Univ. „Babeş—Bolyai". Math. 3, (1986), pp. 52—57.
6. Davis A.M., *An Interactive Analysis System for Execution-time Errors*. Tech. Rept. Univ. Illinois, Dep. Comp. Sci. UIUCDCR—R—75—695, 1975.
7. Greibach S. *The Theory of Program Structures; Schemes, Semantics, Verification*. Lect. Notes Comp. Sci. 36 (1975) Springer—Verlag.
8. Johnson M.S. *An Annotated Software Debuging Bibliography*. Tech. Note Series, Hewlett—Packard, CSL—82—4, 1982.
9. Zelkovitz M. V., *Reversible execution as a Diagnostic Tool (Preliminary Draft)* Tech. Rept. Cornell Univ. Dep. Comp. Sci. no 71—92, 1971.
10. Zelkovitz M. V., *Reversible execution*. Comm. ACM 16, no 9 (1973), pp. 566.

STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXII, 3, 1987

AN ALGORITHM CORRESPONDING TO THE METHOD OF CHORDS IN FRÉCHET SPACES

SEVER GROZE*

Received: June 1, 1987

1. Let be the operator equation

$$P(x) = \theta \tag{1}$$

where $P : X \to Y$ is a nonlinear continous mapping on the Fréchet space X in the Frechet space Y, θ being the null element of the space Y.

To approximate the solution of the equation (1) we shall use the algorithm

$$x_{n+1} = x_n - \Lambda_n P(x_n), \quad (n = 0, 1, 2, \ldots) \tag{2}$$

where $x_0, x_{-1} \in X$ are given elements, and $\Lambda_n = [x_n, x_{n-1}; P]^{-1}$.

In the papers [1], [2] sufficient conditions for the convergence of the sequence (x_n) generated by (2) are given, the limit of the sequence being a solution of equation (1). These conditions are not so restrictive than these given in the paper [3].

In the present paper some existence and uniqueness theorems, in weaker conditions than in the previous paper, are proved.

2. Let $P : X \to Y$ be a nonlinear operator which has first ordered divided differences [4], only.

We denote by $)|\cdot|(: X \to R_+$ the quasinorm induced by an invariant distance $d : X \times X \to R_+$, i.e. $d(x, y) = d(x - y, 0)$ and $)|x|(= d(x, 0)$ [4].

Now, we prove the

THEOREM 1. *Suppose that the following conditions are satisfied:*

1°. *For some initial approximations* $x_0, x_{-1} \in S \subset X$ *the mapping*

$$\Lambda_0 = [x_0, x_{-1}; P]^{-1} \text{ exists and })|\Lambda_0|(\leqslant B_0;$$

2°. *There exist* η_0 *and* η_{-1} *such that*

$$)|x_0 - x_{-1}|(\leqslant \eta_{-1} \text{ and })|\Lambda_0 P(x)|(\leqslant \eta_0, \ \eta_0 \leqslant \eta_{-1};$$

3°. *There exist* $K > 0$ *such that for every* $x', x'', x''' \in S(x_0, 2\eta_0)$ *we have*

$$)|[x', x''; P] - [x'', x'''; P]|(\leqslant K)|x' - x'''|(;$$

4°. $h_0 := B_0 K(\eta_0 + \eta_{-1}) < \frac{1}{4}$.

* University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania

Then the equation (1) has in S a solution x^*, which is the limit of the sequence (2), the convergence order being given by the inequality

$$\|x_n - x^*\| \leqslant 2^{1-s_n}\left(\frac{8}{9}\right)^{s_n - 1}(4h_0)^{s_n}\eta_0 \tag{3}$$

where s_n is the general term of the sequence of the partial sums of a Fibonacci sequence u_n, with $u_1 = u_2 = 1$.

Proof. From the condition 1°, 2° and the relation (2), we have

$$\|x_1 - x_0\| \leqslant \eta_0 < 2\eta_0 \tag{4}$$

so $x_1 \in S$, and

$$\|x_1 - x_{-1}\| \leqslant \eta_0 + \eta_{-1}. \tag{5}$$

According to the definition of the generalized divided differences and the algorithm (2), we have

$$x_1 = x_{-1} - \Lambda_0 P(x_{-1}). \tag{6}$$

Now we show that (x_1, x_0) also satisfies the hypotheses of theorem 1.

a) Let us consider the operator

$$I - \Lambda_0[x_1, x_0; P] = \Lambda_0([x_0, x_{-1}; P] - [x_1, x_0; P]).$$

Taking account of the condition 3° and the relation (4), we can write

$$\|I - \Lambda_0[x_1, x_0; P]\| \leqslant B_0K(\eta_0 + \eta_{-1}) = h_0 < \frac{1}{4} < 1$$

and from the Banach's theorem, it follows the existence of the operator

$$H = (I - (I - \Lambda_0[x_1, x_0; P]))^{-1} = (\Lambda_0[x_1, x_0; P])^{-1}$$

and $\|H\| \leqslant \frac{1}{1 - h_0}$.

Because

$$H\Lambda_0 = [x_1, x_0; P]^{-1} = \Lambda_1$$

it results the existence of Λ_1, for which we have

$$\|\Lambda_1\| \leqslant \frac{B_0}{1 - h_0} \equiv B_1 > B_0$$

so, the condition 1° is satisfies.

b) To prove that condition 2° holds, we consider the equality

$$P(x_1) - (P(x_{-1}) + [x_0, x_{-1}; P](x_1 - x_{-1})) =$$
$$= ([x_1, x_{-1}; P] - [x_0, x_{-1}; P])(x_1 - x_{-1})$$

which, using the condition (6), may be written

$$\|\Lambda_0 P(x_1)\| \leqslant B_0K(\eta_0 + \eta_{-1})\eta_0 = h_0\eta_0.$$

It results

$$\|\Lambda_1 P(x_1)\| = \|H\Lambda_0 P(x_1)\| \leqslant \|H\| \cdot \|\Lambda_0 P(x_1)\| \leqslant \frac{h_0}{1-h_0}\eta_0 = \eta_1, \ \eta_1 < \eta_0.$$

c) The hypothesis 3° is evidently verified, $x_1, x_0, x_{-1} \in S$.
d) For the hypothesis 4°, we have

$$h_1 = B_1 H(\eta_1 + \eta_0)$$

which, taking account of B_1 and η_1, leads to

$$h_1 = \frac{B_0}{1-h_0} K\left(\frac{h_0}{1-h_0}\eta_0 + \eta_0\right) = \frac{1}{(1-h_0)^2} \cdot \frac{h_0^2}{2} + \frac{1}{2}\frac{h_0}{1-h_0} \leqslant \frac{8}{9} h_0 < \frac{1}{4}$$

By induction, it can be proved that the properties 1°—4° are verified for any x_n given by the iterative method (2) and that the following relations take place:

$$B_n = \frac{B_{n-1}}{1-h_{n-1}} \tag{7}$$

$$\eta_n = \frac{h_{n-1}\eta_{n-1}}{1-h_{n-1}} \tag{8}$$

$$h_n = \frac{h_{n-1}h_{n-2}}{(1-h_{n-1})^2}. \tag{9}$$

Using (9), it follows, for $n \geqslant 2$,

$$h_n \leqslant 2h_{n-1}h_{n-2}$$

relation which allowes the following evaluations for h_i, $(i = 0, 1, \ldots)$:

$$h_0 = h_0, \ h_1 \leqslant \frac{8}{9}h_0, \ h_2 \leqslant \frac{8}{9}h_0^2, \ h_3 \leqslant 2^2\left(\frac{8}{9}\right)^2 h_0^3,$$

$$h_4 \leqslant 2^4\left(\frac{8}{9}\right)^3 h_0^5, \ h_5 \leqslant 2^7\left(\frac{8}{9}\right)^5 h_0^8, \ \ldots$$

We can see that the powers α and β of the constant $\frac{8}{9}$, respectively h_0, satisfie the realtions

$$\alpha_n = \alpha_{n-1} + \alpha_{n-2}$$
$$\beta_n = \beta_{n-1} + \beta_{n-2},$$

so, they are the terms of a Fibonacci sequence with the general term

$$u_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

So, we have

$$h_n \leqslant 2^{u_{n+1}-1}\left(\frac{8}{9}\right)^{u_n} h_0^{u_{n+1}} = \frac{1}{2}\left(\frac{8}{9}\right)^{u_n}(2h_0)^{u_{n+1}}$$

Denotting by

$$s_n = \sum_{i=1}^{n} u_n = u_{n+2} - 1$$

the general term of sequence of partial sums of the Fibonacci sequence and taking account of (7) and (8), we get

$$\eta_1 \leqslant \frac{4}{3} h_0 \eta_0$$

$$\eta_2 \leqslant \left(\frac{4}{3}\right)^2 \frac{8}{9} h_0^2 \eta_0 = \left(\frac{2}{3}\right)^2 \frac{8}{9} (2h_0)^2 \eta_0$$

$$\dots\dots\dots\dots\dots\dots$$

$$\eta_n \leqslant \left(\frac{4}{3}\right)^n \left(\frac{8}{9}\right)^{s_{n-1}} (2h_0)^{s_n} \eta_0.$$

It results that

$$\| x_{n+p} - x_n \| \leqslant \sum_{i=1}^{n+p-1} \eta_i \leqslant 2^{1-s_n} \eta_0 \left(\frac{8}{9}\right)^{s_n - 1} \cdot (4h_0)^{s_n}. \tag{10}$$

For $n \to \infty$, the limit of second member of relation (10) is 0, therefore sequence (x_n) generated by (2) is convergent. The space X being complet, it results that

$$\lim_{n \to \infty} x_n = x^* \in S(x_0, 2\eta_0).$$

From

$$P(x_n) + [x_n, x_{n-1}; P](x_n - x_{n-1}) = 0,$$

taking into account that $[x_n, x_{n-1}; P] \in (X \to Y)^*$ — the set of linear and bounded operators, it follows that $\lim_{n \to \infty} P(x_n) = \theta$, i.e. $P(x^*) = \theta$.

From (10), for $p \to \infty$, it follows (3).

So, the theorem is completely proved.

To prove the uniqueness of solution of equation (1), we have

THEOREM 2. *In the conditions of theorem 1, in the ball $S(x_0, 2\eta_0) \subset X$, the solution is unique.*

Proof. Suppose that $\bar{x} \in S \subset X$ is another solution of equation (1).

We consider the operator $F^{(i)}(x): X \to X$, given by

$$F^{(i)}(x) = x - \Lambda_i P(x)$$

with properties

$$F^{(i)}(\bar{x}) = \bar{x}\,;\ F^{(i)}(x_i) = x_i - \Lambda_i P(x^i) = x_{i+1}$$
$$[u,\ v;\ F^{(i)}] = I - \Lambda_i[u,\ v;\ P]\,;\ \forall u,\ v \in S.$$

For $i = 0$ and taking into account that $[x_0,\ x_{-1};\ F^{(0)}] = \theta$, we have:

$$\|\bar{x} - x_1\| = \|F^{(0)}(\bar{x}) - F^{(0)}(x)\| = \|[\bar{x},\ x_0;\ F^{(0)}](\bar{x} - x_0)\| =$$
$$= \|([\bar{x},\ x_0;\ F^{(0)}] - [x_0,\ x_{-1};\ F^{(0)}])(\bar{x} - x_0)\| =$$
$$= \|\Lambda_0([x_0,\ x_{-1};\ P] - [\bar{x},\ x_0;\ P])(\bar{x} - x_0)\| \leqslant$$
$$\leqslant \|\Lambda_0\|\,\|[x_0,\ x_{-1};\ P] - [x,\ x_0;\ P]\|\,\|x - x_0\|.$$

Based on the hypothesis and the evidently relation

$$\|x - x_{-1}\| \leqslant \|x - x_0\| + \|x_0 - x_{-1}\| \leqslant 2\eta_0 + \eta_{-1}$$

we have

$$\|\bar{x} - x_1\| \leqslant B_0K(2\eta_0 + \eta_{-1})^2\, 2\eta_0 < 2^2 B_0 K(\eta_0 + \eta_{-1})\eta_0 =$$
$$= 2^2 \frac{h_0\eta_0}{1 - h_0} = 2^2\eta_1.$$

Generally, for any n, the following inequality takes place:

$$\|\bar{x} - x_n\| \leqslant 2^{n+1}\eta_n < 2\left(\frac{2}{3}\right)\left(\frac{8}{9}\right)^{s_n - 1}(4h_0)^{s_n}\eta_0. \quad (11)$$

For $n \to \infty$, (11) becomes $\|\bar{x} - x_n\| \to 0$, therefore

$$\lim_{n\to\infty} x_n = \bar{x} = x^*,$$

so, the solution of equation (1) is unique.

3. Theorem 1 can be improved by eliminating the hypothesis of bounding of Λ_0. We prove:

THEOREM 3. *If there are some $x_0, x_{-1} \in S$ for which the following conditions are satisfied:*

1. *Exists $\Lambda_0 = [x_0,\ x_{-1};\ P]^{-1}$;*
2. *There exist η_0 and η_{-1} such that*

$$\|x_0 - x_{-1}\| \leqslant \eta_{-1} \ \textit{and}\ \|\Lambda_0 P(x_0)\| \leqslant \eta_0,\ \eta_0 \leqslant \eta_{-1};$$

3. *There exist $\tilde{K} > 0$ such that for every $x',\ x'',\ x''' \in S\,(x_0,\ 2\eta_0)$ we have*

$$\|\Lambda_0([x',\ x'';\ P] - [x'',\ x''';\ P])\| \leqslant \tilde{K}\|x' - x''\|;$$

4. $\tilde{h}_0 = \tilde{K}(\eta_0 + \eta_{-1}) \leqslant \frac{1}{4}$

then the following statements take place:

i) *The sequence generated by (2) is convergent;*
ii) *If $x^* = \lim x_n$, then $x^* \in S(x_0, \eta_0)$ is a solution of equation (1);*

iii) *The convergence order is caracterized by*

$$\|x^* - x_n\| \leqslant 2^{1-s_n}\left(\frac{8}{9}\right)^{s_n - 1}(4\tilde{h}_0)^{s_n}\eta_0. \tag{3'}$$

Proof. We show that the conditions of theorem 3 imply the conditions of theorem 1, for the equivalent equation with (1)

$$\tilde{P}(x) \equiv \Lambda_0 P(x) = \tilde{0}. \tag{1'}$$

To generated the sequence of appeoximation of a root of (1'), we consider the algorithm

$$\tilde{x}_{n+1} = \tilde{x}_n - \tilde{\Lambda}_n \tilde{P}(\tilde{x}_n)$$

where $\tilde{\Lambda}_n = [\tilde{x}_n, \tilde{x}_{n-1}; \tilde{P}]^{-1}$.

It's easy to show that if $\tilde{x}_0 = x_0$ and $\tilde{x}_{-1} = x_{-1}$, then the sequence $(\tilde{x}_n)$ generated by (2') is identicaly with the sequence generated by (2).

Now, we verify the condition 1°—4° of theorem 1.

1°. $\tilde{\Lambda}_0 = [x_0, x_{-1}; \tilde{P}]^{-1} = \Lambda_0[x_0, x_{-1}; P]^{-1} = I$,
so $\tilde{\Lambda}_0$ exists and $\|\tilde{\Lambda}_0\| = 1 = B_0$

2° $\|\tilde{\Lambda}_0\tilde{P}(x_0)\| \leqslant \|\tilde{\Lambda}_0\| \cdot \|\tilde{P}(x_0)\| = \|\Lambda_0[x', x''; P] - \Lambda_0[x'', x'''; P]\| \leqslant$
$\leqslant \tilde{K}\|x' - x'''\|, \ \forall x', x'', x''' \in S(x_0, 2\eta_0)$

3° $\|[x', x''; \tilde{P}] - [x'', x'''; \tilde{P}]\| = \|\Lambda_0[x', x''; P] - \Lambda_0[x'', x'''; P]\| \leqslant$
$\leqslant \tilde{K}\|x' - x'''\|, \ \forall x', x'', x''' \in S(x_0, 2\eta_0)$

4° $\tilde{h}_0 = \tilde{K}(\eta_0 + \eta_{-1}) < \frac{1}{4}$.

