STUDIA
UNIVERSITATIS BABEŞ-BOLYAI

MATHEMATICA

3

1991

CLUJ-NAPOCA

REDACTOR ŞEF: **Prof. I. HAIDUC, membru corespondent al Academiei Române**

REDACTORI ŞEFI ADJUNCŢI: **Prof. A. MAGYARI, prof. P. MOCANU, conf. M. PAPAHAGI**

COMITETUL DE REDACŢIE AL SERIEI MATEMATICĂ: **Prof. M. BALÁZS, prof. GH. COMAN, prof. I. MUNTEAN, prof. A. PÁL, prof. I. PURDEA, prof. I. A. RUS (redactor coordonator), prof. D. STANCU, prof. M. ŢARINA, conf. M. FRENŢIU, conf. T. PETRILA, lector FL. BOIAN (secretar de redacţie — informatică), lector R. PRECUP (secretar de redacţie — matematică)**

Anul XXXVI 1991

S T U D I A

UNIVERSITATIS BABEŞ-BOLYAI

MATHEMATICA

3

R e d a c ţ i a : 3400 CLUJ-NAPOCA str. M. Kogălniceanu nr.1 ▸ Telefon: 116101

S U M A R - C O N T E N T S - S O M M A I R E

IOANA MARIA BOIER, A Binary Tree Clasifier Based on Fuzzy Sets ■ Arbore binar de clasificare bazat pe mulţimi fuzzy..... 3

V. CIOBAN, V. PREJMEREAN, S. MOTOGNA, On Independent Sets of Graphs ■ Asupra seturilor independente de grafe.....11

GHEORGHE COMAN, On Some Parallel Methods in Linear Algebra ■ Asupra unor metode paralele în algebra liniară.....17

SEVER GROZE, On the Convergence of the Three-order Methods in Fréchet spaces ■ Asupra convergenţei metodelor de ordinul trei în spaţii Fréchet..35

CRISTIAN LENART, Software for Classification ■ Software pentru clasificare 41

CRISTIAN LENART, Topographical Data Management System ■ Sisteme de gestiune a datelor topografice.....51

ILIE PARPUCEA, Theoretical Support for Object-Oriented and Parallel Programming ■ Un model teoretic privind programarea paralelă şi orientată-obiect.....61

DRAGOŞ POP, Formal Integration of Certain Classes of Functions ■ Integrarea formală a unor clase de funcţii.....73

DOINA TĂTAR, A New Method for the Proof of Theorems ■ O metodă nouă de demonstrare a teoremelor.....83

P.241 93

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

A BINARY TREE CLASIFIER BASED ON FUZZY SETS

IOANA MARIA BOIER*

Dedicated to Professor P. T. Mocanu on his 60-th aniversary

Received: February 4, 1991
AMS subject classification: 68P99

Rezumat. Arbore binar de clasificare bazat pe mulţimi fuzzy. În această lucrare este descris un algoritm de proiectare şi implementare a unui clasificator binar. Acest algoritm îşi propune îmbunătăţirea algoritmului propus de Fu şi Mui [3]. O mulţime de date de test este utilizată în construcţia clasificatorului. În abordarea acestei probleme, Fu şi Mui folosesc proiecţia datelor în plan şi inspecţia vizuală ca metode de separare a clusterilor. Abordarea de faţă propune o separare automată, bazată pe mulţimi fuzzy.

The design of the binary tree classifier.

A method to design a binary tree classifier has been proposed in [3]. According to Fu and Mui, there are three major tasks to be implemented, to design a binary tree classifier:

a) a tree skeleton or hierarchical ordering of class labels

b) the choice of features at each nonterminal node

c) the decision rule to be used at each nonterminal node.

These tasks involve the specification of the following parameters:

a) the number of descendant nodes at each nonterminal node

b) the number of features used at each nonterminal node

c) an appropriate decision rule to be considered at each nonterminal node.

Since any conventional single stage classification scheme can be represented by a binary tree classifier which has exactly two immediat descendant nodes for each nonterminal node [3], we

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

consider the number of descendant nodes at each nonterminal node to be two. The next parameter to be specified is the maximum number of features used at each nonterminal node. This number depends on the specific classification problem and it is a constant for the problem. Let us denote it by K. To determine K, the number of all features, the size of test sample and the average number of samples per class are to be considered. The decision rule chosen at each nonterminal node is:

if $d(X,L^1) \leq d(X,L^2)$ then X is classified into calss A_1 (1)

otherwise X is classified into class A_2,

where X is the feature vector of the unknown sample to be classified, L^i is the prototype of the class A_i (i=1,2) [1] and d is a norm induced by distance in $\mathbb{R}^p$:

$$d(x,y)=\|x-y\| .$$

The next steps we have to perform are to design the tree skeleton or hierarchical ordering of class labels and to establish the actual features used at each nonterminal node. The fundamental problem which appears when the tree skeleton is built is the separation of the two groups of classes in each nonterminal node and the choice of features which are effective in separating these groups of classes. But, generally, the choise of the most effective features depends on the classes to be separated and the separation of the classes depends on what features are used. A method to break this deadlock is proposed in what follows. Using General Fuzzy Isodata algorithm [1] a fuzzy class is divided into two groups. Then, a method similar to the one presented by Fu and Mui [3] is used to chose the features which are "most effective"

in separating the two groups of classes.

Let us assume that the predetermined number of classes is n and that the classes are labeled 1, 2,...,n. We also assume that the dimension of the features space is p. Suppose we have reached with the construction of the tree skeleton to a nonterminal node. Let C be the fuzzy set describing the membership degrees of class label i to this node, for all i from 1 to n. For example, at the beginning, when the nonterminal node is the root, the membership degrees are $C(i)=1.0$ for all i from 1 to n. Further, using the General Fuzzy Isodata algorithm, a fuzzy partition [1] $P = \{A_1, A_2\}$ of C is detected. According to the definition of a fuzzy partition, we have:

$$C(i) = A_1(i) + A_2(i) \ , \ i=\overline{1,n}$$

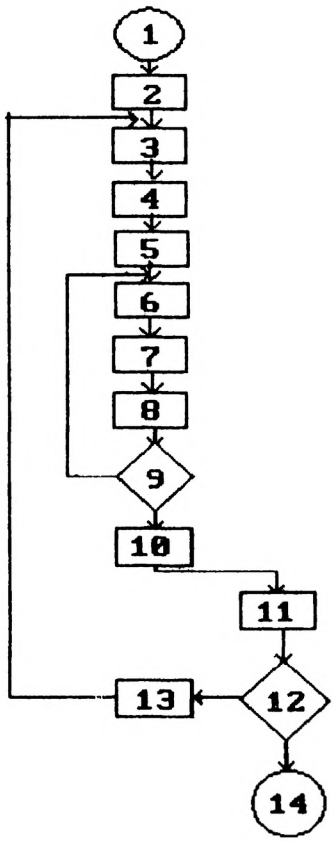
For the classification accuracy, the following correction rule is used:

$$\text{if } A_j(i) < 0.1 \text{ then } A_{2-j+1}(i)=C(i) \text{ and } A_j(i)=0.0 \ , \ i=\overline{1,n}, \ j=1,2$$

In determining the partition P, we use n feature vectors, representing the mean values of the features for each of the n classes. However, it is possible that not all the p features are needed to split the class C into A_1 and A_2. Using the set of test samples, we shall find the "best" up to K features in separating the two groups of classes. First, the best single feature is selected an this feature is used to perform classification based on the decision rule [1]. The result of the classification is computed and represents the number of test samples well classified. The "best 2" up to the "best K" feature subsets are

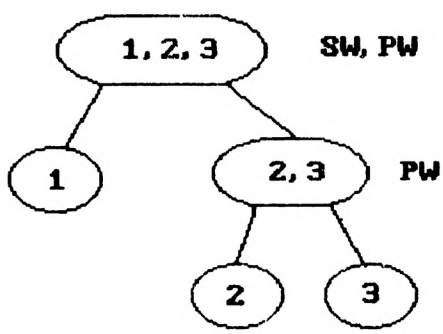
obtained. The feature subset which give the best classification result of the K "best" feature subsets is chosen as the feature subset for the node considered. When an unknown sample to be classified reaches this node and we use the decision rule to go further, only those features from the feature vector of the unknown sample which correspond to the feature subset associated with the current node will be considered in order to compute the distances to the prototypes of the descendants.

The flowchart which describes the binary tree classifier design process is given below:



1. Start
2. Find the mean values of the features for each of the n classes
3. Obtain separable clusters using General Fuzzy Isodata algorithm
4. If needed, use the correction rule
5. $l = 1$
6. Find the "best l" features
7. Perform classification using these l features
8. $l = l + 1$
9. Is $l > K$?
10. Find the best classification result from the result corresponding to each of the K "best" feature subsets
11. Use the best result obtained to build up the decision tree
12. No new nonterminal node?
13. Get a new nonterminal node
14. Stop

Results. The method described above has been used to design a binary tree classifier for the classification of 147 samples of Iris spread over 3 classes [2]: Iris Setosa, Iris Virginica and Iris Versicolor. There are 4 characteristics taken into consideration: petal width (*PW*), petal length (*PL*), sepal width (*SW*), sepal length (*SL*). Considering for each of the 3 classes the mean values of the 4 characteristics listed below and the set of test samples as consisting of the first 20 samples from each class listed in Anex *A*, the following tree classifier is obtained (*K*=2):



Although all the lass labels (1,2,3) appear in each node, only those which have the membership degree to the node *i* (*i*=1,...,5) greater than have been represented for the node *i* in the figure above. Beside each nonterminal node is the set of features used.

	Setosa	Versicolor	Virginica
PW	0.2	1.4	2.5
PL	1.4	4.7	6.0
SW	3.5	3.2	3.3
SL	5.1	7.0	6.3

mean values of the 4 characteristics

The classification results are as follows:

	samples no.	well classified	percent
Setosa	49	49	100%
Virginica	49	26	51.02%
Versicolor	49	49	100%
Total	147	127	85.03%

REFERENCES

1. D.Dumitrescu, *Teoria clasificării*, Universitatea Cluj-Napoca, Facultatea de Matematica, 1991.
2. R.A.Fischer, *The Use of Multiple Measurements in taxonomic problems*, Ann. Eugenics, 7, 179-188, 1936.
3. K.S.Fu and J.K.Mui, *Automated Classification of Nucleated Blood Cells Using a Binary Tree Classifier*, IEEE Transactions on Pattern Analysis and Machine Intelligence", vol. pami-2, no.5, September 1980.

Anexa A

Iris Setosa				Iris Versicolor				Iris Virginica			
SL	SW	PL	PW	SL	SW	PL	PW	SL	SW	PL	PW
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5
5.4	3.7	1.5	0.2	5.0	2.0	3.5	1.0	6.5	3.2	5.1	2.0
4.8	3.4	1.6	0.2	5.9	3.0	4.2	1.5	6.4	2.7	5.3	1.9
4.8	3.0	1.4	0.1	6.0	2.2	4.0	1.0	6.8	3.0	5.5	2.1
4.3	3.0	1.1	0.1	6.1	2.9	4.7	1.4	5.7	2.5	5.0	2.0
5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3
5.1	3.8	1.5	0.3	5.6	2.5	3.9	1.1	6.0	2.2	5.0	1.5
5.4	3.4	1.7	0.2	5.9	3.2	4.8	1.8	6.9	3.2	5.7	2.3
5.1	3.7	1.5	0.4	6.1	2.8	4.0	1.3	5.6	2.8	4.9	2.0
4.6	3.6	1.0	0.2	6.3	2.5	4.9	1.5	7.7	2.8	6.7	2.0
5.1	3.3	1.7	0.5	6.1	2.3	4.7	1.2	6.3	2.7	4.9	1.8
4.8	3.4	1.9	0.2	6.4	2.9	4.3	1.3	6.7	3.3	5.7	2.1
5.0	3.0	1.6	0.2	6.6	3.0	4.4	1.4	7.2	3.2	6.0	1.8
5.0	3.4	1.6	0.4	6.8	2.8	4.8	1.4	6.2	2.8	4.8	1.8
5.2	3.5	1.5	0.2	6.7	3.0	5.0	1.7	6.1	3.0	4.9	1.8
5.2	3.4	1.4	0.2	6.0	2.9	4.5	1.5	6.4	2.8	5.6	2.1
4.7	3.2	1.6	0.2	5.7	2.6	3.5	1.0	7.2	3.0	5.8	1.6
4.8	3.1	1.6	0.2	5.5	2.4	3.8	1.1	7.4	2.8	6.1	1.9
5.4	3.4	1.5	0.4	5.5	2.4	3.7	1.0	7.9	3.8	6.4	2.0
5.2	4.1	1.5	0.1	5.8	2.7	3.9	1.2	6.4	2.8	5.6	2.2
5.5	4.2	1.4	0.2	6.0	2.7	5.1	1.6	6.3	2.8	5.1	1.5
4.9	3.1	1.5	0.2	5.4	3.0	4.5	1.5	6.1	2.6	5.6	1.4
5.0	3.2	1.2	0.2	6.0	3.4	4.5	1.6	7.7	3.0	6.1	2.3
5.5	3.5	1.3	0.2	6.7	3.1	4.7	1.5	6.3	3.4	5.6	2.4
4.9	3.6	1.4	0.1	6.3	2.3	4.4	1.3	6.4	3.1	5.5	1.8
4.4	3.0	1.3	0.2	5.6	3.0	4.1	1.3	6.0	3.0	4.8	1.8
5.1	3.4	1.5	0.2	5.5	2.5	4.0	1.3	6.9	3.1	5.4	2.1
5.0	3.5	1.3	0.3	5.5	2.6	4.4	1.2	6.7	3.1	5.6	2.4
4.5	2.3	1.3	0.3	6.1	3.0	4.6	1.4	6.9	3.1	5.1	2.3
4.4	3.2	1.3	0.2	5.8	2.6	4.0	1.2	5.8	2.7	5.1	1.9
5.0	3.5	1.6	0.6	5.0	2.3	3.3	1.0	6.8	3.2	5.9	2.3
5.1	3.8	1.9	0.4	5.6	2.7	4.2	1.3	6.7	3.3	5.7	2.5
4.8	3.0	1.4	0.3	5.7	3.0	4.2	1.2	6.7	3.0	5.2	2.3
5.1	3.8	1.6	0.2	5.7	2.9	4.2	1.3	6.3	2.3	5.0	1.9
4.6	3.2	1.4	0.2	6.2	2.9	4.3	1.3	6.5	3.0	5.2	2.0
5.3	3.7	1.5	0.2	5.1	2.5	3.0	1.1	6.2	3.4	5.4	2.3

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

ON INDEPENDENT SETS OF GRAPHS

V. CIOBAN, V. PREJMEREAN, S. MOTOGNA*

Received: March 3, 1991
AMC Subject Classification: 05C90,68R10

Rezumat. - Asupra seturilor independente de grafe. Lucrarea trece în revistă unii algoritmi de determinare a mulţimilor independente (mulţimi interior stabile) referitoare la un graf. În prima parte se prezintă cîţiva algoritmi care au la bază expresii şi/sau ecuaţii booleene precum şi un algoritm recursiv şi anume algoritmul dat de Taulbee şi Bednarek. În final autorii dau un algoritm recursiv inspirat din acest ultim algoritm.

1. Definition, properties. Let $G = (V,T)$ be an undirected graph where:

- V is the set of vertices and $|V| = n$;
- $T : V \to V$ is the aplication which defines the graph.

DEFINITION : Let $S \subset V$. S is an independent set (IS) iff $\forall\, v \in S$, $T_v \cap S = \emptyset$.

(where we denote $T(v)$ by T_v).

In other words the vertices of S don't have any edges between each other.

Observations:

1^0 We may define $G = (V,E)$ where E is the set of edges, $E \subset V \times V$, an edge is $[x,y]$, $x,y \in V$ and $[x,x] \notin E$.

2^0 Let S be an IS. S is called maximal if S is maximal by sets inclusion.

3^0 We denote by $\mathbf{S}$ the set of all maximal IS of G.

We remember that:

a) $\alpha(G)$ is the number of internal stability:

* *University of Cluj-Napoca, Department of Computer Science, 3400 Cluj-Napoca, Romania*

$$\alpha(G)=\max_{S\in\mathcal{S}}|S|$$

b) $\gamma(G)$ is the chromatic number of G, $\gamma(G)$ is the smallest number of IS, disjoints, which cover G.

4⁰ Moon and Moser have proved that:

$$\alpha(G)\leq\begin{cases}3^{\frac{n}{3}} & ,\text{if } n=3k\\ 4*3^{\frac{n-1}{3}-1} & ,\text{if } n=3k+1\\ 2*3^{\frac{n-2}{3}} & ,\text{if } n=3k+2\end{cases}$$

2. **Algorithms for determining** *IS*. In many problems it is important to find the *IS* family.

There are some algebraic or combinatorial algorithms to find *IS*.

2.1. Maghout and Weissman'S algorithm based on boolean expression.

2.2. Malgrange algorithm's which finds every squared matrix containing only 0 (zero) of the adjacent matrix where:

$$A=(a_{i,j})\ ;\ i=\overline{1,n}\ ;\ j=\overline{1,n}\ \text{ with}$$

$$a_{ij}=\begin{cases}1 & ,\text{if } [v_i,v_j]\in E\\ 0 & ,\text{otherwise}\end{cases}$$

2.3. The Rudeanu's method, using the boolean equations which caracterize the *IS* family.

Let $G=(V,E)$, where $V=\{v_1,\ldots,v_n\}$.

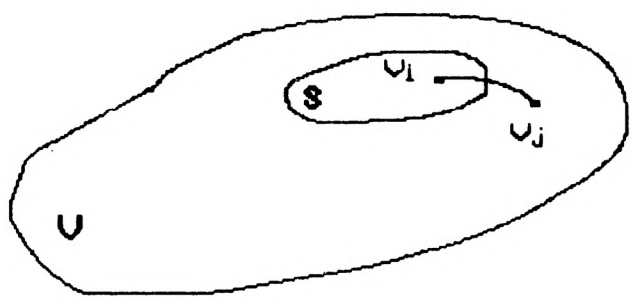
If $S\subset V$ is an *IS* then we associate to each $v_i\in V$ an boolean variable b_i define by :

$$b_i = \begin{cases} 1 & ,\text{if } v_i \in S \\ 0 & ,\text{if } v_i \notin S \end{cases}$$

We have the following result:

If $a_{ij} = 1$ then it results $a_{ij}.b_i.b_j = 0$ (1)

(because $v_i \in S$ and $v_j \notin S$, see the diagram).



So from (1) it results that $\bigvee_{a_{ij}=1} b_i \cdot b_j = 0$ iff $\bigwedge_{a_{ij}=1} \overline{b_i \cdot b_j} = 1$

iff $\bigwedge_{a_{ij}=1} (\overline{b_i} \vee \overline{b_j}) = 1$ iff $\bigvee \overline{b_{k_1}} \cdot \overline{b_{k_2}} \ldots \overline{b_{k_p}} = 1$

So, for each factor $\overline{b_{k_1}} \cdot \overline{b_{k_2}} \ldots \overline{b_{k_p}} = 1$ we have an *IS* :

$$S = \{x_{k_{p+1}}, x_{k_{p+2}}, \ldots, x_{k_n}\}.$$

2.4. Bednarek and Taulbee's recursiv algorithm

Let $G = (V,E)$ be an undirected graph:

- $\forall\ k = 1,\ldots,n$ we denoted by $V_k = \{v_1,\ldots,v_k\}$;
- for each subgraph with $V_k = \{v_1,\ldots,v_k\}$; we denoted by L_k the maximal *IS* family;
- we also denote $Y_k = \{y \in V_k \ / \ [x_k,y] \notin E\}$.

The steps of the algorithm are:

S1. Let $Y_1 = \{v_1\}$, $L_1 = \{v_1\}$, k=1.

S2. One finds the next family: $I_k=\{S/S=M\cap Y_{k+1},\ M\in L_k\}$.

S3. One finds $I_k' = \{I \ / \ I \subset I_k,\ I$ maximal with respect to sets inclusion$\}$.

S4. One finds L_{k+1}^* family, for each $M \in L_k$:

a) if $M \subset Y_{k+1} \rightarrow M \cup \{v_{k+1}\} \in L_{k+1}^*$

b) if $M \not\subset Y_{k+1} \rightarrow M \in L_{k+1}^*$ and $\{v_{k+1}\} \cup (M \cap Y_{k+1}) \in L_{k+1}^*$ iff $M \cap Y_{k+1} \in I_k'$

The L_{k+1}^* family contains only these sets of S4.

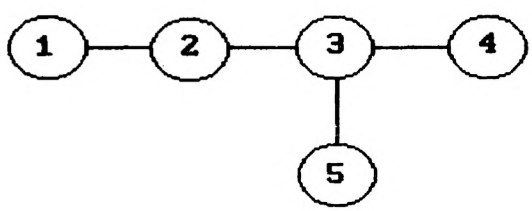
S5. One finds the maximal family L_{k+1} from L_{k+1}^* with respect to sets inclusion.

S6. Repeat S2,S3,S4,S5 for k=2,...,n-1. Finally we have L_n which contains the maximal *IS* of G.

Example :

Let $G = (V,E)$ be an undirected graph with

$V = \{1,2,3,4,5\}$, $E = \{[1,2],[2,3],[3,4],[3,5]\}$



In the next table we have:

k	Y_{k+1}	I_k	I_k'	L_{k+1}^*	L_{k+1}
1	{2}	ø	ø	{1},{2}	{1},{2}

2	{1,3}	ø,{1}	{1}	{1,3},{2}, {3}	{2}, {1,3}
3	{1,2,4}	ø,{1},{2}	{1},{2}	{1,3},{1,4}, {2,4},{4}	{1,3}, {1,4}, {2,4}
4	{1,2,4,5}	ø,{1},{1,4}	{1,4}	{1,3}, {1,4,5}, {2,4,5},{5}	{1,3}, {1,4,5} {2,4,5}

→ L = {{1,3}, {1,4,5}, {2,4,5}}

2.5. In what follows we suggest the next algorithm:

The notations used:

Let $G = \{V,E\}$ be an undirected graphs and:

$V_k = \{v_1,\ldots,v_k\}$, $|V| = n, 1 \le k \le n$;

L_k = the sets family of *IS* associated with V_k, $1 \le k \le n$.

The steps of the algorithm are:

S1. $L_1 = \{v_1\}$, k=2.

S2. One finds L_k :

a) if $M \in L_{k-1}$ → $M \in L_k$.

b) if $M \in L_{k-1}$ and $\forall\, y \in M$ with $[y,x_k] \notin E$ → $M \cup \{x_k\} \in L_k$.

c) $\{v_k\} \in L_k$.

Repeat S2 for k=2,3,...,n.

S3. Reducing L_n with respect to sets inclusion:

$\forall\, M,N \in L_n$ and $M \subset N$ → $L_n = L_n \setminus M$.

For the previous graph we have:

L_1 : {1}.

L_2 : {1}, {2}.

L_3 : {1}, {2}, {1,3}, {3}.

L_4 : {1}, {2}, {1,3}, {3}, {1,4}, {2,4}, {4}.

L_5 : {1}, {2}, {1,3}, {3}, {1,4}, {2,4}, {4}, {1,5}, {2,5}
{1,4,5}, {2,4,5}, {4,5}, {5}.

Applying S3 we obtain:

L : { {1,3}, {1,4,5}, {2,4,5} }.

The algorithm is very simple and it works only with a single sets family.

R E F E R E N C E S

1. Tomescu,I., *Introduction in combinatorics*, Ed. Tehnică, 1972, pp. 191-195.
2. Moon,J.W., Moser,L., *On cliques in graphs*, Israel Journal of Mathematics, vol. 3 no.1, 1965, pp. 23-28.
3. Maghout,K., *Sur la determination de nombre de stabilite et du nombre chromatique d'une graphe*. Comptes Rendus de l'Academie des Sciences, Paris, 248, 1959, pp. 3522-3523.
4. Weissman,J. *Boolean Algebra*, map coloring and interconection Am. Math. Monthly, no. 69, 1962, pp. 608-613.
5. Malgrange,Y. *Recherche des sous-matrices premieres d'une matrice a coefficients binaires*. Applications de certains problemes des graphs. Deuxieme Congres de l'AFCALTI, oct., 1961, Gauthier-Villars, Paris, 1962, pp. 231-242.

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

ON SOME PARALLEL METHODS IN LINEAR ALGEBRA

GH. COMAN*

Received : March 2, 1991
AMS subject classification: 65FXX, 68Q10

REZUMAT. - Asupra unor metode paralele în algebra liniară. Sînt studiate din punct de vedere al complexităţii mai multe metode numerice de inversare a matricelor şi de rezolvare a sistemelor algebrice liniare.

The parallel computation had became an actual problem in many application fields.

Of course, not each mathematical method can be efficientely projected in a parallel version.

To characterize the depth of the parallelism of a given method there exists specifically criterions. Such criterions are the speed and the efficiency. The goal of this paper is to discus some methods in linear algebra from the parallelism point of view.

Let X be a linear space, X_0 a subset of X, $(Y,\|.\|)$ a normed linear space and $S,S: X_0 \to Y$, a given operator. The problem: for given $\varepsilon>0$ and $x \in X_0$ find an $y \in Y$ such that $\|S(x) - y\| \leq \varepsilon$ is called a S - problem, x is the problem element, S is the solution operator and $s = S(x)$ is the solution element. $\tilde{s} \in Y$ for which $\|\tilde{s} - s\| \leq \varepsilon$ is called an ε - approximation of the solution s.

In order to solve a S - problem there are necessary some informations on the problem element x. So, let Z be a set (the set of informations). The operator $\mathcal{I}: X \to Z$ is called the informational operator and $\mathcal{I}(x)$, $x \in X_0$, is the information on

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

x. To compute a solution of a S - problem for a given information $\Im(x)$ we need an algorithm, which is defined as an application α: $\Im(X_0) \to Y$. So, for a given $x \in X_0$, $\alpha(\Im(x))$ is the approximation of the solution $S(x)$ given by the algorithm α with the information $\Im(x)$ as the input data. If $\alpha\ (\Im(x))$ is an ε - approximation of $S(x)$ then $\Im$ and α are called ε - admissible. So, to solve a S - problem means to find an ε - admissible informational operator and an ε - admissible algorithm for it.

DEFINITION 1. A couple $(\Im,\alpha)$ with $\Im: X \to Z$ and $\alpha : \Im\ (X_0) \to Y$ is called a method associated to a S - problem.

If $\Im$ and α are ε - admissible then the corresponding method is called also ε - admissible.

Next, one denotes by $M(S)$ the set of all admissible methods for the problem S. A method $\mu \in M(S)$, $\mu = (\Im,\alpha)$, is called a serial method if all the computations are described as a single instructions stream (α is a serial algorithm). If the computations are described as a multiple instructions streams then μ is called a parallel method (α is a parallel algorithm).

To distinguish the two kind of methods one denotes by $M_s(S)$ the set of all serial methods for the problem S and by $M_p(S)$ the set of all parallel methods for S.

For a method $\mu \in M(S)$ one denotes by $CP(\mu;x)$, $x \in X_0$, its computational complexity for the element x or the local complexity, while

$$CP(\mu) = \sup_{x \in X_0} CP(\mu;x)$$

is the complexity of the method μ for the problem S(global

complexity) [3].

DEFINITION 2. The method $\overline{\mu} \in M_s$ (S) for which

$$CP(\overline{\mu}) = \inf_{\mu \in M(S)} CP(\mu)$$

is called the optimal method with regard to the complexity.

Now, let μ be a serial method, $\mu \in M_s$ (S).

Generally speaking, by a parallel method $\mu_p \in M_p$ (S), associated to μ we understand a method in which all the operations, independent to each others, are performed in parallel (in the same time). So, we can image the serial method divided in many parts (segments - streams of instructions) independently or partial independently from the computation point of view, say $\mu_1, \ldots, \mu_r$. Then

$$CP(\mu_p) = \max_{1 \le i \le r} CP(\mu_i)$$

is the complexity of the corresponding parallel method μ_p.

DEFINITION 3. Let S be a given problem, $\mu_p \in M_p(S)$ a parallel method and $\overline{\mu_s} \in M_s$ (S) the optimal serial method with regard to the complexity.

Then

$$S(\mu_p) = \frac{CP(\overline{\mu}_s)}{CP(\mu_p)}$$

is called the speed of the parallel method μ_p.

Remark 1. The speed is also denoted by $S(\mu_p; r)$, where r is the number of the instructions streams of the method μ_p.

Obviously, $S(\mu_p; r) \le r$.

Remark 2. A more practical value to judge the parallel version μ_p of a serial method μ_s is

$$s(\mu_p;r) = \frac{CP(\mu_s)}{CP(\mu_p)}.$$

We also have $s(\mu_p;r) \geq S(\mu_p;r)$.

DEFINITION 4. The value

$$E(\mu_p) = \frac{S(\mu_p;r)}{r}$$

is called the efficiency of the parallel method μ_p.

As $0 \leq S(\mu_p;r) \leq r$ it follows that $0 \leq E(\mu_p) \leq 1$.

Next, we consider first some examples.

E.1. Let $\mathcal{E}$ be the following expression :

$$\mathcal{E} = t_1 \ \rho \ t_2 \ \rho \ \ldots \ \rho \ t_n$$

where ρ is an associative operation.

The serial computational complexity of $\mathcal{E}$ is

$$CP(\mathcal{E}) = (n-1)\, CP(\rho),$$

where $CP(\rho)$, is the complexity of the operation ρ.

A parallel version $\mathcal{E}_p$ of the expression $\mathcal{E}$ is obtained as follows: in the first step we compute, say $t_i^1 := t_{2i-1} \rho \ t_{2i}$, for all possible i. To do it more clear, let $m \in \mathbb{N}$ be such that $2^{m-1} < n \leq 2^m$. If $n < 2^m$ then we supplement the expression $\mathcal{E}$ by

$$t_{n+1} = \ldots = t_{2m} = 0, \text{ i.e.}$$

$$\mathcal{E} = t_1 \ \rho \ t_2 \ \rho \ \ldots \ \rho t_n \ \rho \ t_{n+1} \ \rho \ \ldots \ \rho \ t_{2^m}$$

so,

$$t_i^1 := t_{2i-1} \ \rho \ t_{2i}, \quad i = 1,\ldots,2^{m-1}.$$

In the second step we have

$$t_i^2 := t_{2i-1}^1 \ \rho \ t_{2i}^1, \qquad i = 1,\ldots,2^{m-2}$$

and so on

$$t_i^k := t_{2i-1}^{k-1} \rho\ t_{2i}^{k-1}\ , \qquad i = 1,\dots,2^{m-k}$$

for $k = 3,\dots,m$. Finally, we have $\mathscr{E} = t_1^m$. Hence, the necessary steps to compute $\mathscr{E}$ is m. Taking into account that $2^{m-1} < n \leq 2^m$, one obtains $m = [\log_2 n]$, where $[x]$, $x \in \mathbb{R}$ is the integer with the property $x \leq [x] < x + 1$.

It follows that

$$CP\ (\mathscr{E}_p) = [\log_2 n]\ CP\ (\rho)\ .$$

So, we have

$$s(\mathscr{E}_p; [n/2]\ ) = \frac{n - 1}{[log_2 n]}$$

and

$$E(\mathscr{E}_p) = \frac{n - 1}{[n/2]\ [log_2 n]} \approx \frac{2}{[log_2 n]}$$

where $[x]$ is the integer part of x.

Remark 3. If we consider the binary tree associated to the expression $\mathscr{E}$ then the complexity of the parallel computation of $\mathscr{E}$ is the depth of the tree [5].

E.2. Let be $X = M_n(\mathbb{R})$, $X_0 = X$, $Y = \mathbb{R}$ and $S : X \to Y$, $A \to \det A$. Hence, S is the problem to compute the determinant $\det A$ of the matrix A. The method used consists in the transformation of the determinant

$$\det A = \begin{vmatrix} a_{11} & a_{12} \cdots & a_{1n} \\ a_{21} & a_{22} \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} \cdots & a_{nn} \end{vmatrix} .$$

in the form

$$\det A := a_{11}^1 * \ldots * a_{nn}^n * \begin{vmatrix} 1 & a_{11}^2 & \ldots & a_{1n}^2 \\ 0 & 1 & \ldots & a_{2n}^3 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \ldots & 1 \end{vmatrix}$$

using the operations :

$$a_{ij}^1 := a_{ij} \ , \ i,j = 1,\ldots,n$$

$$a_{ip}^{p+1} := \frac{a_{ip}^p}{a_{pp}^p} \ , \ i = p+1,\ldots,n$$

$$a_{ij}^{p+1} := a_{ij}^p - a_{pj}^p * a_{ip}^{p+1} \ , \quad i,j = p+1,\ldots,n$$

for $p = 1,\ldots,n-1$.