Accouding to theorem 1, it results that equation (1') has a solution $x^* \in S$, which is the limit of sequence (x_n) generated by (2) or (2'), the order of convergence being given by (3) or (3').

4. Now, we present an application of the chord method at resolution of a sistem of two real equations with two real unknows.

Let be the system

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0. \end{cases}$$

In this case $P: \mathbf{R}^2 \to \mathbf{R}^2$ is given by

$$P(x, y) = (f(x, y), g(x, y)).$$

Considering as the initial approximations the points (x_1, y_1), (x_2, y_2), one verifies the theorem 1 conditions, it can be implemented, according to the flow-chart the algorithm of approximative solving of system.

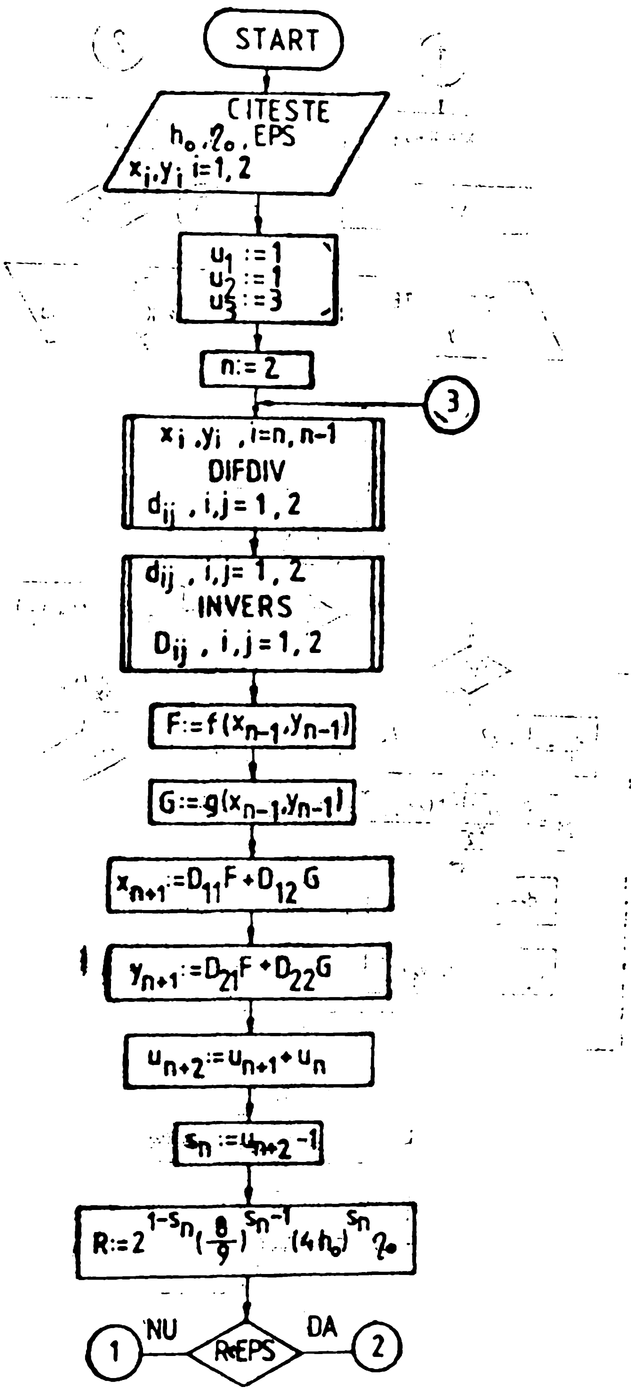
START
CITESTE
h₀, η₀, EPS
x_i, y_i i=1, 2
u_1 := 1
u_2 := 1
u_3 := 3
n := 2
3
x_i, y_i, i=n, n-1
DIFDIV
d_ij, i,j = 1, 2
d_ij, i,j = 1, 2
INVERS
D_ij, i,j = 1, 2
F := f(x_{n-1}, y_{n-1})
G := g(x_{n-1}, y_{n-1})
x_{n+1} := D_11 F + D_12 G
y_{n+1} := D_21 F + D_22 G
u_{n+2} := u_{n+1} + u_n
s_n := u_{n+2} - 1
R := 2^{1-s_n} (8/9)^{s_n - 1} (4h₀)^{s_n} η₀
NU
R<EPS
DA
1
2

Fig. 1

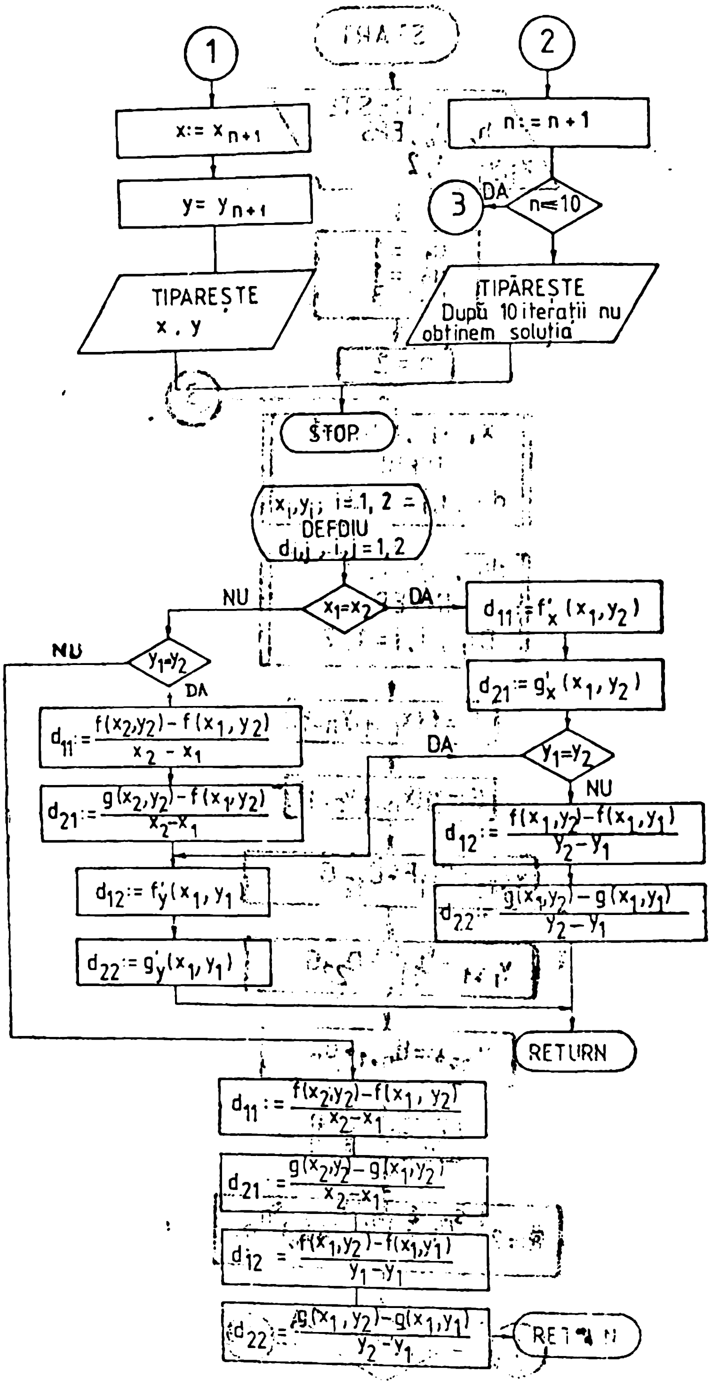
START
1
x:= x n+1
y= y n+1
TIPAREȘTE
x , y
2
n:= n+1
DA
n≤10
3
TIPĂREȘTE
După 10 iterații nu obținem soluția
STOP
DEFDIU
NU
x1=x2
DA
d11 := f'x (x1, y2)
d21 := g'x (x1, y2)
NU
y1=y2
DA
d11 := (f(x2,y2) - f(x1,y2)) / (x2 - x1)
d21 := (g(x2,y2) - f(x1,y2)) / (x2-x1)
d12 := f'y(x1, y1)
d22 := g'y(x1, y1)
DA
y1=y2
NU
d12 := (f(x1,y2)-f(x1,y1)) / (y2-y1)
d22 := (g(x1,y2) - g(x1,y1)) / (y2-y1)
RETURN
d11 := (f(x2,y2)-f(x1, y2)) / (x2-x1)
d21 := (g(x2,y2)-g(x1,y2)) / (x2-x1)
d12 := (f(x1,y2)-f(x1,y1)) / (y1-y1)
d22 := (g(x1,y2)-g(x1,y1)) / (y2-y1)

Fig. 2

We use the fact that if $u = (x_1, y_1)$, $v = (x_2, y_2)$, where $x_1 \neq x_2$ and $y_1 \neq y_2$, then

$$[u, v; P] = \begin{pmatrix} \frac{f(x_2, y_2) - f(x_1, y_2)}{x_2 - x_1} & \frac{f(x_1, y_2) - f(x_1, y_1)}{y_2 - y_1} \\ \frac{g(x_2, y_2) - g(x_1, y_2)}{x_2 - x_1} & \frac{g(x_1, y_2) - g(x_1, y_1)}{y_2 - y_1} \end{pmatrix}.$$

If $x_1 = x_2$ or $y_1 = y_2$, the elements of previos matrix are substituted by cooresponding partial derivates.

We present the flow-chart of the algoritm, using the following notations:

1) n, the iteratios number.

2) INVERS, the subroutine for obtaining the inverse of the matrix $d_{i,j}$, denotet by D_{ij}. This subroutine is found in many forms, for any nonsingular squere matrix, in the mathematical library of computers.

3) *DIFDIV*, the subroutine for obtaining the first ordered divided differences, with parameters (x_i, y_i) $i = 1, 2$ and d_{ij}, $i, j = 1, 2$.

REFERENCES

1. Groze, S., Goldner, G., Jankó, B., *Asupra metodei coardei in rezolvarea ecuațiilor operaționale definite în spații supermetrice*, St. Cec. Mat., 25, 5, 1971, p. 719–725.
2. Groze, S., *Asupra condițiilor de convergență la metoda coardei în spații supermetrice*, Studia Univ. Babeș–Bolyai, Math.-Mec., 1, 1973, p. 55–59.
3. Goldnar, G., Balaz's, M., *Asupra metodei coardei și a unei modificări a ei pentru rezolvarea ecuațiilor operaționale neliniare*, St. Cerc. Mat., 7, 1964, 20.
4. Collatz, L., *Funktionalanalysis und Numerische Matematic*, Vorlag Springer, Berlin, 1965.

STUDIA UNIV. BABEŞ–BOLYAI, MATHEMATICA, XXXII, 3, 1987

OPTIMAL ALGORITHMS FOR THE SOLUTION OF NONLINEAR EQUATION WITH REGARD TO THE EFFICIENCY

GH. COMAN*

Received: June 6, 1987

REZUMAT. — Algoritmi optimali, în raport cu eficiența, pentru rezolvarea ecuațiilor neliniare. În lucrare se studiază problema optimității, în raport cu eficiența, în clasa algoritmilor de aproximare a soluțiilor unei ecuații neliniare pe R, obținuți prin procedeul interpolării inverse Taylor. Rezultatul principal este formulat în teorema dată.

1. *Introduction.* In the paper by J. F. Traub [3] it was studied the optimality problem, with regard to the efficiency, in the class of inverse Taylor interpolation algorithms, for the solution of nonlinear equations. In the mentioned paper, instead of the efficiency expression is used an approximation of it.

In this paper, it is studied the same problem using a more finer approximation for the efficiency.

2. *Preliminaries.* Let X be a linear space over the real or complex field K, $(Y, ||.||)$ a normed linear space over K, X_0 a subset of X and S, $S: X_0 \to Y$, a given operator. One considers the following problem [4]: for a given ε, $\varepsilon > 0$, to find an ε-approximation $y = y(x)$, $y \in Y$, to $s = S(x)$ for all $x \in X_0$. S is called the solution operator, x is a problem element and s is a solution element. The considered problem is refered as the problem S.

If $X = C[a, b]$, $X_0 = \{f \in X \mid f(x) \leqslant 0,\ f(b) \geqslant 0$ and f has an unique zerou in the interval $[a, b]\}$, $Y = \mathbf{R}$ and $S: X_0 \to \mathbf{R}$ is given by $S(f) = f^{-1}(0)$, then S is an ε-approximation problem for the solution of the equation $f(t) = 0$, $t \in (a, b)$.

For $X_1 \subseteq X$, such that $X_0 \subseteq X$, one denotes by $\mathcal{J}$, $\mathcal{J}: X_1 \to Z$, where Z is a given set, the information operator. $\mathcal{J}(x)$, for $x \in X_1$, is called the information of x. Let also, α, $\alpha: \mathcal{J}(X_0) - Y$, be an algorithm for solving a problem S with the information $\mathcal{J}$, and $\mathcal{A}(S, I)$ the set of all such algorithms. The value $e(S, \mathcal{J}, \alpha)$ (breifly $e(\alpha)$), defined by

$$e(\alpha) = \sup_{x \in X_0} ||S(x) - \alpha(\mathcal{J}(x))||$$

is the error of the algorithm α.

Let $\mathcal{R}$ be a set of elementary operations. Next, one supposes that $\mathcal{J}$ and α are $\mathcal{R}$-admissible, i.e. $\mathcal{J}(x)$ respectively $\alpha(\mathcal{J}(x))$ can be computed with a finite number of operations from R (taking into account that some of the operations

* University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania

can be aplied several times). If $r_1, \ldots, r_m \in \mathfrak{R}$ are the necessary operations to compute $\mathcal{J}(x)$, the value

$$\mathrm{CPE}(\mathcal{J}(x)) = \sum_{i=1}^{m} p_i \mathrm{CP}(r_i),$$

where p_i is the performing number of the operation r_i and $\mathrm{CP}(r_i)$ is the complexity of r_i, is called the complexity of the information $\mathcal{J}(x)$. Also, if $\rho_1, \ldots, \rho_n \in \mathfrak{R}$ are the necessary operations to compute $\alpha(\mathcal{J}(x))$, the value

$$\mathrm{CPC}(\alpha(\mathcal{J}(x))) = \sum_{i=1}^{n} q_i \mathrm{CP}(\rho_i),$$

where q_i is the performing number of the operation ρ_i, is the combinatorial complexity of the algorithm α for the problem element x. The value

$$\mathrm{CP}(\alpha(\mathcal{J}(x))) = \mathrm{CPE}\,(\mathcal{J}(x)) + \mathrm{CPC}(\alpha(\mathcal{J}(x)))$$

is the complexity of the algorithm α for x, $x \in X_0$, or the local complexity of the algorithm α.

The value

$$\mathrm{CP}(\alpha) \doteq \sup_{x \in X_0} \mathrm{CP}\,(\alpha(\mathcal{J}(x)))$$

is called the complexity of the algorithm α for the problem S with the information $\mathcal{J}$.

An algorithm $\alpha^* \in \mathcal{A}(S, \mathcal{J})$ for which

$$\mathrm{CP}(\alpha^*) = \inf_{\alpha \in \mathcal{A}(S, \mathcal{J})} \mathrm{CP}(\alpha)$$

is called an optimal complexity algorithm in the class $\mathcal{A}(S, \mathcal{J})$.

It follows that the complexity can be used as a criteria to evaluate the „goodness" of an algorithm.

The mathematical problems can be devided in two closses: finite-complexity problems and infinite complexity problems. A finite-complexity problem is a problem for which there exists at least a $\mathfrak{R}$-admissible algorithm α using a $\mathfrak{R}$-admissible information $\mathcal{J}$, that solves it exactly ($\varepsilon=0$), with a finite complexity ($\mathrm{CP}(\alpha) < +\infty$). A problem is an infinite-complexity problem iff it is not a finite-complexity problem.

With a usual set of operations $\mathfrak{R}$, the most problems of mathematics, science and engeneering, are infinite-complexity problems.

Of course, the problem to approximate a solution of a nonlinear equation is a infinite-complexity problem. It is, also called, an iterative problem.

In such a case, first we must determine the class $\mathcal{A}(S, \mathcal{J})$ of the algorthms α for which $\alpha(\mathcal{J}(x))$ are ε-approximations of the solution element, $S(x)$. This problem is practically rather dificult. For example, if α is an iterative algorithm, we can not apriori know, the necessary number of iterations in order to get an ε-approximaticn.

In these cases, it was defined a new characteristic of an algorithm that depends on of its complexity, as well as of its order (of convergence or of approximation) [3].

Definition 1. Let α be an algorithm for the problem S with the information $\mathcal{J}(\alpha \in \mathcal{A}(S, \mathcal{J}))$. The number $p, p = p(\alpha)$, with

$$\lim_{h \to 0} \frac{e(S, \mathfrak{I}, \alpha)}{h^p} = C, \quad (C \neq 0)$$

where C is a constant, is called the order of the algorithm α.

Definition 2. The complexity of an algorithm α that solves a problem S approximative, in called analytic complexity and is denotes by CPA (α).

For example, if α is an iterative algorithm, CPA (α) is the computational complexity of one iteration.

Definition 3. The value

$$E(S, \mathcal{J}, \alpha) = \frac{\log p(\alpha)}{\mathrm{CPA}(\alpha)}$$

is called the efficiency of the algorithm α.

Remark 1. For all logarithms from this paper, we use the base 2.

Definition 4. An algorithm $\tilde{\alpha} \in \mathcal{A}(S, \mathcal{J})$, for which

$$E(S, \mathcal{J}, \tilde{\alpha}) = \sup_{\alpha \in \mathcal{A}(S, \mathfrak{I})} E(S, \mathcal{J}, \alpha)$$

is called an optimal efficiency algorithm.

3. *Numerical solution for nonlinear equations.* For $X = C[a, b]$, $X_0 = \{f \in C[a, b] \mid f(a) \leqslant 0,\ f(b) \geqslant 0$ and f has an unique zerou in the interval $[a, b]\}$, $Y = \mathbf{R}$ and $S : X_0 \to \mathbf{R}$ is defined by $S(f) = f^{-1}(0)$, S is a problem for the solution of the equation $f(t) = 0$.

Next, we consider, for the solution of this problem, the class of algorithms generated by inverse Taylor interpolation procedure, i.e.

$$\mathcal{A}^T(S, \mathcal{J}) = \{\alpha_m^T \mid m \in \mathbf{N},\ m \geqslant 2\}$$

with

$$\alpha_m^T(t) = t - \sum_{k=1}^{m-1} \frac{(-1)^k}{k!} [f(t)]^k g^{(k)}(f(t)), \tag{1}$$

where g is the inverse function of $f (g = f^{-1})$.

It can be observed that the information of an element $f \in X_0$, is $\mathcal{J}(f) = (f(t), f'(t), \ldots, f^{(m-1)}(t))$.

So, the complexity of the information $\mathcal{J}(f)$ is

$$\mathrm{CPE}(\mathfrak{I}(f)) = \sum_{i=0}^{m-1} \mathrm{CP}(f^{(i)}),$$

where CP(h) is the computational complexity for the evaluation of $h(t)$, $t \in (a, b)$.

Also, it is well known that the order p of the algorithm α is $p(\alpha_m^T) = m$.

The difficulty is, to determine the combinatorial complexity CPC$(\alpha_m^T(\mathfrak{I}(f)))$, i.e. we have to compute the derivatives $g^{(k)}$, $k = \overline{1, m-1}$.

First, we give a lower bound for the combinatorial complexity CPC(α_m^T).

Lemma. *If* $\mathfrak{A} = \{+, -, *, /\}$, *we have* CPC $(\alpha_2^T(f))$ = CP(+) + CP(/), CPC $(\alpha_m^T(f)) \geq \frac{m^2 - 3m + 2}{2}$CP(+) + $\frac{m^2 + 3m - 14}{2}$CP(*) + $(2m - 3)$CP(/) *for any* $m > 2$, *with equality for* $m = 3, 4$.

Proof. We have

$$g^{(k)} = (-1)^{k-1} P_k / (f')^{2k-1}, \tag{2}$$

where P_k is a polynomial in the variables $f^{(i)}$, $i = \overline{1, k}$, that satisfies the recurrence relations:

$$P_1 = 1; \; P_{k+1} = (2k - 1) f'' P_k - f' P_k', \; k = 1, 2, \ldots \tag{3}$$

It follows that all the coefficients of the polynomial P_k are integer numbers.

But [5]

$$g^{(k)}(f) = \sum_{(i_1, \ldots, i_k) \in I} (-1)^{k-1+i_1} \frac{(2k - 2 - i_1)!}{i_2! \ldots i_k! (f')^{2k-1}} \left(\frac{f'}{1!}\right)^{i_1} \ldots \left(\frac{f^{(k)}}{k!}\right)^{i_k}, \tag{4}$$

where I is the set of integer solutions of the system

$$\begin{cases} i_2 + 2i_3 + \ldots + (k-1) i_k = k - 1 \\ i_1 + i_2 + \ldots + \quad i_k \quad = k - 1 \end{cases} \tag{5}$$

Remark 2. From the second equations of the system (5), it follows that P_k is a homogeneous polynomial of the degree $k - 1$.

Using the notations

$$r_j = f^{(j)}/f', \; j \in \mathbf{N},$$

one obtains

$$g^{(k)}(f) = \frac{(-1)^{k-1}}{(f')^k} P_k(r_2, \ldots, r_k). \tag{6}$$

We remind that P_k is a polynomial of the degree $k - 1$ in the variables $r_2, \ldots, r_k$, with integer coefficients. For example, we have

$P_1 = 1$
$P_2 = r_2$
$P_3 = 3r_2^2 - r_3$
$P_4 = 15 r_2^3 - 10 r_2 r_3 + r_4$
$P_5 = 105 r_2^4 - 105 r_2^2 r_3 + 10 r_3^2 + 15 r_2 r_4 - r_5$

Remark 3. From (3), it follows that the expression of the polynomial P_{k+1} contains at least one term more than the expression of P_k, for $k > 1$. Also, excepting one term of P_k, all the others are formed by two or more factors.

By (1) and (6), one obtains:

$$F_m^T(t) = t - \sum_{k=1}^{m-1} \frac{1}{k!} r_0^k(t) P_k(r_2(t), \ldots, r_k(t)). \tag{8}$$

Now, we evaluate the combinatorial complexity of the algorithm α_m^T for the set of elementary operations $R = \{+, -, *, /\}$, where „$-$" is identifies with „$+$".

Additions: $(m - 1)$ to evaluate the expression from (8), which contains m terms; at least $(1 + 2 + \ldots + m - 3)$ to evaluate the polynomials $P_3, \ldots, P_{m-1}$ (remark 3). So, the total number of additions is at least $(m^2 - 3m + 4)/2$.

Multiplications: $(m - 3) + (m - 2) + (m - 2) + (m - 3)$ to compute $(m - 1)!$, r_0^k, $k = \overline{2, m-1}$, $r_0^k P_k$, $k = \overline{2, m-1}$ respectively r_2^k, $k = \overline{2, m-2}$; at least $(1 + 2 + \ldots + m - 3)$ to evaluate the polynomials $P_3, \ldots, P_{m-1}$ (remark 3). Hence, the number of multiplications is at least $(m^2 + 3m - 14)/2$.

Divisions: $2m - 3$ to compute $r_k (r_k = f^{(k)}/f')$, $k = \overline{0, m-1}$, $k \neq 1$ respectively $r_k/k!$, $k = \overline{2, m-1}$.

Finally, one obtains

$$\mathrm{CPC}\,(\alpha\ (f)) \geqslant \frac{m^2 - 3m + 4}{2}\,\mathrm{CP}(+) + \frac{m^2 + 3m - 14}{2}\,\mathrm{CP}(*) + (2m - 3)\,\mathrm{CP}(/) \tag{9}$$

for any $m > 2$, with equality for $m = 3, 4$. The equality

$$\mathrm{CPC}(\alpha_2^T(f)) = \mathrm{CP}(+) + \mathrm{CP}(/)$$ is obviously.

Next, we approximate the combinatorial complexity $\mathrm{CPC}(\alpha_m^T)$ by $\overline{\mathrm{CPC}}(\alpha_m^T(f)) = \frac{m^2 - 3m + 4}{2}\,\mathrm{CP}(+) + \frac{m^2 + 3m - 14}{2}\,\mathrm{CP}(*) + (2m - 3)\,\mathrm{CP}(/)$.

Remark 4. As, we are going to deal with the class $\mathcal{A}(S, \mathcal{J}(f))$ of the algorithms $\alpha_m^T(f)$, for a given $f \in X_c$, it can be used the local analytic complexity:

$$\mathrm{CPA}(\alpha_m^T(f)) = \mathrm{CPE}(\mathcal{J}(f)) + \mathrm{CPC}\,(\alpha_m^T(f))$$

instead of

$$\mathrm{CPA}\,(\alpha_m^T) = \sup_{f \in X_c} \mathrm{CPA}\,(\alpha_m^T(f)).$$

It follows that

$$\mathrm{CPA}\,(\alpha_m^T(f)) = \sum_{k=0}^{m-1} \mathrm{CP}\,(f^{(k)}) + \mathrm{CPC}\,(\alpha_m^T(f)).$$

If

$$\overline{\mathrm{CPA}}\,(\alpha_m^T(f)) = \sum_{k=0}^{m-1} \mathrm{CP}(f^{(k)}) + \overline{\mathrm{CPC}}\,(\alpha_m^T(f))$$

then

$$\mathrm{CPA}\,(\alpha_m^T(f)) \geqslant \overline{\mathrm{CPA}}(\alpha_m^T(f)), \text{ for any } m > 1.$$

So,

$$E(S, \mathcal{J}, \alpha_m^T(f)) = \frac{\log m}{\mathrm{CPA}(\alpha_m^T(f))}$$

and

$$E(S, \mathcal{J}, \alpha_m^T(f)) \leqslant \bar{E}(S, \mathcal{J}, \alpha_m^T(f)) = \frac{\log m}{\overline{\mathrm{CPA}}(\alpha_m^T(f))}. \tag{10}$$

Remark 5. Suppose that $\mathrm{CP}(*) = \mathrm{CP}(/) = 2\,\mathrm{CP}(+)$ and $\mathrm{CP}(+) = 1$, and $\mathrm{CP}(f^{(k)}) = \mathrm{CP}(f)'$ for any $k = \overline{1, m-1}$.

In these conditions, we have

$$\bar{E}(S, \mathcal{J}, \alpha\ (f)) \approx \tilde{E}(S, \mathcal{J}, \alpha_m^T(f)) = \frac{\log m}{m\mathrm{CP}(f) + (3m^2 + 11m - 36)/2}.$$

So, we have

$$\tilde{E}(S, \mathcal{J}, \alpha_m^T(f)) = \begin{cases} \dfrac{1}{2\mathrm{CP}(f) + 3}, & \text{for } m = 2 \\ \dfrac{\log m}{m\mathrm{CP}(f) + (3m^2 + 11m - 36)/2}, & \text{for } m > 2 \end{cases} \tag{11}$$

THEOREM. *Let $\mathcal{A}^T(S, \mathcal{J}(f))$ be the class of all algorithms $\alpha_m^T(f)$, $m \geqslant 2$, for a given f. Then, the optimal algorithm, with regard to the efficiency, in the class $\mathcal{A}(S, \mathcal{J}(f))$, in the conditions of remark 5, is α_2^T for* $\mathrm{CP}(f) \leqslant 42$ *and α_3^T for* $\mathrm{CP}(f) > 42$

Proof. By (11), if follows that $\tilde{E}$ is a positive and decreasing function with regard to m, for $m > 2$ and for any $\mathrm{CP}(f)$, $\mathrm{CP}(f) > 0$. So,

$$\tilde{E}(S, \mathcal{J}, \alpha_3^T(f)) < \tilde{E}(S, \mathcal{J}, \alpha_m^T(f)) \text{ for any } m > 3.$$

Now, from the relation

$$\tilde{E}(S, \mathcal{J}, \alpha_3^T(f)) - \tilde{E}(S, \mathcal{J}, \alpha_2^T(f)) = \frac{(\log 9 - 3)\,\mathrm{CP}(f) + 3(\log 3 - 4)}{(2\mathrm{CP}(f) + 3)(3\mathrm{CP}(f) + 12)}$$

it follows that

$$\tilde{E}(S, \mathcal{J}, \alpha_3^T(f)) > \tilde{E}(S, \mathcal{J}, \alpha_2^T(f)) \text{ for } \mathrm{CP}(f) > 42 \tag{12}$$

and

$$\tilde{E}(S, \mathcal{J}, \alpha_3^T(f)) < \tilde{E}(S, \mathcal{J}, \alpha_2^T(f)) \text{ for } \mathrm{CP}(f) \leqslant 42. \tag{13}$$

As, $\bar{E}(S, \mathcal{J}, \alpha_m^T) = E(S, \mathcal{J}, \alpha_m^T)$ for $m=2, 3, 4$ and $\bar{E}(S, \mathcal{J}, \alpha_m^T(f)) < E(S, \mathcal{J}, \alpha_m^T(f))$ for $m > 4$, the inequalities (12) and (13) are presevred for the real case, of course, in the conditions that $CP(f^{(k)}) = CP(f)$, for $k = \overline{1, m-1}$, and the theorem is proven.

Remark 6. If CP $(f^{(k)})$ is an icreasing function with regard to k, then α_3^T can become optimal on the class $\mathcal{A}^T(S, \mathcal{J}(f))$. For example, we have such a case for

$$CP(f'') > (CP(f) + CP(f') + 2)(\log 3 - 1) + \log 3.$$

If $CP(f^{(k)})$ is a decreasing function in the variable k, then an algorithm α_m^T for $m > 3$, can become optimal in the class $\mathcal{A}^T(S, \mathcal{J}(f))$.

REFERENCE

1. Coman Gh., *On the complexity of some numerical algorithms.* „Babeş–Bolyai" University, Research Seminaries, Preprint Nr. 4, 1–33.
2. Smale, S., *On the efficiency of algorithms of analysis.* Bulletin of the AMS 13, 1985, 87–121.
3. Traub J. F., *Theory of optimal algorithms.* In Software for numerical mathematics (ed. by D. F. Evans), Acad. Press 1974.
4. Traub J. F., Wozniakowski H., *A general theory of optimal algoritms.* Acad. Press 1980.
5. Turowics B. A., *Sur les derivées d'ordre superieur d'une fonction inverse.* Colloq. Math. 1959, 83–87.

STUDIA UNIV. BABEŞ–BOLYAI, MATHEMATICA, XXXII, 3, 1987

THE STORING OF DATA COLLECTIONS IN ACCORDANCE WITH THE CONSECUTIVE RETRIEVAL PROPERTY

LEON ŢÂMBULEA*

Received: June 8, 1987

ABSTRACT. — A few results regarding the consecutive retrieval property for maximal and connected question sets are given in the first part of the paper. Next, an algorithm determining the order in which data must be stored so that the consecutive retrieval property may occur is provided for these types of sets. These results are then extended to data collection endowed with consecutive retrieval property relative to a certain set of questions.