So, we have

$$\det A = a_{11}^1 * a_{22}^2 * \ldots * a_{nn}^n \ .$$

Remark 4. Next we suppose that $CP(+) = 1$ (a unit time) and $CP(*) = CP(/) = 3$.

If one denotes by μ_s the serial method to compute det A, one obtains

$$CP(\mu_s) = \frac{1}{6}(n-1)(8n^2+5n+18) \ .$$

A parallel version of the considered method using n parallel instructions streams (n processors) is :

```
begin
    det A: = 1;
    for p: = 1 step 1 until n -1 do
    begin
        (det A: = det A * a_pp^p; (p + 1 ≤ j ≤ n)  a_pj^{p+1} := a_pj^p / a_pp^p )   ;
        ((p+1 ≤ j ≤ n) for i:=p+1 step 1 until n do
```

$$a_{ij}^{p+1} := a_{ij}^{p} - a_{ip}^{p} * a_{pj}^{p+1})$$

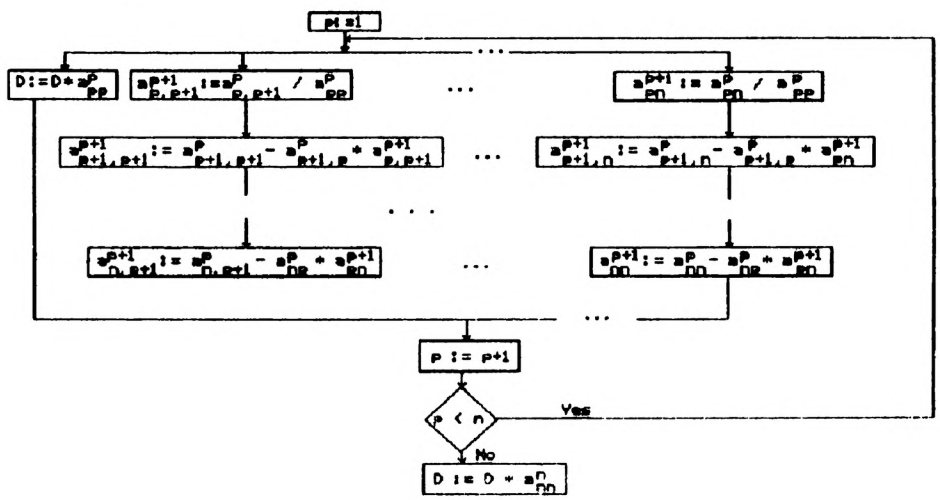
end

det A := det A * a_{nn}^{n}

end

Remark 5. $(I, (1 \le k \le m)\ I_k)$ means that the instructions I, I_1, ..., I_m are performed in parallel.

For a better illustration of the parallel method, say μ_p, we give the next diagram (D:= det A) :



The complexity of the parallel method μ_p, as it can be easy seen, is:

$$CP(\mu_p; n) = n(2n - 1).$$

So,

$$s(\mu_p; n) = \frac{(n-1)(8n^2 + 5n + 18)}{/6n(2n - 1)} \approx \frac{2}{3} n + \frac{1}{12}$$

and

$$E(\mu_p) \approx \frac{2}{3} .$$

E.3. For $X = M_n(\mathbb{R})$, $X_0 = \{A \mid \det A \neq 0, A \in X\}$, $Y = M_n(\mathbb{R})$ and $S(A) = A^{-1}$, S is the problem to compute the inverse of a matrix.

We use the method based on the succesive transformations of the matrix $[A \mid I_n]$ in the matrix $[I_n \mid A]$, where I_n is the unit matrix of order n. The transformations are : first one denotes the elements of the matrix $[A \mid I_n]$ by t_{ij}^1, $i=1,\ldots,n$; $j=1,\ldots,2n$. Now,

$$t_{pj}^{p+1} := \frac{t_{pj}^p}{t_{pp}^p} , \quad j=p+1,\ldots,2n$$

$$t_{ij}^{p+1} := t_{ij}^p - t_{ip}^p * t_{pj}^{p+1} , \quad i=1,\ldots,n, \ i\neq p ; \ j=p+1,\ldots,2n$$

$$t_{nj}^n := \frac{t_{nj}^n}{t_{nn}^n} , \quad j=n+1,\ldots,2n ,$$

for all $p = 1,\ldots,n-1$.

So,

$$A^{-1} = (t_{ij}^n) \ i = \overline{1,n} ; \quad j = \overline{n+1, 2n}$$

If μ_s is the corresponding serial method then

$$CP(\mu_s) = \frac{3}{2} n (4n^2 - 5n + 3) .$$

A parallel method, μ_p, can be projected as follows :

```
begin
      t²₁₂ := t¹₁₂ / t¹₁₁ ;
      for p:=1 step 1 until n - 1 do
      begin for j:=p+1 step 1 until 2n do
```

$$\left(t_{p,j+1}^{p+1} := \frac{t_{p,j+1}^{p}}{t_{pp}^{p}} \; ; \; (1 \le i \le n,\ i \ne p) \;\; t_{ij}^{p+1} := t_{ij}^{p} - t_{ip}^{p} * t_{pj}^{p+1} \right)$$

end;

$$\left((n+1 \le j \le 2n) \quad t_{nj}^{n} := \frac{t_{nj}^{n}}{t_{nn}^{n}} \right)$$

end

We have

$$CP(\mu_p; n) = 6(n^2 - n + 1)$$

and

$$s(\mu_p; n) \approx n - \frac{1}{4}$$

respectively

$$E(\mu_p) \approx 1.$$

Remark 6. From these three examples we can see that the matrix inversion permites a very good parallelism ($E(\mu_p) \approx 1$), while for the determinant computation $E(\mu_p) \approx 2/3$ and in the first example

$$E(\mathcal{E}_p) \approx 2/[\log_2 n] \ .$$

Linear algebraic systems.

If $X = \{[A|b] \mid A \in M_n(\mathbb{R}),\ b \in M_{n,1}(\mathbb{R})\}$, $X_0 = \{[A|b] \in X \mid \det A \ne 0\}$ $S([A|b]) = A^{-1}b$ then S is the problem to solve the system $As = b$.

Next, there are discused serial and parallel versions for some well known numerical methods for the solution of linear algebraic systems.

I. **Cramer's method.** Taking into account that the solution is given by $s_i = D_i/D, i = 1, \ldots, n$, where $D = \det A$ and D_i is the determinant obtained by D changing the i-th column vector by b.

So, we have to compute $n + 1$ determinants of order n, with the complexity $CP(\mu_s)$ from the example E_2, and n divisions. It follows that the serial complexity of Cramer's method μ_s^c is $CP(\mu_s^c) = (n+1)CP(\mu_s) + nCP(/)$, i.e.

$$CP(\mu_s^c) = \frac{1}{6}(8n^4 + 5n^3 + lon^2 + 13n - 18). \qquad (1)$$

A natural parallel method here is to compute in parallel the $(n+1)$ determinants and than to perform the n divisions. So,

$$CP(\mu_p^c) = \frac{1}{6}(8n^3 - 3n^2 + 13n) \qquad (2)$$

where μ_p^c is the mentioned parallel method.

Hence, one obtains

$$s(\mu_p^c;\ n+1) = (n+1) - \frac{18}{8n^3-3n^2+13n} \approx n+1$$

and

$$E(\mu_p^c) \approx 1. \qquad (3)$$

As a conclusion we can remark the very good parallelism of Cramer's method $(E(\mu_p^c) \approx 1)$.

II. **Gaussian elimination method.** As, it is well known first the given matrix $[A|b] \in X_0$ is transformed in the matrix $[T_n \mid \bar{b}]$, where T_n is an upper triungular matrix ($T_n = (a_{ij}^i)$ $i=1,\ldots,n;\ j=i+1,\ldots,n;\ a_{ii}^i=1$) using the relations

$$a_{pj}^{p} := a_{pj}^{p} \;/\; a_{pp}^{p},\ j=p+1,\ldots,n;\quad b_{p}^{p} \;/\; a_{pp}^{p}$$

$$a_{ij}^{p+1} := a_{ij}^{p} - a_{ip}^{p} * a_{pj}^{p},\quad i,j=p+1,\ldots,n$$

$$b_{i}^{p+1} := b_{i}^{p} - a_{ip}^{p} * b_{p}^{p},\quad i=p+1,\ldots,n$$

for $p = 1,\ldots,n-1$, and $b_n^n := \dfrac{b_n^n}{a_{nn}^n}$, where for the begining $a_{ij}^1 := a_{ij}$, $b_i^1 := b_i$, $i,j=1,\ldots,n$.

The complexity of this computation is $n(n^2-1)/3*[CP(+)+CP(*)] + n(n+1)/2 * CP(/)$. Now the triangular system $T_n s = b$ is solved by back substitution method:

$$s_n := b_n^n$$

$$s_i := b_i^i - \sum_{j=i+1}^{n} a_{ij}^i * x_j,\quad i=n-1,\ldots,1,$$

with the computational complexity $n(n-1)/2 * [CP(+) + CP(*)]$. It follows that

$$CP(\mu_g^G) = \frac{1}{6}\,(8n^3 + 21n^2 - 11n). \qquad (4)$$

A parallel version μ_p^G of the Gauss method is :

```
begin
for p:=1 step 1 until n-1 do
    begin
```

$$\left((p+1\le j\le n+1)\quad a_{pj}^{p} := \frac{a_{pj}^{p}}{a_{pp}^{p}}\right);$$

```
    for i:=p+1 step 1 until n do
    begin ((p+1≤j≤n+1)  a_ij^(p+1) := a_ij^p - a_ip^p * a_pj^p) ;   b_i^(p+1) := b_i^p - a_ip^p * b_p^p  end
    end;
```

$a^n_{n,n+1} := \dfrac{a^n_{n,n+1}}{a^n_{nn}}$

for k:=1 step 1 until n - 1 do

$\left((k \le i \le n-1) \quad a^n_{n-i,n+1} := a^n_{n-i,n+1} - a_{n-i,n-k+1} * a^n_{n-k+1,n+1}\right)$

end

where $a^p_{p,n+1} = b^p_p$.

So, $s_i := a^n_{i,n+1}$, $i=1,\ldots,n$.

It follows that

$$CP(\mu^G_p; n) = 2n^2 + 5n - 11 \tag{5}$$

and

$$s(\mu^G_p; n) \approx \frac{2}{3}\, n + \frac{1}{2}$$

respectively

$$E(\mu^G_p) \approx \frac{2}{3} \,. \tag{6}$$

III. **Total elimination method.** The matrix $[A|b] \in X_0$ is transformed in the matrix $[\, I_n \mid b^n]$.

First,

$a^1_{ij} := a_{ij}$, $a^1_{i,n+1} = b_i$, $i,j=1,\ldots,n$.

Now, one applies the succesive transformations

$a^{p+1}_{pj} := \dfrac{a^p_{pj}}{a^p_{pp}}$, $j=p+1,\ldots,n+1$;

$a^{p+1}_{ij} := a^p_{ij} - a^p_{ip} * a^{p+1}_{pj}$; $i=1,\ldots,n$; $i \neq p$; $j=p+1,\ldots,n+1$

for all $p = 1,\ldots,n$.

So, the solution is $s_i := a^{n+1}_{i,n+1}$, $i=1,\ldots,n$.

The computational complexity of this method in the serial version (μ^T_s) is

$$CP(\mu_s^T) = \frac{1}{2}(4n^3 + 3n^2 - n). \tag{7}$$

As a parallel version (μ_p^T) of the total elimination method is the following :

begin

$a_{12}^2 := \dfrac{a_{12}^1}{a_{11}^1}$;

for p:=1 step 1 until n - 1 do

begin

for j:=p+1 step 1 until n do

$$\left(a_{p,j+1}^{p+1} := \frac{a_{p,j+1}^{p}}{a_{pp}^{p}};\quad (1\le i\le n\quad i\neq p)\quad a_{ij}^{p+1} : a_{ij}^{p}-a_{ip}^{p}*a_{pj}^{p+1}\right)$$

$$\left(a_{p+1,p+2}^{p+2} := \frac{a_{p+1,p+2}^{p+1}}{a_{p+1,p+2}^{p+1}};\quad (1\le i\le n,\ i\neq p)\quad a_{i,n+1}^{p+1} := a_{i,n+1}^{p}-a_{ip}^{p}*a_{p,n+1}^{p+1}\right)$$

end

$$\left(a_{n,n+1}^{n+1} := \frac{a_{n,n+1}^{n}}{a_{nn}^{n}};\quad (1\le i\le n-1)\quad a_{i,n+1}^{n+1} := a_{i,n+1}^{n}-a_{in}^{m}*a_{n,n+1}^{n+1}\right)$$

end

We have

$$CP(\mu_p^T) = 2n^2 + 2n + 3. \tag{8}$$

So,

$$s(\mu_p^T ; n) \approx n - \frac{1}{4}$$

and

$$E(\mu_p^T) \approx 1. \tag{9}$$

IV. **Iterative methods.** One considers two iterative methods.

IV.1. **Jacobi iteration.** For a given $x^{(0)} = (x_1{}^{(0)},\ldots,x_n{}^{(0)})^T$, the sequence of the succesive approximation $x^{(m+1)}$ is given by

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}} a_{ij} x_j^{(m)}\right), \quad i = 1, \ldots, n.$$

If $CPI(\mu_s^T)$ is the computational complexity of one iteration then the serial complexity of the Jacobi method is

$$CP(\mu_s^T) = m_J(\varepsilon) \quad CPI(\mu_s^J),$$

where $m_J(\varepsilon)$ is the iterations number for which $x^{(m_J(\varepsilon))}$ is an ε-approximation of the solution. So, we have

$$CP(\mu_s^J) = (4n^2 - n)\, m_J(\varepsilon) \,. \tag{10}$$

A parallel version of the method μ_s^J is to compute, in parallel, each $x_i^{(m+1)}$, $i=1,\ldots,n$.

Hence,

$$CP(\mu_p^J; n) = (4n-1)\, m_J(\varepsilon) \,. \tag{11}$$

It follows that

$$S(\mu_p^J; n) = n$$

and

$$E(\mu_p^J) = 1$$

IV.2. **Gauss - Siedel iteration.** Starting with $x^{(0)}$, the iterations are given by

$$x_i^{(m+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)}\right), i=1, \ldots, n.$$

The serial complexity of the Gauss-Siedel method is

$$CP(\mu_s^{GS}) = (4n^2 - n)\, m_{GS}(\varepsilon) \,, \tag{13}$$

where $m_{GS}(\varepsilon)$ is the iterations number.

It is obviously that the parallelism of the Gauss-Siedel method is more less than of the Jacobi iteration. Certainly we solve for $x_2^{(m+1)}$ using already the "new" value $x_1^{(m+1)}$, for $x_3^{(m+1)}$ it is used the "new" values $x_1^{(m+1)}$, $x_2^{(m+1)}$ and so on. Hence, $x_2^{(m+1)}$ can be computed only when the computation of $x_1^{(m+1)}$ is finished and the computation of $x_3^{(m+1)}$ must wait for $x_1^{(m+1)}$ and $x_2^{(m+1)}$ and so on. It follows that a parallel version μ_p^{Gs} is to do the computation begining with the first line ($x_1^{(m+1)}$) than the second one ($x_2^{(m+1)}$) and so on. One obtains

$$CP(\mu_p^{GS}) = n([\log_2 n] + 6)\, m_{GS}(\varepsilon)$$

and

$$E(\mu_p^{GS}) = \frac{4(1 - 1/n)}{[\log_2 n] + 6} .$$

Conclusions. Taking into account the serial and parallel complexity of the above methods for linear algebraic systems it follows:

PROPOSITION 1. $CP(\mu_s^G) < CP(\mu_s^T) < CP(\mu_s^C)$, $\forall n > 2$.

The proof follows directly by (1), (4) and (7).

Remark 7. Of the Gauss - Siedel procedure may be viewed as an acceleration of Jacobi method, so we generally have $m_{GS}(\varepsilon) \leq m_J(\varepsilon)$ i.e.

$$CP(\mu_s^{GS}) < CP(\mu_s^J) .$$

Now, from (2) and (10), it follows :

PROPOSITION 2. *If* $m_{GS}(\varepsilon) \leq [n/3]$ *then*

$$CP(\mu_s^{GS}) < CP(\mu_s^G) .$$

Remark 8. For the systems with a lagre number of equation (such that $[(n/3) + 1]$ iterations are sufficient to get a good

approximation the Gauss - Siedel iteration is better than of the Gauss elimination method.

The following two propositions give some informations regarding with the parallel methods.

PROPOSITION 3. $CP(\mu_p^T) < CP(\mu_p^G) < CP(\mu_p^C) \quad \forall n > 2$.

The proof is based on the relations (2), (5) and (8).

Remark 9. For the parallel version μ_p^G and μ_p^T we have $CP(\mu_p^G) > CP(\mu_p^T)$ just if in the serial case the relation is $CP(\mu_s^G) < CP(\mu_s^T)$. So, generally a good serial method does not conduct to a good parallel version.

PROPOSITION 4. *If* $m_T(\varepsilon) < [n/2]$ *then* $CP(\mu_p^T) < CP(\mu_p^T)$.

Remark 10. In the parallel case it can be done just $[n/2]$ iterations without passing the complexity of the best parallel method μ_p^T

Finally, from (3), (6), (9) and (12) it follows that the best parallelism is possessed by the Jacobi iteration method ($E(\mu_p^T) = 1$). Also, a good parallelism has the total elimination method ($E(\mu_p^T) \approx 1 - \frac{1}{4n}$) and the Cramer's method ($E(\mu_p^C) \approx 1$). But the complexity of the Cramer method is, in both serial and parallel versions, a polynomial function on degree with a unity greater than the other ones. So, the Cramer's method is never recommended from the computational complexity point of view.

REFERENCES

1. Blum E.K., *Numerical Analysis and computation.* Theory and practice, Addison - Wesley Publishing Company, 1972.
2. Coman Gh., *On the parallel complexity of some numerical algorithms for solving linear systems.*"Babeş-Bolyai" University, Cluj-Napoca Research Seminars, Preprint Nr. 6, 7-16, 1987.

3. Coman Gh., Johnson D., *The complexity of algorithms*. "Babeş-Bolyai" University, Cluj-Napoca, 1987.
4. Fadeeva V.N., Fadeev D.K., *Parallel computation in linear algebra*. Kibernetica, 6, 28-40, 1977.
5. Heller D., *A survey of parallel algorithms in numerical linear algebra*. SIAM Rev. 20, 740-777, 1978.
6. Solodovnikov V.I., *Upper bounds on complexity of solving systems of linear equations*. Zap.naucin.seminars (LOMI) , 159-187, 1982.
7. Stone H.S., *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*. J.ACM, 20, 27-38, 1973.
8. Traub J.F., Wozniakowski H., *A General Theory of Optimal Algorithms*, Academy Press, 1980.

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

ON THE CONVERGENCE OF THE THREE-ORDER METHODS IN FRECHET SPACES

SEVER GROZE*

Received: July 3, 1991
AMS subject classification: 65Hxx

REZUMAT. - Asupra convergenţei metodelor de ordinul trei în spaţii Fréchet. În lucrare se demonstrează existenţa şi unicitatea existenţei ecuaţiei (1) precum şi convergenţa metodei iterative $(2)_{,1}$ renunţînd la uniform mărginirea operatorului $\Lambda = [x', x''; P]^{-1}$.

1. It is known that the rapidity of convergence for the sequence of approximates (x_n) of solution of the operatorial equation

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

given by an iterative method, can be improved if the first and the second order divided differences, which enter in the algorithm exprimation, are taken on special nodes.

In the case of operatorial equation

$$P(x) \equiv x - F(x) = \theta \tag{2}$$

using the metod

$$x_{n+1} = x_n - \Lambda_n (I - [x_n, u_n, v_n; P]\ \Lambda_n\ P(u_n)\ \bar{\Lambda}_n)^{-1}\ P(x_n) \tag{3}$$

where

$$\Lambda_n = [x_n, u_n; P]^{-1};\quad \bar{\Lambda}_n = [u_n, v_n; P]^{-1}$$

and

$$u_n = F(x_n);\ v_n = F(u_n) = (F(x_n))$$

this property is proved in the paper [1]. The following theorem

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

are proved:

THEOREM A. *If for* $x_0 \in X$, *there exist* μ_0, B, $M>1$ *and* N *so that the following conditions:*

1) $)| P(x_0) |(< \mu_0$;

2) *For any* $x', x'', x''', x^{IV} \in S(x_0, R)$, R *remaining to be defined, we have*

a. $\Lambda = [x', x''; P]^{-1}$ *exists and* $)| \Lambda |(< B$;

b. $)| [x', x''; F] |(< M$;

c. $)| [x', x'', x'''; P] |(< K$;

d. $)| [x', x'', x^{IV}; P] - [x', x'', x'''; P] |(< < N)| x^{IV} - x''' |($

3) $G_0 h_0 < 1$ *where* $h_0 := B^2 MK\mu_0 < 1/2$ *and*

$$G_0^2 := \frac{M(1+BK\mu_0)\,[1+BK\mu_0(1+M)]}{(1+h_0)^2(1-2h_0)}\left(1+\frac{N}{BK^2}\right)$$

hold, then the equation (2) *has the solution* $x^* \in S(x_0, R)$, *where*

$$R=(1+M)\,\mu_0+M^2Q \quad \text{and} \quad Q=\frac{B\mu_0}{1-h_0}\sum_{m=0}^{n}(G_0h_0)^{3^m-1}$$

solution which is the limit of the sequence genereted by (3), *the rapidity of convergence being given by*

$$)|x^*-x_m|(\ , \quad (G_0h_0)^{3^m-1}Q.$$

THEOREM B. *In the conditions of Theorem A, the solution of equation* (2) *is unique.*

In the following, we will change the condition 2a of Theorem A, removing the uniform bounded of the operator Λ.

2. Let us consider the equation

$$P(x) = x - F(x) = \theta \quad .$$

where $P: X \to X$ is a continuous operator considered with its generalized divided difference [2] up to the second order, inclusively, X is a Fréchet space with a quasinorm induced by a distance invariant to translation, i.e $)| x |(= d(x,\theta)$, $x,\theta \in X$ [3].

To solve the equation (2) we consider the algoritm (3).

Concerning the convergence of the method (3), we prove

THEOREM. *If for $x_0 \in X$ exists $\overline{\mu_0}$, $\overline{M} > 1$, $\overline{K}$ and $\overline{N}$ such that the following conditions:*

1^0 *For any $x', x'', x''', x^{IV} \subset S$, where $S = \{x \mid)|x - x_0|(< R\}$,*

$$R = (1+\overline{M})\mu_0 + \overline{M}^2 \overline{D} \; ; \quad \overline{D} = \frac{\overline{\mu_0}}{1-\overline{h_0}} \sum_{n=0}^{n} (\overline{G_0} \overline{h_0})^{3^n - 1}$$

we have:

a. $\Lambda = [x', x''; P]^{-1}$ *exists;*

b. $)| \Lambda[x', x''; F] |(< \overline{M}$;

c. $)| \Lambda[x', x'', x'''; P] |(< \overline{K}$;

d. $)| \Lambda ([x', x'', x^{IV}; P] - [x', x'', x'''; P]) |(< \overline{N})|x^{IV} - x'''|($

2^0 $)| \Lambda P(x_0) |(< \overline{\mu_0}$;

3^0 $\overline{G_0}\,\overline{h_0} < 1$ *where* $\overline{h_0} := \overline{M}^2 \overline{K}\,\overline{\mu_0} < 1/2$ *and*

$$\overline{G_0} = \frac{\overline{N}(1+\overline{K}\overline{\mu_0})[1+\overline{K}\overline{\mu_0}(1+\overline{M})]}{(1+h_0)^2(1-2h_0)}\left(1+\frac{\overline{N}}{K^2}\right)$$

hold, then the equation (2) has the unique $x^* \subset S(x_0, R)$, which is

the limit of the sequence genereted by (3), *the rapidity of convergence being given by*

$$)\,|\,x^*-x_n\,|\,(\;<\;(\overline{C}_0\,\overline{h}_0)^{3^n-1}\cdot\overline{Q}.$$

Proof. We consider the equation

$$\overline{P}(x)=\theta \tag{6}$$

where

$$\overline{P}(x)=\Lambda P(x)=\Lambda(x-F(x)),\quad \Lambda=[x',x'';P]^{-1}$$

equation which is equivalent to (2).

Indeed, if x^* is a solution of equation (2), i.e. $P(x^*)=\theta$, due to liniarity of Λ, it results

$$\Lambda P(x^*)=\overline{P}(x^*)=\theta. \tag{7}$$

Reciprocal, if x^* is a solution of equation (6), i.e.

$$\overline{P}(x^*)=\Lambda P(x^*)=\theta$$

from the existence of operator Λ, it results $\Lambda^{-1}=[x',x'';P]$ which, applied to the left of the equation (6), leads to $P(x^*)=\theta$.

For solving this equation, we have the iterative method

$$\overline{x}_{n+1}=\overline{x}_n-\overline{\Lambda}_n(I-[\overline{x}_n,\overline{u}_n,\overline{v}_n;\overline{P}]\overline{\Lambda}_n\overline{P}(\overline{u}_n)\;\tilde{\overline{\Lambda}}_n)^{-1}\overline{P}(\overline{x}_n)\;. \tag{8}$$

Using the induction, one can prove that for $x_0=\overline{x_0}$, $u_0=\overline{u_0}$, $v_0=\overline{v_0}$

the sequence given by (8) is identical with the sequence (3).

For the operator $\bar{P}$, the conditions of Theorem A and B are true. Indeed

1^0 $)|\ \bar{P}\ (x_0)|(\ =\ )|\Lambda P(x_0)|(\ <\ \bar{\mu}_0$;

2^0 For any $x',x'',x''',x^{IV} \in S(x_0,R)$, we have

a) $\bar{A}=[x',x'';\ \bar{P}\ ]^{-1} = (\Lambda[x',x'';P]^{-1} = I$, then $\bar{A}$ exists and $)|\ \bar{A}\ |(\ =1=\ \bar{B}$;

b) $)|[x',x'';\ \bar{P}\ ]|(\ =\ )|\Lambda[x',x'';F]|(\ <\ \bar{M}$;

c) $)|[x',x'',x''';\ \bar{P}\ ]|(\ =\ )|\Lambda[x',x'',x''';P]|(\ <\ \bar{K}$

d) $)|[x',x'',x^{IV};\ \bar{P}\ ]-[x',x'',x''';\ \bar{P}\ ]|(\ =$

$=\ )|\Lambda([x',x'',x^{IV};P]-[x'',x'',x''';P])|(\ <$

$<\ \bar{N}\ )|x^{IV}-x'''|(\ ;$

3^0 $\bar{G}_0\bar{h}_0 < 1$, where $\bar{h}_0 := \bar{B}^2\,\bar{M}\,\bar{K}\,\bar{\mu}_0 < \frac{1}{2}$ and

$$\bar{G}^2 := \frac{\bar{M}(1+\bar{B}\,\bar{K}\,\bar{\mu}_0)\,[1+\bar{B}\,\bar{K}\,\bar{\mu}_0(1+\bar{M})]}{(1+h_0)^2(1-2h_0)}\left(1+\frac{\bar{N}}{B\bar{K}^2}\right)$$

It results that the hypotesis of Theorem A are satisfied by P, hence the equation (6) has a solution $x^* \in S$, which is the limit of sequence generated by the algoritm (3) or (8), the rapidity of convergente being given by (4).

Because (6) is equivalente to (2), the statement results.

REFERENCES

1. Groze,S., Chiorean,I., *On the convergence of Method analogous to the Method of tangent hyperbolas in Fréchet Spaces*, Research Seminar, Preprint Nr.9, 1989, pp.41-49.
2. Groze,S., Janko,B., *Asupra diferenţelor divizate generalizate*, Anal. Univ. "Al.I.Cuza", Iaşi, Matematica, T.XVI, 1977, pp.375-379.
3. Rolewicz,S., *Metric linear spaces*, PWN, Warszawa, 1972.

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

SOFTWARE FOR CLASSIFICATION

CRISTIAN LENART*

Received: August 4, 1991
AMS subject classification: 68T10

Rezumat. Software pentru clasificare. Articolul prezintă un sistem de programe destinat clasificării automate a unei colecţii de obiecte caracterizate prin valorile mai multor parametri. Programele au fost elaborate de autor şi se bazează pe o serie de algoritmi din literatură, precum şi pe unii originali. Principalele componente ale sistemului sînt: extractorul de caracteristici, clasificatorul ierarhic diviziv, clasificatorul neierarhic, clasificatorul bazat pe arborescenţa de acoperire minimală şi componenta destinată interpretării calitative a partiţiilor obţinute. Pentru fiecare componentă se prezintă structura şi funcţiile ei, algoritmii implementaţi, datele de intrare şi ieşire.

0. Introduction. The aim of this paper is to describe a program system designed for pattern preprocessing, classification and interpretation of data sampled from a non-homogeneous population. The programs, which belong to the author of the paper, implement classical algorithms as well as some original ones. The main components of the system are: the pattern preprocessor, the divisive hierarchical classifier, the single-level classifier, the minimum forest classifier and the component which enables a qualitative interpretation of the obtained partitions. The input of the system is the collection of objects to be classified, characterized by the values of d variables recorded within a usual text file.

1. The pattern preprocessor. This component performs the transformation of data from the original pattern space to the feature space. The output is also a text file in which s features

* *"BABEŞ-BOLYAI" University, Faculty of Mathematics, 3400 Cluj-Napoca, Romania*

($s \leq d$) for each object are recorded.

This program has several processing options: normalization of the original patterns, Mahalanobis distance, principal component analysis and combinations of the above options. The first processing type is a simple scaling of the original variables by the overall mean and standard deviation such that they become comparable. The second option implements the Mahalanobis distance by an appropriate coordinate transformation. The principal component analysis projects the original patterns onto the eigenvectors of the covariance matrix of parameters corresponding to the first s eigenvalues in their decreasing order. A given threshold indicates what percentage of the original information should be preserved after data compression.

2. **The divisive hierarchical classifier.** This component implements the fuzzy divisive hierarchical algorithm [2], and its corresponding hard version. These algorithms perform a hierarchical descendant classification, iteratively splitting the current fuzzy (hard) cluster into two fuzzy (hard) subclusters until the clustering degree of this binary partition [3] becomes less than a given threshold; in this case, the cluster is considered homogeneous.

The input of this classifier is the file containing the features of the objects to be classified. There are three output files: one containing the hard partition (in the fuzzy case, it is obtained by defuzzification), another containing the fuzzy partition (in the fuzzy case) and the last one containing the

prototypes of the partition clusters.

The structure of the classifier is presented in fig. 1. Module HIER performs the hierarchical classification. Module SPLIT implements the generalized Fuzzy c-Means algorithm with two clusters, which is used to split the current cluster. It consists of the reiteration of modules: CENTRE (computation of

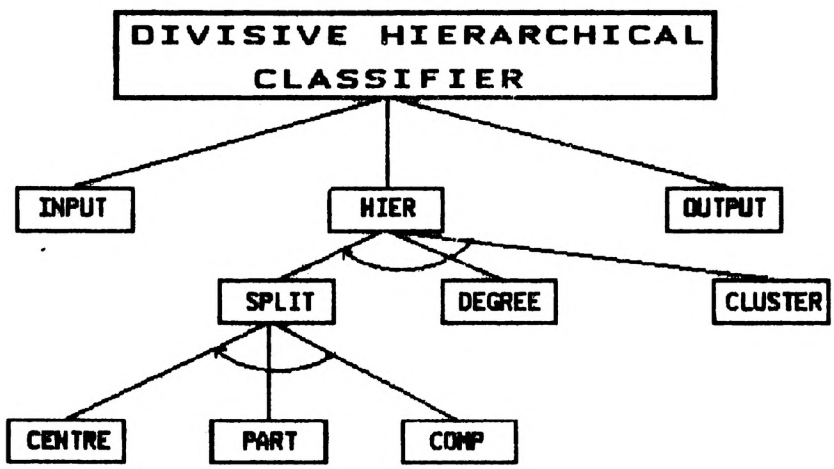


Fig.1

the sub-clusters centres), PART (computation of a new partition - fuzzy or hard) and COMP (comparison of the last two partitions). Module DEGREE computes the clustering degree of the current cluster binary partition. Module CLUSTER displays the clusters from the hierarchy and records within an output file final clusters (which are no longer splitted).