Let $\mathcal{C}$ be a data collection (data base) stored on medium $\mathcal{S}$, that we consider to be linear. Let us ussume that the data requirements from $\mathcal{C}$ are questions, and Q is the set of these questions. For a question $q \in Q$ we mark with $q(\mathcal{C})$ the answer, i.e. the elements from $\mathcal{C}$ useful to the question q. For giving the answer $q(\mathcal{C})$, a time $t(q)$ (response time) is needed.

The most important parameters for organizing the data collection are ([1, 3]):

— the medium (support) space necessary for storing the collection;

— the average time for answering questions.

As a rule, these two parameters cannot be reduced at the same time, the decrease any could imply on increase of the other.

S. P. Ghosh ([2]) discovered the consecutive retrieval property, in connection to the description of an optimum organizing of a data collection, in which both the medium space and the response time were reduced to a minimum.

DEFINITION 1. The data collection $\mathcal{C}$ has the consecutive retreival property (CR-property) relative to Q if all the data from $q(\mathcal{C})$ are stored consecutively on medium $\mathcal{S}$, $\forall q \in Q$.

In view of achieving optimal results, this property has been intensively studied. For a proper characterization of the data collection that have this property, linear families of sets, interval graphs, as well as boolean matrix have been used. Various approximations disparaging one of the above mentioned parameters have been found for data collections lacking this property.

DEFINITION 2. For the data collection $\mathcal{C} = \{d_1, \ldots, d_m\}$ and the question set $Q = \{q_1, \ldots, q_n\}$ we consider the boolean matrix A, with m lines and n columns,

$$A_{ij} = \begin{cases} 1 & \text{if } d_i \in q_j(\mathcal{C}), \\ 0 & \text{otherwise.} \end{cases}$$

* University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania

DEFINITION 3. The data d_i and d_j are different if the lines i and j in the A matrix differ.

THEOREM 1. *If* $\mathcal{C} = \{d_1, \ldots, d_m\}$, *with all differing data, has CR-property relative to a set* $Q = \{q_1, \ldots, q_n\}$, *then* $m \leqslant 2n - 1$.

Proof. Let us consider that $M_1 = M_2 = \Phi$. We assume that the data from the collection $\mathcal{C}$ are stored on the medium $\mathcal{S}$ in such a way that $q(\mathcal{C})$ are consecutive, $\forall q \in Q$. It results that in every column of the A matrix the elements equal to 1 are consecutively placed.

For every $q_j \in Q$ we mark with $b(q_j)$ respectively $e(q_j)$ the address (position) of the first, respectively the last datum, in $q_j(\mathcal{C})$, on the medium $\mathcal{S}$. Since $\mathcal{S}$ is linear, these values are extant. With these values we form the following sets:

$$C_1 = \{d_{b(q_j)}, j = 1, \ldots, n\} \text{ and } C_2 = \{d_{e(q_j)}, j = 1, \ldots, n\}.$$

We have: $|C_1| \leqslant n$ and $|C_2| \leqslant n$.

Lines 0 and $m + 1$ with all elements egual with 0 are to be added to matrix A. Let d_0 and d_{m+1} be the data which correspond to these two lines. Considering that for every $i = 0, 1, \ldots, m$, the data d_i and d_{i+1} are different, thus exists j_0 so that $A_{i,j_0} \neq A_{i+1,j_0}$.
If: a) $A_{i,j_0} = 0$, then $A_{i+1,j_0} = 1$, therefore $b(q_{j_0}) = i + 1$ and $d_{i+1} \in C_1$. In this case d_i is to be added to M_1;
b) $A_{i,j_0} = 1$, then $A_{i+1,j_0} = 0$, therefore $e(q_{j_0}) = i$ and $d_i \in C_2$. In this case d_i is to be added to M_2.

From these assertions it results that:

$$M_1 \subseteq C_1, \ M_2 \subseteq C_2,$$

and:

$$m + 1 = |M_1| + |M_2| \leqslant |C_1| + |C_2| \leqslant 2n,$$

therefore the theorem is proved.

DEFINITION 4 [5]. The set $T \subseteq \mathcal{C}$ is an atom if it is nonempty and it can be represented as follows:

$$T = \bigcup_{i=1}^{n} \tilde{M}_i,$$

where $M_i = q_i(\mathcal{C})$ and $\tilde{M}_i = M_i$ or $\bar{M}_i$, for every $i = 1, \ldots, n$.

REMARK. An atom can be represented as follows:

$$T_I = \left(\bigcap_{i \in I} M_i\right) \cap \left(\bigcap_{i \in \bar{I}} \bar{M}_i\right), \ I \neq \Phi. \quad (1)$$

$I \neq \Phi$ because every datum of $\mathcal{C}$ is useful for at least one question from Q. Otherwise, this datum is deleted from $\mathcal{C}$.

THEOREM 2. *If* $\mathcal{C}$ *has CR-property relative to* $Q = \{q_1, \ldots, q_n\}$, *then the data of every atom can be stored consecutively on* $\mathcal{S}$.

Proof. We assume that $\mathcal{C}$ is stored on $\mathcal{S}$ so that the data in every M_i are consecutive, being delimited by the addresses (positions) a_1^i and a_2^i, $a_1^i \leqslant a_2^i$, for $i = 1, \ldots, n$. Let a_1^T and a_2^T be, $a_1^T \leqslant a_2^T$ the extreme addresses between which the atom of T_I are stored (T_I has the form (1)). We have:

$$a_1^i \leqslant a_1^T \leqslant a_2^T \leqslant a_2^i, \quad \forall i \in I \subseteq \{1, 2, \ldots, n\}, \ I \neq \Phi; \tag{2}$$

$$a_2^T < a_1^i \text{ or } a_2^i < a_1^T, \ \forall \in \bar{I} = i\{1, 2, \ldots, n\} - I. \tag{3}$$

We assume that T_I isn't stored consecutively on $\mathcal{S}$, then: $\exists d \in \mathcal{C}$, $d \notin T_I$, stored at address a, and:

$$a_1^T < a < a_2^T \tag{4}$$

Since $d \notin T_I$, one of the following two conditions is true:

a) $d \notin \bigcap_{i \in I} M_i$, then: $\exists s \in I$, $d \notin M_s$, or:

$$\exists s \in I : a \notin [a_1^s, a_2^s]; \tag{5}$$

b) $d \notin \bigcap_{i \in \bar{I}} \bar{M}_i$, then: $\exists t \in \bar{I}$, $d \notin \bar{M}_t$ ($d \in M_t$), or:

$$\exists t \in \bar{I} : a \in [a_1^t, a_2^t]. \tag{6}$$

From (2) and (4) we obtain:

$$a_1^i \leqslant a_1^T < a < a_2^T \leqslant a_2^i, \quad \forall i \in I,$$

from which it results that (5) is false.

From (3) and (4) we obtain:

$$a < a_2^T < a_1^i \quad \text{or} \quad a_2^i < a_1^T < a, \quad \forall i \in \bar{I},$$

from which it results that (6) is false.

From these two contradictions it results that the initial assum tion are false and hence that the theorem is proved.

THEOREM 3. *If* $\mathcal{C}$ *has* CR*-property relative to* Q, $|Q| = n$, *then the maximum number of atoms is* $2n - 1$.

Proof. Let A be the matrix of definition 2. All lines corresponding to the data of an atom are equal. This theorem results from this remark as well as from theorem 1.

DEFINITION 5. Two atoms T_I and T_J of the form (1) are neighbours if: $\exists i_0 \in I$, $i_0 \notin J$ and $I = J \cup \{i_0\}$.

DEFINITION 6. Let $\mathcal{C}$ be a data collection having CR-property relative to Q. Two atoms are consecutive if they are stored consecutively on medium $\mathcal{S}$.

DEFINITION 7. The set Q is maximal relative to Q if:

$$\forall i, j \in \{1, 2, \ldots, n\}, i \neq j : q_i(\mathcal{C}) \nsubseteq q_j(\mathcal{C}) \text{ and } q_j(\mathcal{C}) \nsubseteq q_i(\mathcal{C}).$$

REMARK. By „$\mathcal{C}$ has CR-property relative to a maximal set Q" we mean that $\mathcal{C}$ has CR-property relative to Q and Q is maximal relative to $\mathcal{C}$.

From definition 7 it results that a maximal set Q relative to $\mathcal{C}$ has at last two elements.

THEOREM 4. *If $\mathcal{C}$ has CR-property relative to a maximal set $Q = \{q_1, \ldots, q_n\}$, then two neighbour atoms are consecutive.*

Proof. Let T_I and T_J be two neighbour atoms and i_0 the value given in definition 5. If: $M_i = q_i(\mathcal{C})$, $i = 1, \ldots, n$, then:

$$T_I = \left(\bigcap_{i \in J} M_i\right) \cap \left(\bigcap_{i \in \bar{I}} \bar{M}_i\right) \cap M_{i_0}; \quad T_J = \left(\bigcap_{i \in J} M_i\right) \cap \left(\bigcap_{i \in \bar{I}} \bar{M}_i\right) \cap M_{i_0} \tag{7}$$

it results that:

$$T_I \subseteq M_i, \ T_J \subseteq M_i, \ \forall i \in J.$$

Let: $a_1^I, a_2^I, a_1^J, a_2^J$; $a_1^i, a_2^i, i = 1, \ldots, n$ be the adresses (positions) on $\mathcal{S}$ situated on the extreme left, respectively right positions, where the T_I, T_J and M_i $(i = 1, \ldots, n)$ sets are stored.

Since $\mathcal{C}$ has CR-property relative to Q, from theorem 2 it results that:

$$a_1^i \leqslant a_1^I \leqslant a_2^I \leqslant a_2^i, \ \forall i \in J \cup \{i_0\}; \tag{8}$$

$$a_2^I < a_1^i \text{ or } a_2^i < a_1^I, \ \forall i \in \bar{I}; \tag{9}$$

$$a_1^i \leqslant a_1^J \leqslant a_2^J \leqslant a_2^i, \ \forall i \in J; \tag{10}$$

$$a_2^J < a_1^i \text{ or } a_2^i < a_1^J, \ \forall i \in \bar{I} \cup \{i_0\}. \tag{11}$$

We assume that on the medium $\mathcal{S}$, T_I is stored before T_J, then $a_2^I < a_1^J$.

If we assume that the theorem is false, T_I as well as T_J are not consecutive, therefore $\exists d \in \mathcal{C}$, $d \notin T_I \cup T_J$ and d is stored on $\mathcal{S}$ (at the address a) between T_I and T_J. It results that:

$$a_1^I \leqslant a_2^I < a < a_1^J \leqslant a_2^J. \tag{12}$$

Since $d \notin T_I \cup T_J$ and (7) is true, we obtain:

$$\exists s \in J : d \notin M_s \quad \text{or} \quad \exists t \in \bar{I} : d \in M_t \tag{13}$$

This condition is equivalent to the fulfilment of one from the following posibilities:

$$\exists s \in J : a \notin [a_1^s, a_2^s]; \tag{14a}$$

$$\exists t \in I : a \in [a_1^t, a_2^t]; \tag{14b}$$

In this case (8), (9), (10), (11), (12), (14a) or (14b) are true. From (8), (10) and (12) it follows that:

$$a_1^i \leqslant a_1^I \leqslant a_2^I \leqslant a \leqslant a_1^J \leqslant a_2^J \leqslant a_2^i, \ \forall i \in J.$$

Hence it results that (14a) isn't true.

From (9) and (11) it results that for $t \in \bar{T}$ one of the following four conditions must be true:

$$\text{C1. } a_2^J \leqslant a_1^t \text{ and } a_2^J < a_1^t;$$

$$\text{C2. } a_2^J < a_1^t \text{ and } a_2^t < a_1^J;$$

$$\text{C3. } a_2^t < a_1^J \text{ and } a_2^J < a_1^t;$$

$$\text{C4. } a_2^t < a_1^J \text{ and } a_2^t \leqslant a_1^J;$$

From C1 and (14b) it results that $a_1^J < a$ and $a_2^J < a$, is in contradiction to (12). From C2, (12) and (14b) it results that.

$$a_1^J \leqslant a_2^J < a_1^t < a < a_2^t < a_1^J \leqslant a_2^J.$$

Since: $T_J \subseteq M_i$, $T_J \subseteq M_i$, $\forall i \in J$, it results that M_t, which is stored between the addressese a_1^t and a_2^t, is included in M_i, $\forall i \in J$, which is contrary to the assumption of the theorem.

From C3 it results that: $a_2^J < a_1^t \leqslant a_2^t < a_1^J$, which is in contradiction to (12).

Fron C4 and (14b) it results that: $a_1^J > a$ and $a_1^J > a$, which is in contradiction to (12).

These four contradictions show that (14b) isn't true. The conditions (14a) and (14 b) have resulted from assumption that the theorem isn't true. Since these two conditions are false, it resuults that the theorem is true.

THEOREM 5. *If $\mathfrak{C}$ has CR-property relative to a maximal set Q, then an atom has at most two neighbour atoms.*

Proof. If the medium is linear, each atom can be bordered by another atom on its right and left side. From this remark as well as from theorem 4 it results that this theorem is true.

DEFINITION 8. The set Q is connected relative to $\mathfrak{C}$ if for every pa-tition $\{Q_1, Q_2\}$ of Q, we have:

$$\left(\bigcup_{q\in Q_1} q(\mathfrak{C})\right) \cup \left(\bigcup_{q\in Q_2} q(\mathfrak{C})\right) \neq \Phi.$$

REMARK. By „$\mathfrak{C}$ has CR-property relative a connected set Q" we mean that $\mathfrak{C}$ has CR-property relative to Q and Q is a connected set relative to $\mathfrak{C}$.

From the definition 8 it results that a connected set Q has at least two elements (questions).

THEOREM 6. *If $\mathfrak{C}$ has CR-property relative to a maximal and connected set $Q = \{q_1, \ldots, q_n\}$, every atom with the form:*

$$T = M_{i_0} \cap \left(\bigcap_{i \neq i_0} \bar{M}_i\right); \quad M_i = q_i(\mathfrak{C}), \; i = 1, \ldots, n, \tag{15}$$

has at least one neighbour atom.

Proof. Let us suppose that atom (15) has no neighbour atoms. In that case:

$$T' = M_{i_0} \cap M_{i_1} \cap \left(\bigcap_{i \neq i_0, i_1} \bar{M}_i \right) = \Phi, \ \forall i_1 \neq i_0,$$

or:

$$M_{i_0} \cap M_{i_1} \subseteq \bigcup_{i \neq i_0, i_1} M_i, \ \forall i_1 \neq i_0. \tag{16}$$

Let $J = \{j \mid j \neq i_0, \ M_{i_0} \cap M_j \neq \Phi\}$ be.

If $J = \Phi$, then Q would not be a connected set, which is in contradiction to the theorem. Hence $J \neq \Phi$.

Let $a_1^T, a_2^T; a_1^i, a_2^i, i = 1, \ldots, n$ be the addresses situated on the extreme left and right positions where the T and M_i $(i = 1, \ldots, n)$ sets are stored on the medium $\mathfrak{S}$.

Since T, from (15), is an atom, every M_i $(i \neq i_0)$ set is stored on the left $(a_1^T < a_2^i)$ or right $(a_2^T < a_1^i)$ side of the set T. It results that at least one of the sets:

$$J_1 = \{j \mid j \in J, \ a_1^T > a_2^j\}, \ J_2 = \{j \mid j \in J, \ a_2^T < a_1^j\}.$$

is nonempty, because: $J = J_1 \cup J_2 \neq \Phi$.