3. **The single-level classifier.** This classifier implements the following algorithms: Fuzzy c-Means, Fuzzy c-Lines, fuzzy clustering algorithms with linear manifold prototypes,

combinations of them [1] and hyperellipsoid prototypes [8], as well as their hard versions. This classifier performs a non-hierarchical classification, hence the number of clusters must be given by the user.

The input files of the classifier are: the file containing the features of the objects to be classified and a file containing an initial partition (fuzzy or hard) or an initial set of prototypes. Thus, the output of the divisive hierarchical classifier may become the input of this classifier, in order to obtain an improved partition. Moreover, if the input is a set of prototypes, this classifier may be used as a trainable classifier: the prototypes are computed from a training set and then unknown samples are classified according to the dissimilarities with respect to these prototypes. The output files are the same as those of the divisive hierarchical classifier.

The structure of this classifier is presented in Fig.2.

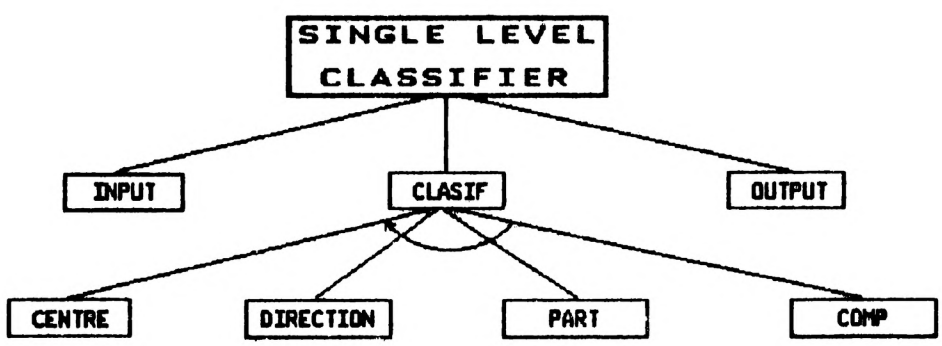


Fig.2

Module CLASIF performs the single-level classification. It consists of the reiteration of modules: CENTRE (computation of

the centres of the clusters), DIRECTION (computation of the directions of prototypes), PART (computation of a new partition - fuzzy or hard) and COMP (comparison of the last two partitions) until the last two partitions coincide (in the hard case) or the maximum difference of corresponding membership degrees does not exceed a given error level (in the fuzzy case).

4. **The minimum spanning forest (MSF) classifier.** From numerical experiments we noticed that Fuzzy c-Means and other related algorithms often misclassify samples situated at the border of the clusters. One way to prevent this situation, if the distribution of the cluster samples is close to a normal one, is to use hyperellipsoid prototypes. If the distribution is arbitrary, we propose Prim's MSF clustering algorithm [9] which is based on a graph-theoretical approach.

The MSF classifier first detects the subclusters containing samples which were surely correctly classified by a Fuzzy c-Means type algorithm and states them as "centres"; this operation, implemented in module SELECT (fig. 3), is done by selecting those samples which have the membership degree in the corresponding fuzzy cluster higher than a given threshold. The remaining samples represent the "objects" (according to the terminology used in [9]) and will be reclassified. Module DISSIM computes object-to-centre dissimilarities as point-to-set dissimilarities, i.e. the least dissimilarities between the objects and the samples in the centres. Module MSF, implementing Prim's MSF clustering algorithm, associates the objects one by one with the

centres. Thus, misclassified samples are reclassified and will probably get into the correct cluster.

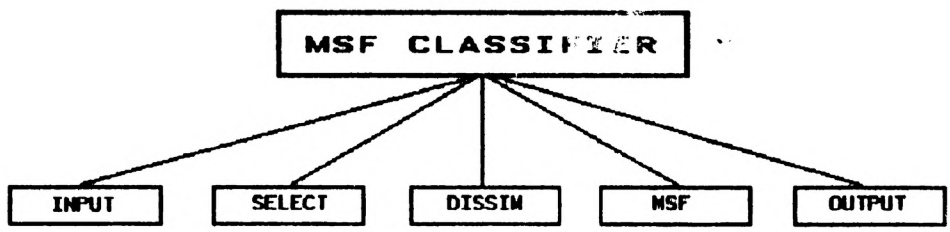


Fig.3

As input data we need the features of the samples, the fuzzy and the defuzzified partitions. The output is a hard partition.

5. **Interpretation of the obtained partitions.** We first give a theoretical model of partition qualitative interpretation. Let X be the set of samples partitioned into the clusters $A_1,\ldots,A_c$. Consider a qualitative feature (or a combination of qualitative features) F defined on X and taking a finite number of values $\{f_1, \ldots,f_k\}$ such that $k \geq c$. We are lookink for the onto function

$$\varphi : \{1,\ldots,k\} \to \{1,\ldots,c\}$$

which maximizes the cardinality of overlapping clusters from the initial partition of X and the one induced by the feature F:

$$S_\varphi = \sum_{i=1}^{k} | F^{-1}(f_i) \cap A_{\varphi(i)} |$$

This problem can be formulated as a maximum matching problem for which we apply the Hungarian algorithm. Thus the cluster A_j may

be interpreted as a mixture of the qualitative values from the set

$$\{f_i | \varphi\ (i) = j\}$$

We also define the matching degree as

$$\frac{S_\varphi}{|X|}$$

which is a sub-unitary value and characterizes the proportion in which the qualitative feature can explain the partition of X.

The component of our program system which enables the qualitative interpretation of obtained partitions enters quantitative and qualitative features of the classified samples fulfilling certain criteria concerning their features. Then a second module classifies the selected samples into groups as it was done by previous clustering procedures or according to the values of grouping features. Groups are identified as codes (for qualitative features) or intervals (for quantitative features). Quantitative features transformations can be performed. If two partitions are thus obtained, they can be compared as it was shown above. Selections, groups and transformed features can be stored in output files. Scatter plots of one quantitative feature against another can also be obtained and are useful in examining the performances of the clustering algorithms we used, the discrimination power of the two features and the regions delimited in the plane by the clusters.

6. **Facilities of the software.** This software gathers in a

unitary conception various aspects of classification: hierarchical and single-level classification, fuzzy and hard partitioning, pattern preprocessing and postprocessing, graph-theoretical methods and methods based on the minimization of a certain functional, supervised and unsupervised classification. Here are now some of the implementation facilities of this system:

- portability, as being written in Pascal language;
- simple text file structure of input and output data, which enables its use in a sequence of processing stages;
- independence of the system components, which permits interchanging their order during processing, omitting or reiterating them in order to obtain improved partitions;
- possibility of modifying the memory limits for data according to the capacity of the computer (this is done simply by modifying some constants and recompiling the programs);
- listing file option for obtaining a list with intermediate or final results;
- supplementary possibilities to limit the execution of iterative procedures by setting time, number of steps and maximum level in the classification hierarchy limits;
- other options which enable a flexible execution of programs.

This software was applied in geology, to the determination of certain types of mineralizations and to the parallelization of tuff horizons [5], in geography, to the regionalization of hydroenergetical potentials [6] and water resources [7] and in

biology, to the determination of plants associations specific to certain environment conditions [4].

REFERENCES

1. Bezdek,J.C., Coray,C., Gunderson,R., Watson,J., *Detection and characterization of cluster substructure*, SIAM J. of Appl. Math. 40 (1981) 339-371.
2. Dumitrescu,D., *Hierarchical pattern classification*, J. Fuzzy Sets and Systems 28 (1988) 145-162.
3. Dumitrescu,D., Lenart,C., *Hierarchical classification for linear clusters*, Studia Univ. "Babeş-Bolyai", Math. 3 (1988) 48-51.
4. Gardo,G., *Quantitative and qualitative structure of ligneous vegetation from the Făget forest Cluj-Napoca*, Thesis, "Babeş- Bolyai" Univ. Cluj-Napoca, 1991.
5. Ghergari,L., Lenart,C., Mârza,I., Pop,D., *Anorthitic composition of plagioclases, criterion for parallelizing tuff horizons in the Transylvanian basin*, to appear in Transylvanian Miocene Symposium.
6. Haidu,I., Lazăr,I., Lenart,C., Imbroane,A., *Modelling of natural hydroenergy organization of the small basins*, Proceedings of World Renewable Energy Congress, Sept. 1990, Reading, England, 3159-3167.
7. Haidu,I., Lenart,C., *Clustering techniques in water resources regionalization*, to appear in Annales Geophysicae.
8. Lenart,C., *A classification algorithm for ellipsoid form clusters*, "Babeş-Bolyai" Univ. Cluj-Napoca, Fac. Math. Res. Seminars 9 (1989) 93-102.
9. Lenart,C., *Graph-teoretical classification methods from a metrical point of view*, to appear in Discrete Mathematics, USA.
10. Tou,J.T., Gonzales,R.C., *Pattern recognition principles*, Addison-Wesley, 1981.

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

TOPOGRAPHICAL DATA MANAGEMENT SYSTEM

CRISTIAN LENART*

Received: August 4, 1991
AMS subject classification: 68T10

Rezumat. - Sistem de gestiune a datelor topografice. Articolul prezintă un sistem original de gestiune a datelor topografice implementat sub sistemele de operare RSX şi PC DOS. Informaţia preluată de pe hărţi o constituie coordonatele punctelor de observaţie (puncte în care s-au efectuat anumite determinări calitative sau cantitative), precum şi entităţile grafice curbe, regiuni, semne convenţionale, texte). Culegerea datelor se realizează prin digitizare, sub controlul unui editor grafic. Exploatarea bazei de date presupune extragerea şi reprezentarea datelor situate în fereastra de lucru, definirea de noi entităţi grafice, calcule simple (arii, medii ale unor funcţii de parametri cantitativi).

0. Introduction. The aim of this paper is to present an original topographical data management system for the acquisition and processing of data taken from maps. We may also consider, instead of maps, any kind of drawing consisting of curves, regions, conventional signs and texts. This system was implemented under the operating systems RSX and PC DOS.

1.Map entities. An item (data element) on a map will be called topographical entity. Two kinds of topographical entities are considered: observation points and graphical entities. Observation points are those points on a map where certain qualitative or quantitative parameters were determined. For instance, on a geological map, mineral resources and petrographical types are qualitative parameters while percentages of certain chemical elements are quantitative parameters. Graphical entities are curves (opened or closed), regions (areas

* *"BABEŞ-BOLYAI" University, Faculty of Mathematics, 3400 Cluj-Napoca, Romania*

delimited by a closed curve and filled with a certain colour), conventional signs (circles, triangles, cross-marks, etc.) and texts. Graphical entities are characterized by three features: their type, the plotting mode (for instance the type and the colour of line for curves, the filling colour for regions, the character set for texts) and the user code. This last feature is an integer associated by the user with each graphical entity in order to handle it easier. The position on the map of a graphical entity is defined by a variable number of points in a given order in the case of curves and regions and by a fixed number of points for the other two entities; thus, one point is needed to indicate the position of a standard conventional sign or text, while two points are needed for variable radius circles or inclined texts.

Observation points are numbered; their qualitative and quantitative parameters as well as the coordinates of their positions on the map are stored in separate files. A group of graphical entities is formed by a single entity with variable number of points or by several entities with fixed number of points and the same features. Three files are used to store graphical entities: an entity file, a coordinate file and a text file. The first one contains a record for each group of graphical entities; this record consists of three features of the graphical entities from the group (entity type *ET*, plotting mode *PM*, user's code *UC*) and two pointers *FP*, *LP* to the first and the last point in the coordinate file which define the position of entities. The coordinate file contains sequences of (x, y, z) - coordinates corresponding to the groups or graphical entities from the entity

file. Points indicating the position of a text have instead of the z - coordinate a pointer to the corresponding text in the text file. This file contains all the texts from the map; the end of each text and the letter of which position was indicated in the coordinate file (if not the first one) are marked. The described structure of graphical entities is given in fig. 1.

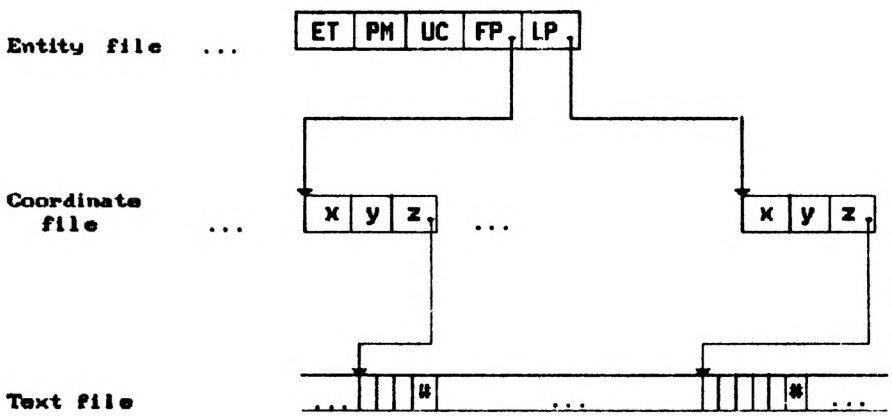


Fig. 1

2. **Data acquisition.** We are now concerned only with the acquisition of coordinates which indicate the position of topographical entities. This is done by means of a digitizer. The map may be fastened to its plane table in any position since the acquisition programs make a corrective rotation. The absolute coordinates are then computed according to the map scale and the coordinates of its origin. Thus, maps of adjacent zones can be assembled to form the general map.

Observation points can be digitized consecutively, in the order specified by the user, examining only not yet digitized points from a given interval or a certains group of points at a

time. In this latter case, we have to provide a file with the number of group to which each observation point belongs.

The acquisition of graphical entities is performed by a graphical editor which plots the entities on the display as they are stored into computer. A graphical cursor may be moved in the current window represented on the display (corresponding to a rectangular area from the map) by means of the arrow keys of the keyboard or a mouse and its coordinates are indicated. The position of the digitizer cursor on the display can also be indicated.

Graphical entities which are edited at a time are stored in the computer memory hierarchically, on four levels. This structure enables a quick performing of editing operations. Graphical entities are divided into fragments which are portions of curves or groups of conventional signs or texts placed or not in the current window. Fragment points are divided into sequences which are stored in certain memory locations called pages. We are now able to describe the tree structure of edited data (fig. 2). The first level is a two-way list of data groups referring to groups of graphical entities. Curves which do not intersect the current window at all do not appear in this list. A data group consists of the three features of the corresponding graphical entities (entity type *ET*, plotting mode, user's code *UC*) and two pointers *FF, LF* to their first and last fragments. On the second level are placed the two-way lists of data groups referring to fragments. Chaining of curve fragments observes the order in which they are placed on the curve. A data group contains the

fragment type *FT* (inside or outside the current window), a pointer *GE* to the group of graphical entities to which it belongs and two pointers *FP, LP* to the first and last point of the fragment. Only points of fragments inside the current window are stored in the internal memory; thus, for fragments outside the window, pointers *FP* and *LP* point directly to the coordinate file. On the third level are placed the two-way lists of pages, each one containing a pointer *PF* to the fragment to which it belongs and the sequence of data groups referring to points. Such a data group consists of the (*x, y, z*) - coordinates and two pointers *P1, P2*, which chain in a two-way list the points placed in the same square on the map. The number of squares into which the current window is divided is given by the user. This chaining enables a quick retrieval of points given a certain neighbourhood of them. For instance, when digitizing a point which has already been digitized, we can search for this point in a neighbourhood of the curently digitized point and replace the coordinates of the latter by those of the former. Connection of curves can thus be carried out without errors.

The following editing operations can be performed:

- acquisition of a graphical entity (coordinates from the digitizer, features and texts from the keyboard);

- prolongation of an open curve (the corresponding extremity of the curves is indicated using the graphical cursor);

- connection of two open curves or of the extremities of an open curve to form a closed one (the extremities of the curves/curve are indicated using the graphical cursor and then

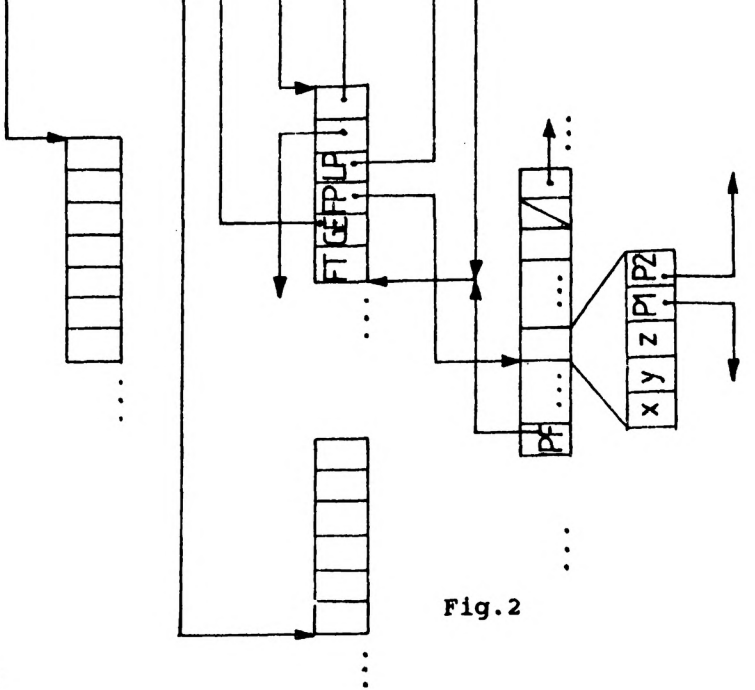


Fig.2

ET PM UC FF LF
...
FT GE FP LP
...
...
PF
...
...
...

the new curve fragment is digitized);

- modification of a curve fragment (a certain curve fragment is deleted and then its extremities are connected by a new fragment);

- deletion of a curve fragment, of a curve entirely, of a conventional sign or text (the corresponding graphical entities are indicated using the graphical cursor);

- change of the features of a graphical entity;

- hardcopy of the current window;

- asking for help on the editing menu;

Deletion and feature change operations can be performed on several graphical entities at a time without viewing their effects. These entities are also selected by indicating their features (wild cards are permitted for some of the features).

3. **Data base enquiry.** Observation points retrieval is carried out in terms of the following criteria: point number, qualitative and quantitative parameters. Qualitative parameters are specified as codes while quantitative ones as intervals. Selected observation points can be classified according to the value of a grouping parameter; hence cluster identifier is a supplementary retrieval criterion. Retrieval criteria for graphical entities are their three features. Accessing of a map entity is followed by plotting it on the display. A curve can be plotted by joining its points with straight-line segments or by smoothing it. Smoothing can be carried out using a cubic spline interpolation or an original method which iteratively halves the

angle between two neighbour straight-line segments until these segments become sufficiently small. Thus, the user can construct a map of an area he desires and containing only the information he indicates. Moreover, he can add graphical entities, perform some elementary computations and blow up a rectangular area.

Once the map is thus constructed, certain entities on it can be identified using the graphical cursor. A temporary selection of observation points is considered. We now explain how entities are identified and what kind of operations can we perform on identified entities.

a) Operations involving a single observation point:

- viewing the observation point with a given number;
- finding the number of an observation point;
- inserting/removing an observation point in/from the temporary selection;
- displaying the quantitative parameters of an observation point or elementary functions of them.

b) Operations on groups of observation points. A group of observation points is formed by the points of a cluster, by the temporary selected points or by the points placed in a rectangular or arbitrary region indicated by the user. The following operations can be performed on such groups:

- displaying the numbers of observation points from the group;
- inserting/removing the points of the group from the temporary selection;
- means computation of the group points quantitative

parameters or functions of them.

c) Operations on graphical entities.

- adding/deleting a graphical entity on/from the display (we indicate its position using the graphical cursor);

- storing/removing a graphical entity in/from the data base;

- modification of graphical entity features;

- displaying the area of a region and the number of observation points placed in it (these parameters can be used together with the quantitative parameters of the observation points to compute the value of certain elementary functions depending on them);

- displaying the coordinates of the graphical cursor.

We conclude by mentioning that the system developed in this paper is a useful tool for the management of a data base containing topographical data which can be then used by other software to construct 3*D* plottings. Referring to the parameters of observation points, the system is also useful for a primary data analysis and for the interpretation of statistical processing or clustering results.

REFERENCES

1. Lenart, C., *Software for classification*, to appear in Studia Univ. "Babeş-Bolyai" Cluj-Napoca, 1991.
2. Russu, A., *Topografie cu elemente de geodezie şi fotogrametrie*, Ed. Ceres, Bucureşti, 1974.
3. Săndulache, Al., Sficlea, V., *Cartografie - topografie*, Ed. didactică şi pedagogică, Bucureşti, 1970.

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

THEORETICAL SUPPORT FOR OBJECT-ORIENTED AND PARALLEL PROGRAMMING

ILIE PARPUCEA*

Received: November 13, 1991
AMC Subject Classification: 68005

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Since the operations specified by means of the operation schemes σ are heterogeneous operations, we consider the symbol of the operation ω to be distributed upon its operands. In other words, the operation scheme σ becomes $\sigma = <n, s_0 s_1 \ldots s_n, a_1 a_2 \ldots a_n a>$. For a complete specification of the HAS^{i+1} by the HAS^i , consider a function F which associates to each operation scheme $\sigma \in \Sigma_\omega$, $\omega \in \Omega^i$, a heterogeneous operation specific to HAS^{i+1}. If $\sigma = <n, s_0 \ldots s_n, a_1 \ldots a_n a>$, then F_σ is a specific operation in HAS_{i+1}, that is , the function:

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

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

and has the form:

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

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

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

Object name : NAME;

Supports : $NAME_1, NAME_2, \ldots, NAME_n$;

Operation schemes : $\sigma_1, \sigma_2, \ldots, \sigma_m$;

Variables : $LIST_1$: $NAMEi_1$

$LIST_2$: $NAMEi_2$

.....

$LIST_k$: $NAMEi_k$

Axioms : $w_{11} = w_{12}$.

$w_{21} = w_{22}$,

.....

$w_{p1} = w_{p2}$;

end NAME.

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

$$HAS^0 = \langle A^0, \Omega^0, H^0 : A^0 \rightarrow I \rangle,$$

where:

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

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

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

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

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

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

where:

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

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

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

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

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

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

is associated.

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

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

is a heterogeneous operation in HAS^1.

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

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

where:

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

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

N = a set containing all possible representation length;

M = a set containing all possible interpretation modes;

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

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

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

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

a heterogeneous operation on the data set A^2.

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

composed data types are specified at this level, too.

Let us denote by $OP(HAS^i)_{i=0,1,2}$ the set of the heterogeneous operations defined on the three hierarchical levels. An equivalence relation, denoted HAS^e, is defined on this set. Two operations ω_1, ω_2 $\epsilon(HAS^i)_{i=0,1,2}$ are, by definition, called equivalent if they have the same definition domain. Consider ω_1: $A_1xA_2x...A_n \to A_{n+1}, \omega_2 : B_1xB_2x...xB_n \to B_{n+1}$, $\omega_1 HAS^e \omega_2$, where

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

$i=1,2,...n+1$, and B_i is obtained in a similar manner. Follows that $A_1=B_1, A_2=B_2, \dots, A_n=B_n$ (equality of sets). Let $E = OP(HAS^i)_{i=0,1,2}/HAS^e$ be the set of the equivalence classes. An equivalence class $e \epsilon E$ consists of all operations with the same definition domain. Let $C_1xC_2x...xC_n$ be the common definition domain for the operations belonging to the equivalence class e. We call object an n-uple $(c_1, c_2, \dots, c_n) \epsilon C_1xC_2x...xC_n$, while the set of all objects of this form will be called the class of objects associated to the equivalence class e. Let us denote by K the set of all classes of objects. We add to each class of objects $k \epsilon K$ the object nil_k which constitutes the nil object of the class k. An object of the class k is either the object nil_k or an instance of the class k. The specification of a class k consists firstly of the specification of the form of the objects belonging to the class k. The form of a class k specifies the n-

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

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

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

Let us consider the heterogeneous algebra *HA=<KL,ME>*, where *KL* is the set of the classes of objects, while *ME* is the set of the methods specified in Section 3. We define the concept of algorithm over the given heterogeneous algebra HA as being the heterogeneous algebraic system *AL=<K1,Me,R>*, where:

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

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

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

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

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

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

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

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

A process $P=<$Processor$,AL>$ will be called parallel (or

concurrent) if the defined algorithm *AL* is a parallel algorithm.

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

REFERENCES

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

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

FORMAL INTEGRATION OF CERTAIN CLASSES OF FUNCTIONS

DRAGOŞ POP*

Received: November 13, 1991
AMS subject classification: 68Q40

REZUMAT. - Integrarea formală a unor clase de funcţii. Lucrarea prezintă o metodă de determinare analitică a primitivei unei funcţii raţionale. Legat de aceasta, sînt expuşi şi algoritmi de manipulare simbolică a polinoamelor precum şi de factorizare a polinoamelor peste Z[X]. Este descrisă de asemenea determinarea substituţiilor prin care problema integrării funcţiilor din anumite clase se poate reduce la cazul raţional.

1. Introduction. The symbolic computation represents the entrance in a new computer usage era, in which the computer becomes smarter and powerful enough to do complex scientific computation, for example the formal integration. We can notice here the software packages for scientific computation MACSYMA, REDUCE, MATHCAD and MATHEMATICA.

In this paper we present the formal integration of rational functions with integer coefficients ($R(x)$) and related to this, the formal integration of functions from the classes $R(\exp)$ and $R(\sin, \cos, \tan)$ where the arguments of the exp, sin, cos and tan functions have the form kx with $k \in Z$.

With these algorithms I realized a Pascal program for IBM PC compatible computers running MS-DOS, which can be easily extended for larger classes of functions.

2. Substitutions. Since the problem of the formal integration of rational functions is simpler than the same problem for another function types, we try to reduce the given

* *University of Cluj-Napoca, Department of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

function to a rational one by using suitable substitutions. For this reason the determination and the effectuation of the suitable substitution represents one of the most important part of a formal integration program.

In our case, we can apply the classical substitutions.

If the function belongs to the $R(\exp)$ class, the suitable substitution is $\exp(x)\to t$ and all the terms $\exp^m(nx)$ become t^{mn}.

If the function belongs to the $R(\sin, \cos, \tan)$ class we can transform the function to a equivalent function f from the $R(\sin, \cos)$ class. We have three cases:

$$f(-\sin, -\cos) = f(\sin, \cos)$$

$$f(-\sin, \cos) = -f(\sin, \cos)$$

$$f(\sin, -\cos) = -f(\sin, \cos)$$

The corresponding substitutions are $\tan(x)\to t$, $\cos(t)\to t$ and $\sin(x)\to t$. If our function doesn't verify any of these conditions, the suitable substitution is $\tan(x/2)\to t$.

Through these substitutions we transform our function in a $R(x)$ class function.

3. **The formal integration of a R(x) class function.** Suppose we have to integrate the function $f(x)=p(x)/q(x)$ where $p,q\in Z[x]$ are primitive polynomials, deg $p(x)<$deg $q(x)$ and $\gcd(p(x), ,q(x))=1$.

Obviously, every polynomial $q\in Z[x]$ has a unique squarefree decomposition:

$$q(x)=q_1(x)\,(q_2(x))^2\ldots(q_k(x))^k$$

where $q_i\in Z[x]$ are squarefree polynomials (some of them can be

constants 1) and $\gcd(q_i(x), q_j(x))=1$ for $1\le i, j\le k$ and $i\neq j$.

This decomposition can be obtained with Yun's algorithm described in section 4.

Using the simple fraction decomposition method, described in section 5, we obtain the polynomials $p_i(x)$ so that

$$\int \frac{p(x)}{q(x)}dx = \sum_{i=1}^{k} \int \frac{p_i(x)}{(q_i(x))^i}dx$$

Certainly, if $q_i(x)=1$ then $p_i(x)=0$.

In order to reduce the numerator's degree and to extract the rational part of the result we use the Hermite-Ostrogradsky method (described in section 6) and we determine the polynomials $s_i(x)$ and $r_i(x)$ for which:

$$\int \frac{p_i(x)}{(q_i(x))^i}dx = \frac{s_i(x)}{(q_i(x))^{i-1}} + \int \frac{r_i(x)}{q_i(x)}dx \quad 1<i\le k$$

In this moment, $\sum_{i=2}^{k} \frac{s_i(x)}{(q_i(x))^{i-1}}$ represents the rational part of the result. The remainder integrals will give us logarithmic or arctangent terms.

We need now the factorization of the polynomials q_i over $Z[x]$.

$$q_i(x) = q_{i1}(x) \ldots q_{in_i}(x)$$

where $q_{ij}(x)$ are irreducible polynomials over $Z[x]$.

This problem can be solved by using the Berlekamp-Hensel algorithm described in section 7.

Using again the simple fraction decomposition algorithm, we determine the polynomials $r_{ij}(x)$ for which:

$$\frac{r_i(x)}{q_i(x)} = \sum_{j=1}^{n_i} \frac{r_{ij}(x)}{q_{ij}(x)}$$

Now we have to compute $\int \frac{r_{ij}(x)}{q_{ij}(x)} dx$ $1 \le i \le k$, $1 \le j \le n_i$.

If $r_{ij}(x) = a * g'_{ij}(x)$ $(a \in Q)$ then the result is the logarithmic term $a \ln(q_{ij}(x))$. However, if deg $r_{ij}(x)$=deg $q_{ij}(x)$-1 we can extract a logarithmic term $\ln(q_{ij}(x))$ in order to reduce the degree of the numerator at the highest deg $s_{ij}(x)$-2.

If deg $q_{ij}(x)$=2 then we have an arctangent or a logarithmic term depending on the sign of the discriminant.

If deg $q_{ij}(x) \in \{3,4\}$ the equation $q_{ij}(x)$ can be solved through radicals and therefore we can factorize $q_{ij}(x)$ in a product of two polynomials of degree 1 or 2, over a radical extension of $Q[x]$.

If deg $q_{ij}(x)>4$ we shall search for a substitution in order to reduce the denominator's degree. Let's suppose we have to determine:

$$\int \frac{u(x)}{v(x)} dx$$

with $v \in Z[x]$ a irreducible polynomial over $Z[x]$, deg $v(x)>4$ and that we can effectuate the substitution $g(x) \to t$. In this situation there exist the polynomials $f, h \in Q[x]$ so that:

$$\frac{u(x)}{v(x)} = \frac{g'(x) f(g(x))}{h(g(x))}$$

If deg $g(x) = a$ then follows:

deg $u(x) = a-1+a$ deg $f(x)$

deg $v(x) = a$ deg $h(x)$

$$u(x) = g'(x)\ f(g(x))$$

$$v'(x) = g'(x)\ h'(g(x))$$

This relations shows that we can search $g'(x)$ (the derivative of the possible substitution $g(x)$) among the divisors of $\gcd(u(x), v'(x))$ with the property that $1+\deg g'(x) = \deg g(x)$ divides $\gcd(1+\deg u(x), \deg v(x))$.

4. **The squarefree decomposition Yun's algorithm.** It is fairly easy to show that if $q\in Z[x]$ and $q_i(x)$ is a polynomial such that it's roots are the i order roots of q, then $q_i\in Z[x]$, all the roots of $q_i(x)$ have the order 1 and $(q_i(x))^i$ divides $q(x)$.

Let's suppose that all the roots of $q(x)$ have the order less or equal to $k\in N$. In this case:

$$q(x)=q_1(x)\ (q_2(x))^2\ldots(q_k(x))^k.$$

Furthermore, since for $i\neq j$ $q_i(x)$ and $q_j(x)$ haven't common roots

$$\gcd(q_i(x),\ q_j(x)) = 1.$$

We can now see that:

$$q'(x) = q_1'(x)\ldots(q_k(x))^k+\ldots+kq_1(x)\ldots q_k'(x)\ (q_k(x))^{k-1}$$

$$c(x) =\gcd(q(x), q'(x)) =q_2(x)\ (q_3(x))^2\ldots(q_k(x))^{k-1}$$

$$r(x) = \frac{q(x)}{c(x)} = q_1(x)\,q_2(x)\ldots q_k(x)$$

$$s(x) =\gcd(c(x), r(x)) = q_2(x)\ldots q_k(x)$$

In this moment $q_1(x) = \frac{r(x)}{s(x)}$ and we see that making $q(x)\leftarrow c(x)$ and repeating the above operations until $q(x)$ become constant, we obtain the polynomials $q_1(x),\ldots,q_k(x)$. We also remark that $r,c,z\in Z[x]$.