We assume that $J_1 \neq \Phi$. Let a be equal to:

$$a = \max \{a_2^j \mid j \in J_1\} = a_2^s,$$

and d — the datum stored on address a.

From (16) it results that $\exists k \in J_1, \ k \neq s$, so that $d \in M_k$. Hence it results that $d \in M_k \cap M_s$ and d is the datum stored on the extreme right of the M_k and M_s data sets. Then either $M_k \subset M_s$ or $M_k \subset M_s$ contradict to the assumptions of the theorem (Q is maximal set).

If $J_1 = \Phi$, then $J_2 \neq \Phi$ and we reach the same contradiction.

These contradictions show that theorem 6 is true.

DEFINITION 9. An atom is extreme if is of the form (15) and it has a single neighbour atom.

THEOREM 7. *If $\mathfrak{C}$ has CR-property relative to a maximal and connected set $Q = \{q_1, \ldots, q_n\}$, then extreme atoms must be stored on the extremities of the medium $\mathfrak{S}$ and there cannot be other extreme atoms.*

Proof. We assume that the first (or the last) atom on the medium $\mathfrak{S}$ is no extreme atom, then it must be enlisted under (1) with $|I| > 1$. Since every M_i, $i \notin I$, is stored consecutively on the medium, starting from the most lower position, it results that: $\exists i_0 \in I, \ M_i \subseteq M_{i_0}, \ \forall i \in I$, which would contradict the assumption of the theorem. The same holds for the right extremity.

Let us assume that on the medium there is yet another extreme atom, T_1, which is not situated on either of the two extremities. Since it would have a single neighbour atom (let us say on the left), we mark with T_2 the atom situated on its right.

Thus:
$$T_1 = M_{i_1} \cap \Big(\bigcap_{i \neq i_1} \bar{M}_i\Big),$$
$$T_2 = \Big(\bigcap_{i \in I} M_i\Big) \cap \Big(\bigcap_{i \in \bar{I}} \bar{M}_i\Big), \; I \neq \{i_1\}.$$

If $i_1 \in I$ then $|I| > 2$ (T_1 and T_2 are not neighbour atoms). In that case, all $M_i (i \in I - \{i_1\})$ sets are stored on the right side, starting from the same position. Since $|I - \{i_1\}| \geqslant 2$, it results that: $\exists i_0 \in I - \{i_1\}$, $M_i \subseteq M_{i_0}$, $\forall i \in I - \{i_1\}$, which would contradict the assumption of the theorem.

If $i_1 \notin I$ and $|I| > 1$, then a similar procedure would lead to the same contradiction. So $|I| = 1$ and T_2 has the form:
$$T_2 = M_{i_2} \cap \Big(\bigcap_{i_1 \neq i_2} \bar{M}_i\Big), \; i_1 \neq i_2.$$

Let $Q_1 \subseteq Q$ be for which $M_i = q_i(\mathcal{C})$ $(q_i \in Q_1)$ is stored to the left of T_1, including T_1, and $Q_2 = Q - Q_1$. Since $\nexists i: T_1 \cup T_2 \subseteq M$, then $Q_1 \cap Q_2 = \Phi$, from which it results that Q isn't conex. This contradiction shows that the theorem is true.

DEFINITION 10. Two atoms T_I and T_J of form (1) are similar if the following conditions are fulfiled:
a) T_I and T_J have no common neighbour atom;
b) $I = K \cup \{i_0\}$; $J = K \cup \{j_0\}$; $K \neq \Phi$; $i_0 \neq j_0$; $i_0, j_0 \in \bar{K}$
(J is obtained by replacing an element from I with an element from $\bar{I}$).

THEOREM 8. *If $\mathcal{C}$ has CR-property relative to a maximal set $Q = \{q_1, \ldots, q_n\}$, two similar atoms must be consecutive atoms.*

Proof. If we assume that T_I and T_J are similar atoms and that the theorem isn't true, it results that $\exists R \in \mathcal{C}$, $R \notin T_I$, $R \notin T_J$ and R is stored between T_I and T_J. Let I, J, K, i_0, j_0 be the elements which are specified to definition 10 and $M_i = q_i(\mathcal{C})$, $i = 1, \ldots, n$.

Since M_i is consecutively stored on the medium $\mathcal{S}$, $i \in K$, and: $T_I, T_J \subseteq M_i$, $\forall i \in K$, it results that:
$$R \in M_i, \; \forall i \in K. \tag{17}$$

If $\exists M_s$, $s \in \bar{I} \cap \bar{J}$ so that: $R \in M_s$, M_s must be stored between T_I and T_J, because $T_I \cap M_s = T_J \cap M_s = \Phi$ and $\mathcal{C}$ has CR-property. Hence it results that $M_s \subseteq M_i$, $\forall i \in K$, which is in contradiction to the assumption of the theorem.

Therefore:
$$R \notin M_s, \; \forall_s \in I \cap \bar{J} = \overline{K \cup \{i_0, j_0\}}. \tag{18}$$

Since $R \notin T_I$ and $R \notin T_J$, from (17) and (18) it results that:
$$R \notin M_{i_0} \text{ and } R \notin M_{j_0}; \tag{19}$$
or
$$R \in M_{i_0} \text{ and } R \in M_{j_0}. \tag{20}$$

From (17), (18) and (19) it results that:

$$R \in \left(\bigcap_{i\in K} \bar{M}_i\right) \cap \left(\bigcap_{i\in \overline{K \cup \{i_0, j_0\}}} \bar{M}_i\right) \cap \bar{M}_{i_0} \cap \bar{M}_{j_0} =$$

$$= \left(\bigcap_{i\in K} M_i\right) \cap \left(\bigcap_{i\in \bar{K}} \bar{M}_i\right) = T' ;$$

and from (17), (18) and (20), it results that:

$$R \in \left(\bigcap_{i\in K\cup\{i_0, j_0\}} M_i\right) \cap \left(\bigcap_{i\in \overline{K\cup\{i_0, j_0\}}} \bar{M}_i\right) = T''.$$

Hence T' (or T'') is a nonempty atom and it is neighbour with T_I and T_J, which would be in contradiction to the theorem (T_I and T_J are similar atoms). This contradiction shows that the theorem is true.

THEOREM 9. *If $\mathfrak{C}$ has CR-property relative to a maximal set Q, an atom has at most two similar atoms.*

Proof. Near an atom can be stored at most two atoms. This remark and theorem 8 prove this theorem.

THEOREM 10. *If $\mathfrak{C}$ has CR-property relative to a maximal and connected set $Q = \{q_1, \ldots, q_n\}$, then two consecutive atoms are neighbour or similar atoms.*

Proof. Let T_I and T_J be two consecutive atoms, T_I stored before of T_J, of form (1). Let also assume that T_I and T_J aren't neighbour or similar atoms and $M_i = q_i(\mathfrak{C})$, $i = 1, \ldots, n$.

If $I \cap J = \Phi$, the storing of all $M_i (i \in I)$ sets would end at the same address. If $|I| > 1$, $\exists i_1 \in I$ so that: $\forall i \in I$, $i \neq i_1$, $M_i \subseteq M_{i_1}$, it results that the Q set isn't maximal. Therefore $|I| = 1$. In a similar way it can be proved that $|J| = 1$. But in this case ($|I| = |J| = 1$) the collection Q is no a connected set (T_I and T_J are extreme atoms. From theorem 7 it results that there are only two extreme atoms, which on stored on the extreme sides). It results that $I \cap J \neq \Phi$.

Let us take $K = I \cap J$, $I = K \cup I_1$, $J = K \cup J_1$, where $I_1 \neq J_1$.

We have the following possibilities: $|I_1| = 0$, $|I_1| = 1$ and $|I_1| > 1$.

If $|I_1| = 0$ ($I_1 = \Phi$) and $|J_1| > 1$, all $M_i (i \in J_1)$ sets on stored starting from the same address, so that $\exists j_1 \in J_1$, and $M_j \subseteq M_{j_1}$, $\forall j \in J_1 - \{j_1\}$ which is in contradiction to the assumption of the theorem. It results that $|J_1| = 1$, but T_I and T_J are neighbour atoms, which is in contradiction to the previous assumption. In that case $|I_1| = 0$ is false.

If $|I_1| > 1$, all $M_i (i \in I_1)$, sets are stored so that their last address is common. It results that the assumption of the theorem is false.

Hence it results that the last case: $|I_1| = 1$ is true.

Similary, one can prove that $|J_1| = 1$.

But in this case ($|I_1| = |J_1| = 1$), the atoms T_I and T_J are similar (atoms), which would contradit the previous assumption, and prove that the theorem is true.

REMARK. If $\mathfrak{C}$ has CR-property relative to a maximal and connected set Q, every atom T which isn't extreme has two consecutive atoms, which are neighbour or similar to T.

We use the previous theorems for an algorithm which determines the order in which the data collection endowed with the CR-property relative to a maximal and connected set Q must be stored.

Algorithm:

1. If Q has only one element, then $\mathcal{C}$ can be stored in any succession.
2. An extreme atom T_1 is determined and stored on the medium $\mathcal{S}$ (from theorem 7 it results that there are two extreme atoms).
3. For the atom T_1 its neighbour atom T_2 is detemined and stored on the medium $\mathcal{S}$ (from definition 9 it results that there is only one neighbour atom).
4. For atom T_2 the T_3 atom must be determined. These atoms must be neighbour or similar ones. T_3 must be stored on the medium $\mathcal{S}$ (from the last remark it results that there are two atoms of the same kind out of which T_1 is already stored).
5. If T_3 is an extreme atom, then Stop; otherwise: $T_1 := T_2$, $T_2 := T_3$ and go to 4.

This algorithm holds good only for maximal and connected Q sets relative to data collection $\mathcal{C}$. These restrictions are very strong, therefore we shall change this algorithm so that it may hold good for any data collection $\mathcal{C}$ which has CR-property relative to a set Q.

DEFINITION 11. For the data collection $\mathcal{C} = \{d_1, \ldots, d_m\}$ and the question set $Q = \{q_1, \cdots, q_n\}$ we consider the digraph $G_1 = (X, U)$ in which:

a) For every $q_i \in Q$ we consider a vertex $x_i \in X$ (we assume that any elements from Q are different);

b) $(x_i, x_j) \in U$ if $q_j(\mathcal{C}) \subset q_i(\mathcal{C})$.

For the digraph G_1 we determine the following sets:

$$\left.\begin{aligned} I_1 &= \{i \mid i \in \{1, 2, \ldots, n\},\ \nexists\, j \in \{1, 2, \ldots, n\},\ j \neq i:\ (x_j, x_i) \in U\};\\ Q_1 &= \{q_i \mid q_i \in Q,\ i \in I_1\};\\ \mathcal{C}_1 &= \bigcup_{q \in Q} q(\mathcal{C}). \end{aligned}\right\} \tag{21}$$

THEOREM 11. a) *The digraph G_1 is acyclic (it has no cycles);*

b) *Q_1 is a maximal set relative to $\mathcal{C}$;*

c) $\mathcal{C}_1 = \mathcal{C}$.

Proof. a) If $\mathcal{C}_1$ has at least one cycle $\mu = [x_{i_1}, x_{i_2}, \ldots, x_{i_p}, x_{i_1}]$, with $p \geqslant 2$, then the following conditions are true:

$$q_{i_1}(\mathcal{C}) \supseteq q_{i_2}(\mathcal{C}) \supseteq \ldots \supseteq q_{i_p}(\mathcal{C}) \supseteq q_{i_1}(\mathcal{C}).$$

From these conditions it results that:

$$q_{i_1}(\mathcal{C}) = q_{i_2}(\mathcal{C}) = \ldots = q_{i_p}(\mathcal{C}),$$

which is in contradiction to the theorem (Q has different elements).

b) Since a from this theorem is true, it results that: $I_1 \neq \Phi$. If Q_1 is no maximal set relative to $\mathcal{C}$, then:

$$\exists q_i, q_j \in Q_1 : q_i(\mathcal{C}) \subseteq q_j(\mathcal{C}).$$

or:

$$\exists i, j \in I_1: (x_j, x_i) \in U,$$

which is in contradiction to the definition of I_1 from (21).

c) From the definition of $\mathcal{C}_1$ it results that $\mathcal{C}_1 \subseteq \mathcal{C}$. If $d \in \mathcal{C}$ then $\exists i: d \in q_i(\mathcal{C})$. Let $q_{i_0}(\mathcal{C})$ be a maximal element of the set: $\{q_i(\mathcal{C}) \mid d \in q_i(\mathcal{C})\}$. Hence:

$$\forall i \in \{1, 2, \ldots, n\}, i \neq i_0 : q_{i_0}(\mathcal{C}) \notin q_i(\mathcal{C}),$$

and $i_0 \in I_1$; $q_{i_0} \in Q_1$, or: $d \in q_i(\mathcal{C}) \subseteq \mathcal{C}_1$. From this remark it results that $\mathcal{C} \subseteq \mathcal{C}_1$, and c from the theorem is true.

Next we shall take into account the subgraphs G_k of G, in such a way that their vertex sets are:

$$X - \left\{x_i \mid i \in \bigcup_{j}^{k-1} I_j\right\}.$$

For these digraphs G_k we consider the following sets: $I_k, Q_k, \mathcal{C}_k$, $k \geq 2$, so that:

$$\left.\begin{aligned} I_k &= \left\{i \mid i \in V = \{1, 2, \ldots, n\} - \bigcup_{j=1}^{k-1} I_j, \nexists j \in V, j \neq i : (x_j, x_i) \in U\right\}; \\ Q_k &= \{q_i \mid q_i \in Q, i \in I_k\}; \\ \mathcal{C}_k &= \bigcup_{q \in Q_k} q(\mathcal{C}). \end{aligned}\right\} \quad (22)$$

Let us consider that the set of vertices in G_{p+1} is empty.

THEOREM 12. a) $\bigcup_{j=1}^{p} I_j = \{1, 2, \ldots, n\}$ *and* $I_j \cap I_k = \Phi$ *for* $j \neq k$;

b) $Q = \bigcup_{j=1}^{p} Q_j$ *and* $Q_j \cup Q_k = \Phi$ *for* $j \neq k$;

Any Q_j *is a maximal set relative to* $\mathcal{C}_j$, $j = 1, \ldots, p$;

c) $\mathcal{C}_j \subseteq \mathcal{C}_{j-1}$, $j = 2, \ldots, p$.

Proof. a and b results from the theorem 11 and (22).

d) Let $d \in \mathcal{C}_j$ be, $j \geq 2$. Hence:

$$\exists k \in I_j : d \in q_k(\mathcal{C}).$$

Since $k \in I_j$ it results that:

$$\exists s \in I_{j-1} : (x_s, x_k) \in U \text{ or } q_k(\mathcal{C}) \cup q_s(\mathcal{C}).$$

Since $s \in I_{j-1}$ it results that:

$$d \in q_k(\mathcal{C}) \subset q_s(\mathcal{C}) \subseteq \mathcal{C}_{j-1}.$$

THEOREM 13. *Let* $\mathcal{C}$ *be a data collection which has the CR-property relative to a connected set* $Q = \{q_1, \ldots, q_n\}$, *and* T_i, $i \in I = \{1, 2, \ldots, s\}$ *the atoms which determine the order in wich the data collection* $\mathcal{C}$ *must be stored. If we add* *a new question* q *to* Q, *with the answer* $q(\mathfrak{C}) = M \subseteq \bigcup_{i=1}^{s} q_i(\mathfrak{C})$, *the CR-property relative to* $Q \cup \{q\}$ *remains the same if and only if we have:*

$$\left.\begin{aligned} &\exists i, j \in I : M \subseteq \bigcup_{k=i}^{j} T_k, \\ &\text{if } i + 1 \leqslant j - 1 \text{ then } T_k \subseteq M,\ k = i + 1, \ldots, j - 1. \end{aligned}\right\} \qquad (23)$$

Proof. Necessity. Since Q is a connected set relative to $\mathfrak{C}$, it results that the order of atoms T_i, $i \in I$, is unique. Let a_1^M and a_2^M be the addresses situated on the extreme left and right position where the M set is stored on the medium $\mathfrak{S}$ (M is consecutively stored). Let T_i be the atom which contains the datum from the a_1^M address, and T_j the atom which contains the datum from the a_2^M address. From this remark it results that the condition in the theorem is true.