The above relations represent the mathematical basis of the

Yun's algorithm. The complete description can be found in [2].

5. **Simple fraction decomposition algorithm.** Assume $p,u,t \in Z[x]$ and $\gcd(u(x),t(x)) = 1$. This algorithm will compute the polynomial $r \in Q[X]$ so that:

$$\frac{p(x)}{u(x)\,t(x)} = \frac{r(x)}{u(x)} + \frac{s(x)}{t(x)}$$ and deg $r(x)$ < deg $u(x)$, where $s \in Q[x]$ can be computed analogously.

From the above relation we obtain that:

$$p(x) = r(x)t(x) + u(x)s(x).$$

and

$$r(x) = r(x) \bmod u(x).$$

This implies that:

$$p(x) \bmod u(x) = r(x)t(x) \bmod u(x)$$
$$= (r(x) \bmod u(x))\ (t(x) \bmod u(x)) \bmod u(x)$$
$$= r(x)\ (t(x) \bmod u(x)) \bmod u(x).$$

Since $\gcd(u(x), t(x)) = 1$, there exist the polynomials $v,w \in Q[x]$ such that:

$$u(x)v(x) + w(x)t(x) = 1.$$

(The polynomials v and w can be computed using the Extended GCD Algorithm).

By dividing this relation by $u(x)$ we can see that:

$$w(x) = t(x)^{-1} \bmod u(x)$$

and this tells us that

$$r(x) = (p(x) \bmod u(x))\ w(x) \bmod u(x).$$

6. **The Hermite-Ostrogradski algorithm.** This algorithm computes the polynomials $a,b \in Q[x]$ so that:

$$\int \frac{p(x)}{(q(x))^n}dx = \frac{a(x)}{(q(x))^{n-1}} + \int \frac{b(x)}{q(x)}dx$$

where $p,q \in Z[x]$ and q is squarefree.

It is easy to show that $\gcd(q(x),q'(x)) = 1$ since q is squarefree. Therefore we can use the Extended GCD algorithm in order to determine the polynomials $v,w \in Q[x]$ so that:

$$v(x)q'(x) + w(x)q(x) = 1.$$

If we multiply this relation with $-p(x)/(n-1)$ and:

$$s(x) = -\frac{p(x)v(x)}{n-1}, \quad t(x) = -p(x)w(x)$$

we obtain that $s(x)q'(x) + \frac{t(x)q(x)}{n-1} = -\frac{p(x)}{n-1}$ and

$$-(n-1)s(x)q'(x) = p(x) + t(x)q(x).$$

Consequently,

$$\left[\frac{s(x)}{(q(x))^{n-1}}\right]' = \frac{s'(x)}{(q(x))^{n-1}} - \frac{(n-1)s(x)q'(x)}{(q(x))^n} =$$

$$= \frac{s'(x)}{(q(x))^{n-1}} + \frac{p(x)+t(x)q(x)}{(q(x))^n} = \frac{p(x)}{(q(x))^n} + \frac{s'(x)+t(x)}{(q(x))^{n-1}}$$

This means that if $r(x) = s'(x) + t(x)$ then

$$\int \frac{p(x)}{(q(x))^n}dx = \frac{s(x)}{(q(x))^{n-1}} - \int \frac{r(x)}{(q(x))^{n-1}}dx$$

It is now clear that using this algorithm for $n-1$ times, we will obtain

$$\int \frac{p(x)}{(q(x))^n}dx = \frac{s_1(x)}{(q(x))^{n-1}} + \ldots + \frac{S_{n-1}(x)}{(q(x))} + \int \frac{b(x)}{q(x)}dx$$

and thus $a(x) = s_1(x) + s_2(x)q(x) + \ldots + s_{n-1}(x)(q(x))^{n-2}$.

7. **The Berlekamp-Hensel algorithm.** Let $f(x) = a_n x^n + \ldots + a_1 x +$

+ a_0 be a squarefree and primitive polynomial with integer coefficients.

Also let

$S = a_0^2+\ldots+a_n^2$

$M(f) = 2^n S$ (1)

$q \geq M(f)$, $q \in Z$

The algorithm presented here computes $r \in N$ and the polynomials $u_1, \ldots, u_r \in Z[x]$ irreducible over $Z[x]$, such that

$f(x) = u_1(x)\ldots u_r(x)$.

It can be prove that if $b \in Z[x]$, $b(x) = b_0+b_1x+\ldots+b_sx^s$ and b divides f then $|b_i| < M(f)$ $i=0,s$. (see [4])

This means that if $b_i>0$ then

$$b_i = b_i \mod q \in \left(0, \frac{q}{2}\right).$$

and if $b_i < 0$ then

$$b_i \mod q = q-b_i \in \left(\frac{q}{2}, q\right) \quad (2)$$

These observations lead us to the idea that the factorization of f over $Z_q[x]$ could be fairly closed to the factorization of f over $Z[x]$, since if

$f(x) = p(x)t(x)$ with $p,t \in Z[x]$

then

$f(x) = p(x)t(x) \mod q$

and according to (2) we can determine the coefficients of $p(x)$ mod q which correspond to negative coefficients of $p(x)$.

The Berlekamp-Hensel algorithm is based on these conclusions and it has the following steps:

S1) Determine a prime number p, the least possible, for which deg $f(x) = n$ (q doesn't divide the leading coefficient of f) and f remain squarefree in $Z_p[x]$.

S2) Use the Berlekamp's algorithm (see [3]) for the factorization of $f(x)$ over $Z_p[x]$

$$f(x) \equiv u_1(x)\ldots u_s(x) \mod p$$

S3) Compute $M(f)$ given by (1).

S4) Pass from the factorization of f over $Z_p[x]$ to the factorization of f over $Z_{p^2}[x], \ldots, Z_q[x]$ using the formula given by the Hensel's lema (see [3]), until $q=p^k \geq 2M(f)$. This step computes the polynomials $u_{1k}, \ldots, u_{sk} \in Z_q[x]$ such that

$$f(x) \equiv u_{1k}(x)\ldots u_{sk}(x) \mod q$$

$$u_{ik}(x) \equiv u_i(x) \mod p, \quad i=1,s.$$

S5) Compute the product of each possible combination of $1,2,\ldots,s$ $u_{ik}(x)$ polynomials in $Z_q[x]$.
Normalise the coeficients of the product according to (2) by subtracting q from the coefficients greater than $\frac{q}{2}$.
If this normalised product divides f then it represents a factor of f and the $u_{ik}(x)$ polynomials which compose the product will be excluded from further combinations since f is squarefree.

Note that this is a polynomial time algorithm. There also exists the Kronecker's algorithm which is simpler and more intuitive but it requires exponential time and it become very inefficient for polynomials of degree greater than 5.

R E F E R E N C E S

1. Purdea,I., Pic,Gh., *Tratat de algebră modernă*, Editura Academiei, Bucureşti, (1977).
2. Davenport,J.H., *Integration Formelle*, Rapport de Recherche Nr. 375, Grenoble, (1983).
3. Knuth,D.E., *Tratat de programarea calculatoarelor*. Algoritmi seminumerici, Editura Tehnică, Bucureşti, (1983).
4. Buchberger,B., Collins,G., Loos,R., (eds), *Computer Algebra*. Symbolical and Algebraic Computation, Springer-Verlag, (1983).

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

A NEW METHOD FOR THE PROOF OF THEOREMS

DOINA TATAR*

Received: November 11, 1991
AMS subject classification: 68T15

Rezumat. În lucrare se prezintă un sistem formal de demonstrare prin respingere a teoremelor. Condiţia necesară şi suficientă impusă acestui sistem se bazează pe metoda lui J.Hsiang de demonstrare a teoremelor cu ajutorul sistemelor de rescriere a termenilor.

1. **Introduction.** Let T be a set of linguistic, algebraic or symbolic objects (as, for instance, first-order terms, programs) and let $\sim$ be an equivalence relation on T.

DEFINITION [2]. A computable function $S:T \to T$ is called a canonical simplifier for the equivalence relation $\sim$ on T iff for all s, $t \in T$:

$$S(t) \sim t$$

$$S(t) \leq t$$

(for some ordering $\leq$ on T)

$$t \sim s \to S(t) = S(s)$$

For computer algebra, the problem of constructing canonical simplifiers is basic, because of the following theorem:

THEOREM [2]. *Let T be a set of linguistic objects and $\sim$ an equivalence relation on T. Then $\sim$ is decidable iff there exists a canonical simplifier S for $\sim$.*

Let $T = T(F,V)$ be the algebra free generated by the set of variables V with the set of functions F; that is T is the minimal set of words on the alphabet $F \cup V \cup \{(,)\}$ such that:

1. $V \subseteq T$

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

2. If $f \in F$, $\alpha(f)$ is its arity, and if $t_1,\ldots,t_{\alpha(f)} \in T$, then

$f(t_1,\ldots,t_{\alpha(f)}) \in T$

Let $E \subseteq T(F,V) \times T(F,V)$ be a set of equations. By the Birkhoff theorem (1935) s and t are semantically equal in the equational theory $E(E \models s = t)$ iff s and t are provably equal in the theory $E(E \vdash s = t)$.

Let $s \sim t$ be the equivalence relation defined by $E \vdash s = t$. Then $\sim$ is decidable iff there exists a canonical simplifier S for $\sim$.

2. Associated term rewriting system and the completion. Let E be a set of equations $E \subseteq T \times T$ and let R_E a term rewriting system (TRS) obtained such that

$\ell \to r \in R_E \Rightarrow \ell = r \in E$ and

$v(r) \subseteq v(\ell)$, where $v(t)$ is the set of variables in the term (object) $t \in T$. This system will be called TRS associated with E. The rewriting relation $\vec{R}_E$ has the inverse relation, transitive closure, the reflexive-symmetric-transitive closure denoted by $\overleftarrow{R}_E$, $\vec{R}_E$ and $\overleftrightarrow{R}_E$ respectively. Also, we have:

$$\overleftrightarrow{E} = \overleftrightarrow{\vec{R}_E}$$

For a TRS denoted R let be the following definition [3], [7], [8]:

DEFINITION. R is noetherian (R has the finite termination property) iff there is no infinite chain

$$t_1 \ \vec{R}_E \ t_2 \ \vec{R}_E \ t_3 \ \vec{R}_E \ldots$$

DEFINITION. R is confluent iff $\forall\, x, y, z \in T \ \exists\, u \in T$ such that if $x \overset{*}{\vec{R}_E} z$ and $x \overset{*}{\vec{R}_E} y$ then $z \overset{*}{\vec{R}_E} u$, $y \overset{*}{\vec{R}_E} u$.

DEFINITION. If $x \in T$, $x \downarrow \ \in T$, $x \overset{*}{\vec{R}_E} x \downarrow$ and it does not exist t such that $x \downarrow \ \vec{R} t$ then $x \downarrow$ is normal form for x in TRS R (denoted $x \downarrow R$).

If R_E which is associated with a system of equation E is noetherian and confluent (i.e. complete) then, for $\forall\, x \in T$, the application $S(x) = x \downarrow R_E$ is a canonical simplifier. Then $\sim$ is decidable, and we have :

$$s \sim t \quad \text{iff} \quad s \downarrow R_E = t \downarrow R_E$$

Stated in the context of confluence, the idea of completion is straightforward:

Given a set of equations E we try to find a set of equations F such that: $\tilde{E}=\tilde{F}$ and the relation $\vec{R}_F$ is confluent.

If this set of equations do not exists, then the completion must terminate with failure or the completion is impossible.

The first completion algorithm for rewrite rules is that of Knuth-Bendix (1967). For a general formulation of this algorithm some additional notion for describing the replacement of terms in terms are needed.

DEFINITION [1],[2],[5]. Let $0(t)$ be the set of occurrences of a term t. If s, $t \in T(F,V)$ and $\upsilon \in 0(t)$ then $t[u \leftarrow a]$ is the term that derives from t if the term occurring at u in t is replaced by the term s (t/u becomes s).

DEFINITION. $s \rightarrow t$ iff there is a rule $a \rightarrow b \in R_E$ (or an equation $((a,b) \in E)$, a substitution τ and an occurrence $u \in 0(s)$ such that

$$s/u = \tau\ (a) \text{ and } t = s\ [u \leftarrow \tau\ (b)]$$

DEFINITION. The terms p and q form a critical pair in E iff

there are equations $(a_1,b_1) \in E$ and $(a_2,b_2) \in E$, an occurrence u in $0(a_1)$ and the substitution τ_1, τ_2 such that:

1. a_1/u is not a variable
2. $\tau_1(a_1/u) = \tau_2(a_2)$
3. $p = \tau_1(a_1)\ [u \leftarrow \tau_2(b_2)]$
   $q = \tau_1(b_1)$

The algorithm Knuth-Bendix is based on the

THEOREM: *A TRS noetherian R_E is confluent iff for all critical pairs (p,q) of E: $p \downarrow R_E = q \downarrow R_E$.*

Then it suggests to augment R_E by the rule $p \downarrow R_E \rightarrow q \downarrow R_E$ or $q \downarrow R_E \rightarrow p \downarrow R_E$. This process may be iterated until, hopefully, all critical pairs have a unique normal form or it may never stops: the algorithm is at least a semidecision procedure for $\sim$.

The completion algorithm for rewrite rules (Knuth-Bendix, 1967) is therefore [2]:

I n p u t : A finite set of equations E such that $\vec{R}_E$ is noetherian.

O u t p u t : 1. A finite set of equations F such that $\leftarrow\!\overset{*}{\vec{R}_E}\!\rightarrow = \leftarrow\!\overset{*}{\vec{R}_E}\!\rightarrow$ and relation $\vec{R}_F$ (therefore system R_F) is confluent (therefore is decidable) or

2. the procedure stops with failure or
3. the procedure never stops

Algorithm [2]:

1. $F := E$;

2. C: = set of critical pairs of F;

3. while $C \neq 0$ do

3.1. if $(p,q) \in C$ and $(p \downarrow R_F \neq q \downarrow R_F)$ then

3.1.1. if $p\downarrow R_F \rightarrow q\downarrow R_F$ leaves R_F noetherian then R_F:=$R_F \cup \{p\downarrow R_F \rightarrow q\downarrow R_F\}$ else if $q\downarrow R_F \rightarrow p\downarrow R_F$ leaves R_F noetherian then R_F : = $R_F \cup \{ q \downarrow R_F \rightarrow p \downarrow R_F \}$ else STOP (FAILURE)

3.1.2. C=$C \cup \{$ critical pairs in $F \cup \{(p \downarrow R_F , q \downarrow R_F)\}\}$

3.1.3. F=$F \cup \{(p \downarrow R_F = q \downarrow R_F)\}$

3.2. C:= $C \setminus \{(p, q)\}$

4. STOP(R_F).

The above crude form of the algorithm can be refined in many ways. The sequence of critical pairs chosen by the procedure in 3.1. may have a crucial influence on the efficiency of the algorithm.

3. The J. Hsiang's completion procedure. It is well known that a formula in first-order predicate calculus is valid, iff the closed Skolemized version of its negation is false under Herbrand interpretation. Equivalentely, a formula is valid if the set of the clauses in its clausal form is insatisfiable. Hsiang [7] first suggested using a complete rewrite system in a resolution-like theorem-proving strategy.

Let $C = \{C_1,\ldots,C_n\}$ the set of clauses of a formula in first-order predicate calculus.

Let $C_1 = L_1 \vee L_2 \vee \ldots \vee L_k$ be a clause where L_j is a literal, and let H be a mapping transforming terms of a Boolean algebra

into terms of a Boolean ring:

$$H(C_i) = \begin{cases} 1 & \text{if } C_i \text{ is empty clause} \\ x+1 & \text{if } C_i \text{ is } x \\ x & \text{if } C_i \text{ is } \overline{x} \\ H(L_1)*H(L_2 V \ldots V L_k) & \text{otherwise} \end{cases}$$

THEOREM (Hsiang[7]: *Given a set of clauses $\mathcal{C}$ in first-order predicate Calculus, $\mathcal{C}$ is inconsistent iff the system*

$$H(C_i) = 0,\ C_i \in \mathcal{C},\ i = 1,n$$

has not a solution.