Sufficiency. If the condition from the theorem is true, then the following nonempty sets are the atoms for $\mathfrak{C}$ relative to $Q \cup \{q\}$, and on the medium $\mathfrak{S}$ these atoms on stored in the following order:

$$T_1, \ldots, T_{i-1},\ T_i \cap M,\ T_{i+1}, \ldots, T_{j-1},\ T_j \cap M,\ T_{j+1}, \ldots, T_s.$$

Let $\mathfrak{C}$ be a data collection having CR-property relative to Q and

$$\mathfrak{C} = \bigcup_{i=1}^{r} \mathfrak{C}_i,\quad Q = \bigcup_{i=1}^{r} Q_i,$$

a partition for $\mathfrak{C}$ and Q, so that $\mathfrak{C}_i$ is a connected set relative to Q_i, $i = 1, \ldots, r$. From theorem 13 it results that if a new question q is added to Q, with the answer $M \subseteq \mathfrak{C}$, then the CR-property for $\mathfrak{C}$ relative to $Q \cup \{q\}$ remains the same if and only if $\exists s$, $M \subseteq \mathfrak{C}_s$. If $T_1, \ldots, T_p$ are the atoms for the data collection $\mathfrak{C}_s$, then the condition (23) is true.

THEOREM 14. *Let* $\mathfrak{C}_1$ *be a data collection with CR-property relative to* Q_1 *and* $T_1, \ldots, T_s$ *the atoms that determine the order in wich* $\mathfrak{C}_1$ *must be stored on the medium* $\mathfrak{S}$. *Let us suppose that the data collection* $\mathfrak{C}_2 \subseteq \mathfrak{C}_1$ *has CR-property relative to a connected set* Q_2 *and* $T_1', \ldots T_r'$ *the atoms that determine the order in which* $\mathfrak{C}_2$ *must be stored on* $\mathfrak{S}$. *The data collection* $\mathfrak{C}_1$ *has CR-property relative to* $Q_1 \cup Q_2$ *if and only if the following two conditions are true:*

1. $\exists (i_k, j_k)$, $k = 1, \ldots, r$, $1 \leqslant i_1 \leqslant j_1 \leqslant i_2 \leqslant \ldots \leqslant i_r \leqslant j_r \leqslant s$,
or $1 \leqslant i_r \leqslant j_r \leqslant i_{r-1} \leqslant \ldots \leqslant i_1 \leqslant j_1 \leqslant s$,

so that $\forall k = 1, 2, \ldots, r$:

$$\begin{cases} T_k' \subseteq \bigcup_{l=i_k}^{j_k} T_l, \\ \text{if } i_k + 2 \leqslant j_k \text{ then } T_v \subseteq T_k',\ v = i_k + 1, \ldots, j_k - 1. \end{cases}$$

2. *If* $i_0 = \min\{i_1, i_r\}$ *and* $j_0 = \max\{j_1, j_r\}$, *then for every* $j \in \{i_1, i_2, \ldots, i_r, j_1, j_2, \ldots, j_r\}$ *and* $i_0 < j < j_0$, *the condition:*

$$T_j \subseteq \bigcup_{k=1}^{r} T'_k$$

is true.

Proof. This theorem results from using theorem 13 for every atom T'_k, $k = 1, \ldots, r$, and from the fact that between two atoms T'_i and T'_{i+1} there is no datum from $\mathcal{C}_1$. Hence it results that $q(\mathcal{C}_2)$, $q \in Q_2$ is consecutively stored.

Since this theorem is true, the atoms that determine the order in which $\mathcal{C}_1$ must be stored on $\mathcal{S}$ ($\mathcal{C}_1$ has CR-property relative to $Q_1 \cup Q_2$) are the nonempty sets out of the following sets:

$$\begin{array}{l}
T_1, \ldots, T_{i_1-1}, T_{i_1} - \mathcal{C}_2, \\
T_{i_1} \cap T'_1, T_{i_1+1}, \ldots, T_{j_1+1}, T_{j_1} \cap T'_1, \\
T_{i_2} \cap T'_2, T_{i_2+1}, \ldots, T_{j_2-1}, T_{j_2} \cap T'_2, \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
T_{i_k} \cap T'_k, T_{i_k+1}, \ldots, T_{j_k-1}, T_{j_k} \cap T'_k \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
T_{i_r} \cap T'_r, T_{i_r+1}, \ldots, T_{j_r-1}, T_{j_r} \cap T'_r, \\
\qquad T_{j_r} - \mathcal{C}_2, T_{j_r+1}, \ldots, T_s.
\end{array} \tag{24}$$

REMARK. If Q_1 is no connected set relative to $\mathcal{C}_1$, but there is a $\mathcal{C}'_1 \subseteq \mathcal{C}_1$, $\mathcal{C}_2 \subseteq \mathcal{C}'_1$ and Q_1 is a connected set relative to $\mathcal{C}'_1$, then theorem 14 is true.

We use theorem 14 for an algorithm that determines the order in wich the data collection $\mathcal{C}$ must be stored, so that every $q(\mathcal{C})$ may be consecutively stored.

Algortihm:

1. The I_k, $k = 1, \ldots, p$ sets (i.e. $\mathcal{C}_k$ and Q_k, $k = 1, \ldots, p$) must be determined.
2. For the $\mathcal{C}_1$ data collection the atoms $T_1, \ldots, T_s$ that specify the order in which $\mathcal{C}_1$ is stored must be determined. The algorithm previously described is to be used.
3. We consider the collections: $\mathcal{C}_2, C_3, \ldots, \mathcal{C}_p$, in the given order. For every connected subsets $\mathcal{C} \subseteq \mathcal{C}_h$ relative to $Q' \subseteq Q_h$, $h > 1$, atoms: $T'_1, T'_2, \ldots, T'_r$ must be determined. If theorem 14 is true then the atoms for the new collections must be determined in accordance with (24).

For determining the sets: I_k, $k = 1, \ldots, p$, the following algorithm may be used. This algorithm is similar to an algorithm from [4] which is used for the division of a matrix M in the submatrices M_{ij}, with M_{ij} other that zero for $|i - j| \leqslant 1$. In this algorithm one uses the following variables:
$V = (v_1, \ldots, v_m)$, $m = |Q|$, is a vector which qualifies the

$I_k (k = 1, \ldots, p)$ sets. We have:

$$v_i = \begin{cases} 0 - \text{if } \exists j, j \neq i : q_i(\mathcal{C}) = q_j(\mathcal{C}) \text{ (the vertices corresponding to equal questions must be eliminated);} \\ 1 - \text{if } i \in I_k \ (q_i \in Q_k); \end{cases}$$

$C = (c_1, \ldots, c_m)$ is a characteristic vector for the vertices from the graph G_k;

$$c_j = \begin{cases} 1 - \text{if in graph } G_k \text{ there is a vertex for the question } q_j \in Q; \\ 0 - \text{otherwise.} \end{cases}$$

$A = (a_{ij})$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, is the adjacency matrix for data collection $\mathcal{C}$ relative to set Q. In this algorithm we mark with $A._i$ the columns i and $A._j$ the column j from matrix A.

$B = (b_{ij})$, $1 \leqslant i, j \leqslant n$, is the adjacency matrix for digraph G_k; where $b_{ij} = 1$ if $(x_i, x_j) \in U$ (or $q_j(\mathcal{C}) \subseteq q_j(\mathcal{C})$)and $b_{ij} = 0$ otherwise. We mark with $B_i.$ line i from matrix B.

$E = (E_1, \ldots, E_n)$ and $D = (D_1, \ldots, D_m)$ are two vectors.

If $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_n)$ and $c = (c_1, \ldots, c_n)$, where $a_i, b_i, c_i \in \{0, 1\}$, $i = 1, \ldots, n$, then the following logical operations must be used in this algorithm:

$c = a \wedge b$ if $c_i = a \wedge b_i$, $i = 1, \ldots, n$ (conjunction);
$c = a \vee b$ if $c_i = a \vee b_i$, $i = 1, \ldots, n$ (disjunction);
$c = \bar{a}$ if $c_i = 1 - a_i$, $i = 1, \ldots, n$ (negation).

Algorithm:

```
1. For i : = 1 to n do c_i : = 1;
2. For i : = 1 to n − 1 do
     Begin b_ii : = 0;
       For j : = i + 1 to n do
         Begin D : = A.i ∧ A.j; b_ij : = 0; b_ji : = 0;
           If D = A.i and D = A.j then
             Begin v_i : = 0; c_j : = 0 end
           else If D = A.i then b_ji : = 1 [q_i(C) ⊂ q_j(C)]
         else If D = A.j then b_ij : = 1 [q_j(C) ⊂ q_i(C)]
       end
     end
3. b_nn : = 0; s : = 1; OK : = true;
4. While OK do
     Begin E : = ∨_{c_j=1} B_j ; E : = Ē ∧ C;
       For j : = 1 to n do
         If E_j = 1 then Begin v_j : = s; c_j : = 0 end;
       If C ≠ 0 then s : = s + 1 else OK : = false
     end;
5. Stop.
```

REFERENCES

1. Cardenas, A. F., *Evaluation and Selection of File Organization: A Data and System*. Comm. ACM, 16, 9, 1973, 540—548.
2. Ghosh, S. P., *File Organization: The Consecutive Retrieval Property*. Comm. ACM, 15, 9, 1972, 802—808.
3. Suzuki, S., Imai, H., *On Optimal Partition of a Query Set into Subsets Having the Consecutive Retrieval Property*. ICS PAS Reports 438, Warszawa 1981, 196—219.
4. Țâmbulea, L., Kalik, C., *Sur la résolution de quelques problèmes aux limités par la méthodes des éléments finis*. Itinerant Seminar on Functional Equations, Approximation and Convexity, Cluj-Napoca, 1983, 169—174.
5. Wong, E., Chang, T., C., *Canonical Structure Attribute Based File Organization*. Comm. ACM, 14, 9, 1971, 593—597.

STUDIA UNIV. BABEŞ–BOLYAI, MATHEMATICA, XXXII, 3, 1987

ON SOME GENERALIZATIONS OF AN OPTIMIZATION PROBLEM FOR DISTRIBUTED DATA BASES

GRIGOR MOLDOVAN* and SERGIU DAMIAN**

Received: June 10, 1987

REZUMAT. — Asupra unor generalizări ale unei probleme de optimizare pentru baze de date distribuite. În această lucrare se consideră o problemă de optimizare pentru baze de date distribuite. Se pornește de la o rețea de calculatoare în nodurile căreia sînt distribuite subbaze de date. Se rezolvă problema redistribuirii subbazelor de date în rețea, considerîndu-se două lanțuri și momentele de lansare ale deplasărilor ca făcînd parte dintr-un interval de timp dat. Se formulează și se rezolvă o problemă de max-min.

1. Introduction. Let us consider a computer net determined by a connected graph $G = (C, \bar{U})$, where $C = \{C_1, C_2, \ldots, C_n\}$ is the set of the computers in the nodes of the net, while $\bar{U}$ is the set of the edges linking the nodes of the net. We suppose that in the nodes of this net there are distributed the subbases B_k, $k = \overline{1, n}$, of a data base, namely

$$B = \{B_1, B_2, \ldots, B_n\}.$$

This distribution of the subbases in the net is determined by the permutation

$$\sigma = \begin{pmatrix} 1 & 2 & \ldots & n \\ i_1 & i_2 & \ldots & i_n \end{pmatrix}$$

with the significance that in the node C_k (which we denote by k) we have the subbase B_{i_k}, for $k = \overline{1, n}$. We consider the subbases B_i, $i = \overline{1, n}$, as independent.

We suppose that a certain application requires that the data base B have another distribution in the nodes of the net. For this purpose, a redistribution of the subbases in the net is needed. The travel of the data subbases from the initial positions towards their new positions in the net is performed by passing through adjacent nodes, during known time intervals and with a certain passing cost. The launching moment for these travels is important because there may appear crowds in some nodes of the net. Some considerations about this problem can be found in [1], [2], [3]. There also are many results, obtained in peculiar, cases, concerning the optimization of the informational flow in a computer net, as one can see in [5].

If the chain on which a subbase is travelling from the node 1 to the node i_1 is $l = (1, k_1, k_2, \ldots, k_p, i_1)$ and if we denote by t_1 the launching moment from

* *University of Cluj-Napoca, Faculty of Mathematics and Physics, 3400 Cluj-Napoca, Romania.*
** *University of Cluj-Napoca, Computing Data Center, 3400 Cluj-Napoca, Romania*

the node 1, then the time moments in the nodes of this chain will be

$$T_1 = \{t_1, t_1 + d^1_{k_1}, \ldots, t_1 + d^1_{k_p}, t_1 + d^1_{i_1}\},$$

where $d^1_{i_1}$ will represent the minimum value of the path linking the node 1 to the node i_1.

Generally, for the travel from the node k to the node i_k we have the moments

$$T_k = \{t_k, t_k + d^k_1, \ldots, t_k + d^k_{ik}\}, \qquad k = \overline{1, n}. \tag{1}$$

The quantities d^k_j are fixed, while the time moments t_k are changing. Represented onto an axis, a division T_k determines some points at known distances, these last ones being determined by the quantities d^k_j.

Measures of the crowding degree of the subbase travelling traffic in a computer net can be defined in various manners. We shall consider in this paper a global measure of the crowding degree. In this meaning we shall form onto the axis a division of the points

$$T = T_1 \cup T_2 \cup \ldots \cup T_n$$

which have the above mentioned properties.

DEFINITION 1. The *global crowding degree* of the subbase travelling traffic in a computer net is the maximum of the smallest distances between the consecutive points of the representation of the divisions (1) onto the real axis, in a given time interval, where $t_1, t_2, \ldots, t_n$ are considered variable.

2. Results for a peculiar case. In [4] there have been given some results for the above formulated problem, for the peculiar case of two chains, the values associated to the nodes of a chain being inserted one by one between the values associated to the nodes of the other chain.

Considering two chains, l_1 and l_2, of the form (1) with $n + 1$ and n nodes, respectively, we denote for simplicity:

$$a_1 = t_1, \qquad a_{i+1} = t_1 + d^1_i, \qquad i = \overline{1, n}$$

and

$$p_1 = t_2, \qquad p_{i+1} = t_2 + d^2_i, \qquad i = \overline{1, n-1}.$$

DEFINITION 2. We call *configuration* the aggregate formed by the sets of real numbers $l_1 = \{a_1, a_2, \ldots, a_{n+1}\}$ and $l_2 = \{p_1, p_2, \ldots, p_n\}$ such that

$$a_1 < p_1 < a_2 < p_2 < \ldots < a_n < p_n < a_{n+1} \tag{2}$$

$$a_{i+1} - a_i = v_{i+1,i}, \qquad i = \overline{1, n} \tag{3}$$

$$p_{i+1} - p_i = v'_{i+1,i}, \qquad i = \overline{1, n-1}, \tag{4}$$

where $v_{i+1,i}$ and $v'_{i+1,i}$ are (known) constants representing travelling durations on the corresponding chains l_1 and l_2, respectively.

The justification of the condition (2) lies in the hypothesis that the launching moments (see Section 1) lie in a given time interval, for instance $[a_1, a_{n+1}]$.