Now, let BR be the complete TRS [7]:

$x + 0 \to 0$

$x + x \to 0$

$x * 1 \to x$

$x * 0 \to 0$

$x * x \to x$

$x *(y+z) \to x * y + x * z$

For each equation $H(C_i) = 0$ let us consider the equation $a_i = b_i$, where a_i is the biggest monomial of boolean polynomial $H(C_i)$ and let E be the system corresponding in this fashion to the system of equations:

$$H(C_i) = 0, i = \overline{1,n}$$

The TRS R_E having all the rules of the form $a_i \to b_i$ is noetherian [7]. In the TRS formed by $R_E \cup BR$ we have:

$$s \underset{H(C_i)=0}{\sim} t \Rightarrow s \underset{E}{\sim} t \Rightarrow s \overset{*}{\underset{R_E \cup BR}{\longleftrightarrow}} t$$

because $a_i = b_i$ is equivalent with $a_i + b_i = H(C_i) = 0$

A critical pair (p,q) may be added to system R_E not only in the form $p{\downarrow}R_E \rightarrow q{\downarrow}R_E$ or in the form $q{\downarrow}R_E \rightarrow p{\downarrow}R_E$, but also in the form $p'{\downarrow}R_E \rightarrow q'{\downarrow}R_E$ where p' is the biggest monomial of Boolean polynomial $P + q$. Hence, the polynomial $p + q$ is an intermediate form to study for critical pair.

Then, the previous theorem becomes:

THEOREM [7]. *A set of clauses $\mathcal{C}$,in first-order predicate calculus is inconsistent iff by Knuth-Bendix completion algorithm applied to the TRS formed by $R_E \cup BR$, where E is the set of equations $a_i=b_i$, $i = 1,\ldots,n$ (a_i is the biggest monomial of $H(C_i)$), the critical pair $1\rightarrow 0$ is obtained.* Let us observe that KB algorithm of completion is allways terminating by STOP.

4. A new method for proving a formula. Let $S = (\Sigma, F, A, R)$ be a formal system, where Σ is the alphabet for the term in a boolean ring (including + and *), F is the set of boolean polynomials, $A = \emptyset$ and R is the single deductive rule denoted "res" or $\vdash$:

$f_i, f_j \vdash f_k$ iff

$f_i, f_j, f_k \in F$ and there exist the monomials α, $\beta \in F$ and the substitution τ_1 and τ_2 such that:

$$(\alpha * \tau_1(f_i)) \downarrow BR = (\beta * \tau_2(f_j) + f_k) \downarrow BR$$

where the equality is modulo associativity and commutativity.

For this formal system the following theorem is true:

THEOREM : *Given a set of clauses $\mathcal{C} = \{C_1,\ldots,C_n\}$ in first-order predicate calculus, $\mathcal{C}$ is inconsistent if in formal system S:*

$H(C_1),\ldots,H(C_n) \vdash 1$.

The proof of theorem in propositional calculus consists of the following three propositions (the proof of theorem in predicate calculus is analogous).

PROPOSITION 1. *If* f_i, $f_j \vdash f_k$ *and* f_i, f_j, f_k *are the clause polynomials then* $H^{-1}(f_i) \wedge H^{-1}(f_j) \to H^{-1}(f_k)$.

Proof. By the assumption:

$f_i = \tilde{a}_{i_1} * \ldots * \tilde{a}_{i_k}$, where .

$$\tilde{a}_{i_s} = \begin{cases} a_{i_s}+1 \\ \text{or} \\ a_{i_s} \end{cases}, \quad s=\overline{1,k}$$

and $f_j = \tilde{b}_{j_1} * \ldots * \tilde{b}_{j_l}$, where

$$\tilde{b}_{j_t} = \begin{cases} b_{j_t}+1 \\ \text{or} \\ b_{j_t} \end{cases}, \quad t=\overline{1,e}$$

If $\tilde{a}_u = a + 1$, $\tilde{b}_v = a$, $u \in \{i_1,\ldots,i_k\}$, $v \in \{j_1,\ldots,j_e\}$: by the commutativity of operation * we can write:

$$f_i = (a + 1) * \gamma$$

$$f_j = a * \gamma$$

In boolean ring the following identity is obvious:

$$\delta*(a+1)*\gamma = \delta * a * \gamma+\delta * \gamma$$

By the comparison with the relation:

$$\alpha * f_i = \beta * f_j + f_k$$

(because $\tau_1 = \tau_2$ = the identic substitution in propositional calculus), we observe that $f_k = \delta * \gamma$, and that $H^{-1}(f_k) = H^{-1}(\delta) \vee H^{-1}(\gamma)$.

In the propositional calculus the following implications are

true:

$$(a \vee a_{i_1}^{\alpha_{i_1}} \vee \ldots \vee a_{i_k}^{\alpha_{i_k}}) \wedge (\bar{a} \vee b_{j_1}^{\alpha_{j_1}} \vee \ldots \vee b_{j_e}^{\alpha_{j_e}}) \rightarrow (a_{i_1}^{\alpha_{i_1}} \vee \ldots \vee b_{j_1}^{\alpha_{i_1}})$$

where $i_s \neq u$, $j_t \neq v$,

$$\alpha_{i_s}, \alpha_{j_t} \in (0,1), s = \overline{1,k} \quad , \quad t = \overline{1,e}$$

and

$$a_{i_s}^{\alpha_{i_s}} = \begin{cases} a_{i_s} & \text{if } \alpha_{i_s} = 1 \\ \overline{a_{i_s}} & \text{if } \alpha_{i_s} = 0 \end{cases}$$

and analogously for $b_{j_t}^{\alpha_t}$.

The above implication is therefore:

$H^{-1}(f_i) \wedge H^{-1}(f_j) \rightarrow H^{-1}(f_k)$

PROPOSITION 2. *If* $\mathfrak{C} = \{C_1, \ldots, C_n\}$ *is a set of clauses, and if:*

$H(C_1), \ldots, H(C_n) \vdash U$

U is clause polinomial, then

$C_1 \wedge \ldots \wedge C_n \rightarrow H^{-1}(U)$

Proof: To prove this proposition we proceed by induction after the length i of the deduction of U from $H(C_1), \ldots, H(C_n)$ in formal system S.

If $i = 0$, then exists j such that $U = H(C_j)$ and

$H^{-1}(H(C_j)) = C_j$.

The following implication is true:

$C_1 \wedge \ldots \wedge C_n \rightarrow C_j \ , \ j = 1, \ldots, n$

We suppose that the proposition 2 is true for the length $\leq i - 1$ of deduction, and let $f_0, \ldots, f_m = U$ a deduction of U with

the length i.

For the three last polynomials f_{m-2}, f_{m-1}, f_m in the system S there is the relation:

$$\alpha * f_{m-2} = \beta * f_{m-1} + f_m$$

Moreover, if f_m is a clause polynomial, f_{m-2} and f_{m-1} are too, and f_{m-2} and f_{m-1} are obtained by the deduction of length $\leq i - 1$.

From the induction hypothesis we have:

$$C_1 \wedge \ldots \wedge C_n \rightarrow H^{-1}(f_{m-2})$$

$$C_1 \wedge \ldots \wedge C_n \rightarrow H^{-1}(f_{m-1})$$

By the formula:

$$\vdash (A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow (A \rightarrow B \wedge C))$$

results by modus ponens:

$$\vdash C_1 \wedge \ldots \wedge C_n \rightarrow H_{-1}(f_{m-2}) \wedge H_{-1}(f_{m-1})$$

From proposition 1 we have:

$\vdash H^{-1}(f_{m-2}) \wedge H^{-1}(f_{m-1}) \rightarrow H^{-1}(f_m)$ and by the rule of syllogism

$$\vdash C_1 \wedge \ldots \wedge C_n \rightarrow H^{-1}(f_m)$$

or

$$\vdash C_1 \wedge \ldots \wedge C_1 \rightarrow H^{-1}(U)$$ q.e.d.

PROPOSITION 3. *If* $H(C_1),\ldots,H(C_n) \vdash 1$ *then* $\mathfrak{C} = \{C_1,\ldots,C_n\}$ *is inconsistent.*

Proof. From the proposition 2 we have:

$$\vdash C_1 \wedge \ldots \wedge C_n \rightarrow H^{-1}(1)$$

but $H^{-1}(1)$ is the empty clause. q.e.d.

But the condition (x) "$H(C_1),\ldots,H(C_n) \vdash 1$ iff $\mathfrak{C} = \{C_1,\ldots,C_n\}$ is inconsistent" is also true hence the implication

"$H(C_1),\ldots,H(C_n) \vdash 1 \rightarrow \mathfrak{C} = \{C_1,\ldots,C_n\}$ is inconsistent" is true even through not all the polynomials f_i, f_j, f_k in the propositions are the clause polynomials.

Exemple: (In propositional calculul $\tau_1 = \tau_2 =$ identic substitution) $\mathfrak{C} = \{P \vee \overline{Q} \vee R,\ \overline{P} \vee Q \vee \overline{R},\ \overline{P} \vee \overline{Q}, Q \vee P, P \vee \overline{R}\}$

$H(C_1) = PQR + QR + PQ + Q$

$H(C_2) = PQR + PR$

$H(C_3) = PQ$

$H(C_4) = QR + Q + R + 1$

$H(C_5) = PR + R$

$H(C_1),\ H(C_2) \vdash PR + PQ + RQ + Q$

(due to the fact that $PQR + PQ + RQ + Q = (PQR + PR) + (PR + PQ + QR + Q)$

$PQ + PR + RQ + Q,\ H(C_3) \vdash PR + RQ + Q$

$PR + RQ + Q\ ,\qquad H(C_5) \vdash RQ + Q + R$

$(PR + RQ + Q = H(C_5) + QR + Q + R)$

$H(C_4),\ RQ + Q + R \vdash 1$

This set of clauses is inconsistent, and the triplet f_i, f_j, f_k is not in each step the clause polynomials (like in proposition 1).

In fact the following observation is true: if A_i is the set of all the clauses with i positive variables (nonnegative): $C_1 \in A_i$ and $C_2 \in A_j$ are two clauses, $|i-j| \geq 2$, and $H(C_1),\ H(C_2) \vdash f_k$ then f_k is not a clause polynomial. Moreover, if $C_1 \in A_i$ and $C_2 \in A_{i+1}$ differ by a number n of variables, with $n \geq 2$, and $H(C_1),\ H(C_2) \vdash f_k$ then f_k is not a clause polynomial.

The condition (*) results from Hsiang's theorem (§ 3) by

following observations:

Let us observe that the deductive rule "res": f_i, $f_j \vdash f_k$ = $\exists\ \alpha,\beta$ (monomials) such that $(\alpha * \tau(f_i))\downarrow BR = (\beta * \tau_2(f_j) + f_k)\downarrow BR$ is a special fashion to calculate a critical pair. Indeed, the biggest monomial in $\alpha * \tau_1(f_i)$ (i.e. *MP* f_i) and the biggest monomial in $\beta * \tau_2(f_j)$ (i.e. *MP* f_j) are equal and:

$(f_k)\downarrow BR = (\alpha * \tau_1(f_i) + \beta * \tau_2(f_j))\downarrow BR = (MP\ f_i + MP\ f_j + \text{REST}\ f_i + \text{REST}\ f_j)\downarrow BR = (\text{REST}\ f_i + \text{REST}\ f_j)\downarrow BR$

This is the case $\tau_1(a_1) = \tau_2(a_2)$ and $(p,q) = (\tau_1(b_1), \tau_2(b_2))$ is a critical pair. The intermediate form $p + q$ of critical pair (in our case f_k) is studied.

THEOREM: *The set of clauses* $\mathcal{C} = \{C_1,\ldots,C_n\}$ *is inconsistent iff*

$$H(C_1),\ldots,H(C_n) \vdash 1$$

Proof: If $\mathcal{C} = \{C_1,\ldots,C_n\}$ is inconsistent, by Hsiang's theorem the system $H(C_i) = 0$, $i = 1,\ldots,n$ has not a solution, or, equivalentely, by completion in R_E the rule $1 \to 0$ is obtained. Therefore, a critical pair $(1,0)$ or $(f_k, 0)$ is obtained. We have:

$$(f_k)\ \downarrow BR = 1 = (1 + P + P)\ \downarrow BR$$

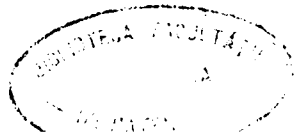
In formal system S we can write $1 + P$, $P \vdash f_k (= 1)$ where P is a boolean polynomial.

Conversely, if $H(C_1),\ldots,H(C_n) \vdash 1$ then there exists a deduction $f_0,\ldots,f_k = 1$ from $H(C_1),\ldots,H(C_n)$.

Therefore, there exists f_i and f_j such that f_i, $f_j \vdash f_k(=1)$. But f_k is a critical pair corresponding to a rule $1 \to 0$, and by Hsiang's theorem $\mathcal{C}$ is inconsistent.

R E F E R E N C E S

1. Avenchaus, J.Denzinger,J. Muller: *"THEOPOGLES - An efficient Theorem Prover based on Rewrite-Techniques"*, Dep. of Comp.Sc.University of Kaiserslautern, 1990.
2. B.Buchberger, R.Loos: *"Algebraic simplification"* Computing, Suppl. 4, 11-43, 1982.
3. B.Buchberger: *"History and Basic Features of the Critical Pair Completion procedure"*, J. Symbolic Computation, 3, 3- 38,1986.
4. J.P.Delahaye: *"Outils logigues pour l'intelligence artificielle"*, Ed. Eyrolles, 1986.
5. M.Dershowitz: *"Completion and its applications"*. "Resolution of Equations in Algebraic Structures", vol.2.
6. J.Hsiang, M.Rusinowith: *"On word problems in equational theories"* 14-th ICALP, Karlsruhe, 1987.
7. J.Hsiang: *"Refutational theorem proving using Term Rewriting System"*, Artificial Intelligence, 25, 255-300, 1985.
8. M.Jantzen: *"Confluent String Rewriting"*, EATCS, Springer Verlag, 1988.
9. J.P.Jouannaud, P.Lescanne: *"La reecriture"*, Technique et Science Inf., 6, 433-452, 1986.
10. J.Muller: *"Topics in completion Theorem Proving"*, Univ. Kaiserlautern, 1990.
11. P.Lescanne: *"Computer Experiments with REVE Term Rewriting System Generator"*, Tenth. Annual ACM Symposium on Princ. of Progr. Lang., 99-108, 1983.
12. P.Rety, C.Kirchner, H.Kirchner, P.Lescanne: *"Narower, a new-algorithm for unification and its application to logic programming"*, LNCS 202, 141-157, 1985.
13. M.Rusinowitch: *"Demonstration automatiques. Techniques de reecriture"*, Intern. Edition, Paris, 1989.
14. D.Tătar: *"Normalised rewriting systems and applications in the theory of program"*, Analele Univ. Bucureşti, nr.2, 76-80, 1989.



În cel de al XXVI-lea an (1991) *Studia Universitatis Babeş-Bolyai* apare în următoarele serii:

matematică (trimestrial)
fizică (semestrial)
chimie (semestrial)
geologie (semestrial)
geografie (semestrial)
biologie (semestrial)
filosofie (semestrial)
sociologie-politologie (semestrial)
psihologie-pedagogie (semestrial)
ştiinţe economice (semestrial)
ştiinţe juridice (semestrial)
istorie (semestrial)
filologie (trimestrial)

In the XXXVI-th year of its publication (1991) *Studia Universitatis Babeş-Bolyai* is issued in the following series:

mathematics (quarterly)
physics (semesterily)
chemistry (semesterily)
geology (semesterily)
geography (semesterily)
biology (semesterily)
philosophy (semesterily)
sociology-politology (semesterily)
psychology-pedagogy (semesterily)
economic sciences (semesterily)
juridical sciences (semesterily)
history (semesterily)
philology (quarterly)

Dans sa XXXVI-e année (1991) *Studia Universitatis Babeş-Bolyai* paraît dans les séries suivantes:

mathématiques (trimestriellement)t
physique (semestriellement)
chimie (semestriellement)
géologie (semestriellement)
géographie (semestriellement)
biologie (semestriellement)
philosophie (semestriellement)
sociologie-politologie (semestriellement)
psychologie-pédagogie (semestriellement)
sciences économiques (semestriellement)
sciences juridiques (semestriellement)
histoire (semestriellement)
philologie (trimestriellement)