Denoting:

$$s_i = p_i - a_i, \quad d_i = a_{i+1} - p_i,$$
$$s = \min_i s_i, \quad d = \min_i d_i,$$

where $i = \overline{1, n}$, there was formulated:

PROBLEM 1. *Considering an initial configuration formed by* $l_1 = \{a_1, a_2, \ldots, a_{n+1}\}$ *and* $l_2^0 = \{p_1^0, p_2^0, \ldots, p_n^0\}$, *determine the configuration formed by* l_1 *and* $l_2 = \{p_1, p_2, \ldots, p_n\}$ *for which* $\min\{s, d\}$ *is maximum, where* l_2 *is obtained from* l_2^0 *by translations.*

For the initial configuration we denote

$$s_i^0 = p_i^0 - a_i, \quad d_i^0 = a_{i+1} - p_i^0,$$
$$s_0 = \min_i s_i^0, \quad d_0 = \min_i d_i^0,$$

where $i = \overline{1, n}$, while a_i, p_i^0 and p_i fulfil the conditions (2) — (4), d_i, p_i^0 being given and p_i being variable.

In order to prove Theorem 1 (see below) we have used:

LEMMA. *If* $s, d \in \mathbb{R}$, *then*

$$\max_{x \in \mathbb{R}} \min\{s - x, d + x\} = \max_{x \in \mathbb{R}} \min\{s + x, d - x\} = \frac{d + s}{2},$$

the first maximum being reached for $x = \frac{s - d}{2}$, *and the second maximum for* $x = \frac{d - s}{2}$.

We present further down, without proving, the results obtained in [4] for the considered peculiar case.

THEOREM 1. *Observing the conditions and notations of Problem* 1, *the following sentences hold:*

a) *the maximum value for* $\min\{s, d\}$ *is* $\frac{s_0 + d_0}{2}$;

b) *the values* p_i *for which* $\min\{s, d\}$ *is maximum are* $p_i = p_i^0 + \frac{d_0 - s_0}{2}$.

THEOREM 2. *Whatever will be the initial configration which fulfils the conditions* (2) — (4), *the solution of Problem* 1 *is unique, namely the values* p_i *for which* $\min\{s, d\}$ *is maximum are unique, while the maximum* $\frac{s_0 + d_0}{2}$ *is invariant with respect to the initial configuration.*

THEOREM 3. *Given the sets of points* $A = \{a_i \mid i = \overline{1, n+1}\}$ *and* $B = \{b_i \mid i = \overline{1, n}\}$ *on the real axis, with the conditions*

$$a_i < a_{i+1}, \quad i = \overline{1, n} \tag{5}$$

and

$$b_i < b_{i+1}, \quad i = \overline{1, n-1}, \tag{6}$$

the following sentences hold:

a) *The necessary and sufficient condition for existing* $t \in \mathbf{R}$ *such that*

$$a_i < b_i + t < a_{i+1}, \quad i = \overline{1, n},$$

is:

$$m_1 = \max_i \{a_i - b_i\} < \min_i \{a_{i+1} - b_i\} = m_2, \tag{7}$$

where $i = \overline{1, n}$, *and then* $t \in (m_1, m_2)$.

b) *If* (7) *holds, then* $\min\{s, d\}$ *is maximum for* $t_0 = \frac{m_1 + m_2}{2}$ *and the maximum value which can be reached by* $\min\{s, d\}$ *is* $\frac{m_2 - m_1}{2}$, *where*

$s = \min_i \{b_i + t - a_i\}$, $d = \min_i \{a_{i+1} - b_i - t\}$ *and* $t \in (m_1, m_2)$, *with* $i = \overline{1, n}$.

In order to prove Theorems 2 and 3, Theorem 1 was used.

Applying Theorem 3 for two systems of points a_i, b_i and a'_i, b'_i wich fulfil the conditions (5) and (6), and for which $m_1 < m_2$ and $m'_1 < m'_2$, we have:

COROLLARY. *The maximum value for* $\min\{s, d\}$ *and* $\min\{s', d'\}$ *is the same if and only if* $m_2 - m_1 = m'_2 - m'_1$.

Concluding, if the conditions (5) — (7) are fulfilled, Theorem 3 allows, starting from more general configurations than those fulfilling the condition (2) and performing a translation with $t \in (m_1, m_2)$, to have the points $p_i = b_i + t$, which, together with the points a_i, fulfil Definition 2; so we obtain the optimum required by Problem 1, for $t = t_0 = \frac{m_1 + m_2}{2}$. Also, according to Theorem 3 and to the Corollary, the optimum we searched for depends only on m_1 and m_2, and its existence is equivalent to $m_1 < m_2$.

3. An extension of the results. Leaving the strict inequalities of the condition (2) and replacing them by

$$a_1 \leqslant p_1 \leqslant a_2 \leqslant p_2 \leqslant \ldots \leqslant a_n \leqslant p_n \leqslant a_{n+1} \tag{2'}$$

and

$$p_i < p_{i+1}, \quad i = \overline{1, n-1}, \tag{2''}$$

Problem 1 is similarly formulated; the only difference consists of the fact that the points a_i, p_i^0 and p_i fulfil the conditions (2'), (2'') instead of the condition (2).

So, Theorem 1 remains valid, the proof being the same as in [4], augmented with the following peculiar cases:

a) There exists $i \in \{1, 2, \ldots, n\}$ such that $p_i^0 = a_i$ and there exists $j \in \{1, 2, \ldots, n\}$ such that $p_{j+1}^0 = a_j$.

Taking into account (2') and (2''), it results $i \neq j$.

Using the notations of the previous section, we have

$$s_i = 0, \qquad d_j = 0$$

and

$$s_0 = \min_k s_k = 0; \qquad d_0 = \min_k d_k = 0,$$

where $k = \overline{1, n}$.

In this case, the translation of the points p_k^0 is performed with $\frac{d_0 - s_0}{2} = 0$, the maximum value for $\min\{s, d\}$ being $\frac{s_0 + d_0}{2} = 0$.

Hence the optimal configuration is that given, but a non-vanishing minimum distance does not exist in the given conditions, for every translation of the points p_k^0 the condition (2′) being not fulfilled.

b) There exists $i \in \{1, 2, \ldots, n\}$ such that $p_i^0 = a_i$, and there exists not $j \in \{1, 2, \ldots, n\}$ such that $p_j^0 = a_{j+1}$.

Then we have $s_i = 0$ and $s_0 = \min_k s_k = 0$, while $d_0 = \min d_k > 0$, where $k = \overline{1, n}$.

In this case, the maximum value for $\min\{s, d\}$ is $\frac{s_0 + d_0}{2} = \frac{d_0}{2}$ and this holds for a translation of the points p_k^0 with $\frac{d_0 - s_0}{2} = \frac{d_0}{2}$.

c) There exists $i \in \{1, 2, \ldots, n\}$ such that $p_i^0 = a_{i+1}$, and there exists not $j \in \{1, 2, \ldots, n\}$ such that $p_j^0 = a_j$.

Then we have $d_i = 0$ and $d_0 = \min_k d_k = 0$, while $s_0 = \min_k s_k > 0$, where $k = \overline{1, n}$.

In this case, the maximum value for $\min\{s, d\}$ is $\frac{s_0 + d_0}{2} = \frac{s_0}{2}$ and this holds for $\frac{d_0 - s_0}{2} = -\frac{s_0}{2}$, corresponding to a translation to the left of the points p_k^0 with $\frac{s_0}{2}$.

Analogously to the extension of Theorem 1, Theorems 2 and 3 can be naturally extended; for Theorem 2, (2) is replaced by (2′) and (2″), while for Theorem 3 the strict inequalities of a) are replaced by non-strict inequalities.

So, Theorems 1, 2, 3 and the Corollary also hold in the above supposed less restrictive conditions.

4. General case and solving algorithm. Let be $m, n \in \mathbf{N}$, $m \geq 1$, $n \geq 2$. We consider two chains, l_1 and l_2, with $n + 1$, respectively m nodes, and the conditions:

$$a_1 \leq p_{11} < p_{12} < \ldots < p_{1j_1} \leq a_2 \leq p_{21} < p_{22} < \ldots < p_{2j_2} \leq a_3 \leq$$
$$\leq p_{31} < \ldots < p_{n-1, j_{n-1}} \leq a_n \leq p_{n1} < p_{n2} < \ldots < p_{nj_n} \leq a_{n+1}, \qquad (8)$$

where $j_1 + j_2 + \ldots + j_n = m$;

$$p_{ij_i} < p_{i+1,1}, \quad i = \overline{1, n-1}. \tag{9}$$

The condition (8) determines a division of the real values p_{ij} into n classes, denoted as follows:

$$M_i = \{p_{i1}, p_{i2}, \ldots, p_{ij_i}\}, \quad i = \overline{1, n};$$

some of these classes can be empty.

Taking into account (9), it results that there are not i and j_i such that $p_{ij_i} = a_{i+1} = p_{i+1,1}$, hence, without restricting the generality, we can replace (8) by

$$a_1 \leqslant p_{11} < p_{12} < \ldots < p_{1j_1} < a_2 \leqslant p_{21} < p_{22} < \ldots < p_{2j_2} < a_3 \leqslant$$
$$\leqslant p_{31} < \ldots < p_{n-1,j_{n-1}} < a_n \leqslant p_{n1} < p_{n2} < \ldots < p_{nj_n} \leqslant a_{n+1}. \tag{10}$$

We associate to each class M_i a fictitious point p'_i such that

$$a_1 \leqslant p'_1 < a_2 \leqslant p'_2 < \ldots < a_n \leqslant p'_n < a_{n+1}; \tag{11}$$

$$s_i = p'_i - a_i = \begin{cases} p_{i_1} - a_i, & \text{if } M_i \neq \Phi \\ \infty, & \text{if } M_i = \Phi; \end{cases} \tag{12}$$

$$d_i = a_{i+1} - p'_i = \begin{cases} a_{i+1} - p_{ij_i}, & \text{if } M_i \neq \Phi \\ \infty, & \text{if } M_i = \Phi; \end{cases} \tag{13}$$

$$s = \min_i s_i, \quad i = \overline{1, n}; \tag{14}$$

$$d = \min_i d_i, \quad i = \overline{1, n}. \tag{15}$$

The case in which $a_1 = p_{11}$ and $a_{n+1} = p_{nj_n}$ simultaneously is not of interest, therefore we shall consider

$$a_n - a_1 > p_{nj_n} - p_{11}. \tag{16}$$

Also, for two chains l_1 and l_2 whose values in the nodes fulfil (16) but do not fulfil (10), one performs a translation such that (10) holds.

In the general case, for two chains, we give:

DEFINITION 3. We call *extended configuration* the set formed by the sets of real numbers

$$l_1 = \{a_1, a_2, \ldots, a_{n+1}\} \text{ and } l_2 = \{p_1, p_2, \ldots, p_m\}$$

such that:

$$a_1 < a_2 < \ldots < a_{n+1}; \tag{17}$$

$$p_1 < p_2 < \ldots < p_m; \tag{18}$$

$$a_1 \leqslant p_1, p_m \leqslant a_{n+1} \text{ and } a_{n+1} - a_1 > p_m - p_1; \tag{19}$$

$$a_{i+1} - a_i = v_{i+1,i}, \quad i = \overline{1, n};$$

$$p_{i+1} - p_i = v'_{i+1,i}, \quad i = \overline{1, m-1},$$

where $v_{i+1,i}$ and $v'_{i+1,i}$ are strictly positive real constants.

In the conditions (17)—(19), equivalent to (10) and (16), — taking into account the above considerations, one determines the classes M_i, $i = \overline{1, m}$, on the set of points p_i, $i = \overline{1, m}$.

We formulate:

PROBLEM 2. *Considering an extended configuration given by* $l_1 = \{a_1, a_2, \ldots, a_{n+1}\}$ *and* $l_2^0 = \{p_1^0, p_2^0, \ldots, p_m^0\}$, *which determines the set of classes* $\mathfrak{M}_0 = \{M_1^0, M_2^0, \ldots, M_n^0\}$, *determine the extended configuration, formed by* l_1 *and* $l_2 = \{p_1, p_2, \ldots, p_m\}$, *or the set of classes* $\mathfrak{M} = \{M_1, M_2, \ldots, M_n\}$ *for which* $\min\{s, d\}$ *is maximum when* l_2 *is obtained from* l_2^0 *by trnaslations.*

So, a_i and p_i^0 are fixed, while p_i are variable, being obtained from p_i^0 by translations, observing Definition 3; s and d are given by (14) and (15), respectively.

Associating the points p_i' to the classes M_i, according to the above exposed method (conditions (11)—(13)), the sets l_1 and $l_2' = \{p_1', p_2', \ldots, p_n'\}$ will be according to the extensions of Section 3, on the basis of which we give further down an algorithm which will provide the optimum required by Problem 2. The algorithm involves the following basic steps:

Step 1. Initialize the configuration counter, the variable for the optimum of the problem and the variables for the optimum of each configuration.

Step 2. If $a_1 < p_1$, perform a translation to the left of the points p_i with $p_1 - a_1$: $p_i := p_i - p_1 + a_1$, $i = \overline{1, m}$ (one obtains the initial configuration with $s_0 = 0$).

Step 3. While $p_m < a_{n+1}$ do:

Substep 3.1. Add 1 to the configuration counter.

Substep 3.2. Initialize the array indices.

Substep 3.3. Determine the set of classes $\mathfrak{M}_c = \{M_{ic} \mid i = \overline{1, n}\}$ corresponding to the current configuration and associate to them the points p_i'.

Substep 3.4. Determine the optimum m_c for the current configuration.

Substep 3.5. Translate the points p_i, $i = \overline{1, m}$, with $2m_c$ (one obtains a new configuration with $s_0 = 0$).

Step 4. Determine the required optimum as being $\max\{m_k\}$, $k = \overline{1, c}$, and the configuration or configurations for which it is obtained.

Starting from an initial configuration which fulfils the conditions of Definition 3, one translates eventually the points p_i in order to obtain $a_1 = p_1$, hence a configuration for which $s_0 = 0$. After determining the optimum for the current configuration, at the Substep 3.5 one translates the points p_i so that the right-hand terminal point of the class (the point of the class to which the maximum value is associated) or classes which determine the optimum coincides with the right-hand terminal point of the interval or intervals $[a_i, a_{i+1}]$ which determine the respective class or classes, obtaining a new configuration. One determines the optimum for successive configurations until the whole interval $[a_1, a_{n+1}]$ is run over by translations. For each configuration one obtains in this manner $s_0 = 0$, hence the case b) of Section 3. The optimum required by Problem 2 is the maximum of the optima of the successive configurations and it is obtained for at least one configuration.

The algorithm is finite; this is ensured by the fact that one can perform at most $m \times n$ translations for running over the interval $[a_1, a_{n+1}]$, therefore until $p_m = a_{n+1}$.

If one starts with the condition $a_{n+1} = p_m$, the algorithm is similar; the translations are performed to the left until $p_1 = a_1$, having for each configuration $d_0 = 0$, hence the case c) of Section 3. One obtains the same configurations, but in the inverse order, therefore, taking also into account Theorem 2, the same optimum will result.

According to Definition 1, the optimum determined in this manner will represent the global crowding degree of the data subbase travelling traffic, in a given time interval, for two chains of the net.

We present further down a detailed procedure corresponding to the above algorithm, represented under the form of pseudocode.

The significance of the data names is:

N — the number of fixed points a_i, $N = n + 1$;

M — the number of variable points p_i, $M = m$;

A — array for the values a_i, $i = \overline{1, n+1}$;

P — array for the variable values p_i, $i = \overline{1, m}$, (being initialized at the procedure input by the valuss p_i^0);

I, J, K — array indices;

C — counter for configurations;

D — the distance between the right-hand terminal point of a class and the right-hand terminal point of the interval which determines it ($D = a_{i+1} - p'_i$, according to (13));

MC — the optimum value for translating a configuration, $MC/2$ being the optimum required by the problem for a configuration ($MC = 2m_c = d$, according to (15));

MP — provides the optimum value, which is $MP/2$ for the problem ($MP = \max_{K=\overline{1,C}} \{m_K\}$, according to the Step 4);

DOP — provides the value with which the initial points p_i^0 are translated in order to obtain an optimum configuration for the problem.

With these ones, the procedure has the following form:

```
proc maxmin (N, M, A, P)
  var N, M, C, I, J, K : integer
  var A : array [1 ... N] of real
  var P : array [1 ... M] of real
  var MC : array [1 ... M×N] of real
  var MP, D, DOP : real
  begin
    C :=0
    MP :=0
```

```
for I=1,M×N
  begin
    MC[I]:=A[N]−A[1]
  end
if P[1]≠A[1]
  begin
    for I=1,M
      begin
        P[I]:=P[I]−P[1]+A[1]
      end
  end
while P[M] < A[N]
  begin
    C:=C+1
    I:=1
    K:=1
    while K≤M
      begin
        if P[K]≥A[I+1]
          begin
            if K≠1
              begin
                D:=A[I+1]−P[K−1]
                if D<MC[C]
                  begin
                    MC[C]:=D
                  end
              end
            I:=I+1
            while P[K]≥A[I+1]
              begin
                I:=I+1
              end
          end
        K:=K+1
      end
    D:=A[I+1]−P[K−1]
    if D<MC[C]
      begin
        MC[C]:=D
      end
    for J=1,M
      begin
        P[J]:=P[J]+MC[C]
      end
  end
for I=1,C
  begin
```

```
    if MC[I]>MP
      begin
        MP:=MC[I]
      end
  end
D:=MP/2
print 'MAXMIN VALUE='D
print 'OPTIMUM CONFIGURATIONS:'
D:=A[1]-P[1]
for I=1,C
  begin
    D:=D+MC[I]
    if MC[I]=MP
      begin
        DOP:=D-MC[I]/2
        print 'TRANSLATION OF P POINTS WITH 'DOP
      end
  end
end
```

Finally, we make some remarks about the procedure:

a) One supposes that N, M, A and the initial values P, are valid and fulfil the conditions of Problem 2; this fact ensures the output of the cycle WHILE into which I is incremented.

b) In (13) it is sufficient to consider the value $A[N]-A[1]$ instead of ∞.

c) At the end, the variable D was used for cumulating the values of the translations in order to obtain the translations corresponding to the optimum configurations.

REFERENCES

1. L. Gârbacea, Gr. Moldovan, *Distributed Data Bases*, Univ. of Cluj-Napoca, Fac. Math. Res. Sem., Preprint 5 (1984) (Seminar of Models, Structures and Information Processing), 70–90.
2. Gr. Moldovan, *Reorganization of a Distributed Data Base*, Univ. of Cluj-Napoca, Fac. Math. Res. Sem., Preprint 5 (1984 (Seminar of Models, Structures and Information Processing), 3–10.
3. Gr. Moldovan, *O problemă privind bazele de date distribuite*, M.E.I., A VIII-a Consfătuire de lucru şi schimb de experienţă a lucrătorilor din unităţile de informatică. Lucrări prezentate, Constanţa, 29 iulie–4 august 1985, p. 102–103.
4. Gr. Moldovan, S. Damian, *On an Optimization Problem for Distributed Data Bases*, Analele Univ. Bucureşti (to appear).
5. W. L. Price, D. W. Davies, D. L. A. Barber, C. M. Solomonides, *Teleinformatica. Reţele de calculatoare şi protocoalele lor*, Ed. Tehnică, Bucureşti, 1983.

STUDIA UNIV. BABEŞ—BOLYAI, MATHEMATICA, XXXII, 3, 1987

GENERALIZED MEANS AND DOUBLE SEQUENCES

GH. TOADER*

Received: December 2, 1986

ABSTRACT. — The properties given in (2) and (3) are considered as axioms of the generalized means. Double sequencies are defined by Eqs. (4) and (5), and their converging to a common limit is demonstrated.

In [3] was defined the following class M_s of means: a continuous mapping $m: \mathbb{R}_+^2 \to \mathbb{R}_+$ belongs to the class M_s of (symmetric) means if it satisfies the following properties:

$$x \leqslant y \Rightarrow x \leqslant m(x, y) \leqslant y$$

$$m(x, y) = m(y, x)$$

$$m(x, y) = x \Rightarrow x = y.$$

An example of mean, given also in [3], is:

$$m(x, y) = \left(\frac{x^p g(x) + y^p g(y)}{g(x) + g(y)}\right)^{1/p}, \quad p \neq 0 \tag{1}$$

where g is an arbitrary continuous mapping from $\mathbb{R}_+$ to $\mathbb{R}_+$. The choise $g(x) = 1$ gives the Minkowski means m_p. Putting $p = 1$ and then choosing $g(x) = x^{q-1}$ in (1), one obtains the means l_q introduced by Lehmer. For $q = 1$, $1/2$ and 0 respectively, l_q gives the arithmetic, the geometric and the harmonic means. Other examples of means may be found in [8] and [9].

We must remark that the class M_s not contains weighted means which are non-symmetric. To enlarge this class, we give the following definition: a continuous mapping $m: \mathbb{R}_+^2 \to \mathbb{R}_+$ belongs to the class M of means if it satisfies the following properties:

$$\min\{x, y\} \leqslant m(x, y) \leqslant \max\{x, y\}, \tag{2}$$

$$m(x, y) = x \Rightarrow x = y. \tag{3}$$

Typical examples of non-symmetric means are:

$$a_p(x, y) = px + (1 - p)y$$

$$g_p(x, y) = x^p y^{1-p}$$

and

$$h_p(x, y) = 1/(p/x + (1 - p)/y)$$

* Polytechnic Institute of Cluj-Napoca, 3400 Cluj-Napoca, Romania

with $p \in (0, 1)$, that is the weighted arithmetic, geometric respectively harmonic means. For $p = 1/2$, we denote simple: $a_{1/2} = a$, $g_{1/2} = g$ and $h_{1/2} = h$.

Given two means m and m' from M and two real numbers x_0 and y_0, one can define a pair of sequences in two ways:

$$x_{n+1} = m(x_n, y_n), \quad y_{n+1} = m'(x_{n+1}, y_n) \tag{4}$$

or

$$x_{n+1} = m(x_n, y_n), \quad y_{n+1} = m'(x_n, y_n). \tag{5}$$

For $m = h$ and $m' = g$, the first way was followed by Archimedes for estimating π. The second method, for $m = a$ and $m' = g$, was used by Gauss. That is why, a pair of sequences defined by (4) is called an Archimedean double sequence, while one given by (5) is called a Gaussian double sequence.

THEOREM 1. *If (x_n), (y_n) is an Archimedean double sequence, then the sequences (x_n) and (y_n) are convergent and have a common limit $t = A(m, m'; x_0, y_0)$.*

Proof. If $x_0 \leqslant y_0$, we have:

$$x_n \leqslant x_{n+1} \leqslant y_{n+1} \leqslant y_n, \text{ for } n > 0 \tag{6}$$

hence (x_n) has a limit t and (y_n) a limit t'. By the continuity of m we have $t = m(t, t')$ and from (3) it results $t = t' = A(m, m'; x_0, y_0)$. The case $x_0 > y_0$ is similar.

We must remark that the special case when m and m' are from M, was proved in [3].

In what follows we consider the Gaussian case. We must suppose that the means m and m' are comparable or weakly comparable, that is $m(x, y) - m'(x, y)$ respectively $(x - y)[m(x, y) - m'(x, y)]$ do not change the sign. For example, any pair from a_p, g_p and h_p is comparable, but a_p and a_q (as well as g_p and g_q, or h_p and h_q) are weakly comparable for $p \neq q$. Indeed, taking for example a_p and g_p we have:

$$x < y \Rightarrow a_p(x, y) - g_p(x, y) > (1 - p)(y - x) > 0$$

and

$$x > y \Rightarrow a_p(x, y) - g_p(x, y) > p(x - y) > 0.$$

For a_p and a_q we obtain:

$$a_p(x, y) - a_q(x, y) = (p - q)(x - y)$$

and so the assertions are verified for this two pairs. For other pairs the proof may be done similarly or may be deduced from these.

THEOREM 2. *If (x_n), (y_n) is a Gaussian double sequence defined by two (weak) comparable means, then the sequences (x_n) and (y_n) are convergent and have a common limit $T = G(m, m'; x_0, y_0)$.*

Proof. If $m(x, y) \leqslant m'(x, y)$ for any positive x and y, we get (6) and the proof may be continued as that of the theorem 1. If $m(x, y) \geqslant m'(x, y)$, all the inequalities from (6) must be reversed but the proof follows on the same way. If m and m' are weakly comparable, the proof is similar of $(x-y)[m(x,y)-m'(x,y)]>0$, but if $(x-y)[m(x,y)-m'(x,y)]<0$, we get:

$$x_0 < y_1 < x_2 < \ldots < y_2 < x_1 < y_0$$

if $x_0 < y_0$ or the opposite inequalities if $x_0 > y_0$. Hence the sequences (x_{2n}) and (y_{2n+1}) have the same limit T, while (x_{2n+1}) and (y_{2n}) converge to T'. From $m'(x_{2n}, y_{2n}) = y_{2n+1}$ we get $T' = T$.

Let us note that the limit is also known in some cases. So, in [1] we can found $A(a, g; x_0, y_0)$ and $G(a, g; x_0, y_0)$. The first result was given by Pfaff and Borchardt and the second by Gauss. In [7] it is determined $A(h, g; x_0, y_0)$ but, as it was remarked in [6], we have:

$$A(h, g; x_0, y_0) = 1/A(a, g; 1/x_0, 1/y_0)$$

and

$$G(h, g; x_0, y_0) = 1/G(a, g; 1/x_0, 1/y_0).$$

Also in [4] it is studied (asymptotically) $A(a, h; x_0, y_0)$ while in [1] is given $G(a, h; x_0, y_0)$. We note also that in [2] is studied $G(a, g; x_0, y_0)$ for complex x_0 and y_0.

As concerns non-symmetric means, in [5] it is given the following conjecture of G. D. Song:

$$G(a_p, g_p; x_0, y_0) = \pi \Big/ \int_{-\infty}^{\infty} \frac{dt}{(t^2+x_0^2)^p (t^2+y_0^2)^{1-p}}.$$

We try now to evaluate also the limit in some simple cases. We begin with $A(a_p, a_q; x_0, y_0)$. From:

$$x_{n+1} = px_n + (1-p)y_n, \quad y_{n+1} = qx_{n+1} + (1-q)y_n$$

if we put:

$$x_n = u_n(x_0 - y_0) + y_0; \quad y_n = v_n(x_0 - y_0) + y_0$$

we have:

$$u_{n+1} = pu_n + (1-p)v_n, \quad v_{n+1} = pqu_n + (1-pq)v_n.$$

Hence:

$$u_{n+1} - v_{n+1} = p(1-q)(u_n - v_n)$$

or

$$u_n - v_n = p^n(1-q)^n$$

and from:

$$v_{n+1} = v_n + pqp^n(1-q)^n$$

we have finally:

$$x_n = pq(x_0-y_0)/(1-p+pq) + (1-p)p^n(1-q)^n(x_0-y_0)/(1-p+pq) + y_0$$

that is:

$$A(a_p, a_q;\ x_0, y_0) = (pqx_0 + (1 - p)y_0)/(1 - p + pq).$$

For $p = q = 1/2$, the result is given in [4].

Making the same computations for the logarithms of x_0 and y_0, we have:

$$A(g_p, g_q;\ x_0, y_0) = (x_0^{pq}y_0^{1-p})^{1/(1-p+pq)}$$

and analogously, using $1/x_0$ and $1/y_0$:

$$A(h_p, h_q;\ x_0, y_0) = (1 - p + pq)/(pq/x_0 + (1 - p)/y_0).$$

To compute $G(a_p, a_q;\ x_0, y_0)$, from:

$$x_{n+1} = px_p + (1 - p)y_n,\quad y_{n+1} = qx_n + (1 - p)y_n$$

we have:

$$x_{n+1} - y_{n+1} = (p - q)(x_n - y_n) = (p - q)^{n+1}(x_0 - y_0).$$

Hence:

$$y_{n+1} = q(p - q)^n(x_0 - y_0) + y_n = q(x_0 - y_0)(1 - (p - q)^{n+1})/(1 - p + q) + y_0$$

that is:

$$G(a_p, a_q;\ x_0, y_0) = (qx_0 + (1 - p)y_0)/(1 - p + q). \qquad (7)$$

Analogously:

$$G(g_p, g_q;\ x_0, y_0) = (x_0^q y_0^{1-p})^{1/(1-p+q)}$$

and

$$G(h_p, h_q;\ x_0, y_0) = (1 - p + q)/(q/x_0 + (1 - p)/y_0).$$

The case $p = q = 1/2$ is now trivially, the sequences being constant.

Of course, the limit A is non-symmetric but:

$$G(m, m';\ x, y) = G(m', m;\ x, y) = G(m, m';\ y, x)$$

if m and m' are from M_s. From (7) we can see that the property is not valid on M.

REFERENCES

1. B. C. Carlson, *Algorithmus involving arithmetic and geometric means*, Amer. Math. Monthly, **78** (1971), 496–505.
2. D. A. Cox, *The arithmetic-geometric mean of Gauss*, L'Enseign. Math., **30** (1984), 275–330
3. D. M. E. Foster, G. M. Phillips, *A generalization of the Archimedean double sequence*, J. Math. Anal. Appl., **101** (1984), 575–581.
4. D. M. E. Foster, G. M. Phillips, *The arithmetic-harmonic mean*, Math. of Comp., **42** (1984), Nr. 165. 183–191.
5. A. Giroux, Q. I. Rahman, *Research problems in function theory*, Ann. Sc. Math. Quebec, **6** (1982), 1, 71–79.
6. G. Miel, *Of calculation past and present: the archimedean algorithm*, Amer. Math. Monthly, **90** (1983), 17–35.
7. G. M. Phillips, *Archimedes he numerical analyst*, Amer. Math. Monthly, **88** (1981), 3, 165–169.
8. A. O. Pittenger, *The symmetric, logarithmic and power means*, Univ. Beograd. Publ. Elektrotehn. Fak., 678–715 (1980), 19–23.
9. K. B. Stolarsky, *Generalizations of the logarithmis mean*, Math. Mag., **48** (1975), 87–92.

INTREPRINDEREA POLIGRAFICĂ CLUJ,
Municipiul Cluj-Napoca, Cd. nr. 501/1987

În cel de al XXXII-lea an (1987) *Studia Universitatis Babes-Bolyai* apare în specialitățile:

matematică
fizică
chimie
geologiie-geografie
biologie
filosofie
științe economice
științe juridice
istorie
filologie

In the XXXII-nd year (1987) of its publication, *Studia Universitatis Babeș-Bolyai* is issued as follows:

mathematics
physics
chemistry
geology-geography
biology
philosophy
economice sciences
juridical sciences
history
philology

Dans sa XXXII-e année (1987), *Studia Universitatis Babeș-Bolyai* parait dans les spécialités:

mathématiques
physique
chimie
géologie-géographie
biologie
philosophie
sciences économiques
sciences juridiques
histoire
philologie

43 875

Abonamentele se fac la oficiile poștale prin factorii poștali și prin difuzorii de presă, iar pentru străinătate prin, „ROMPRESFILATELIA", sectorul export-import presă, P. O. Box 12—201, telex 10376 prsfir, București, Calea Griviței nr. 64—66.

Lei 35