JOURNAL OF COMPLEMENTARY MEDICINE RESEARCH, 2020
VOL 11, NO. 1
10.5455/jcmr.2020.11.01.41

REVIEW ARTICLE

Open Access

Implementation of Relations (Predicates) in Column-Based Intelligent Systems

Aleksandr Mikhailovich Chesnokov*, Chesnokov A.M.
Candidate of Technical Sciences,V. A. Trapeznikov Institute of Control Sciences of RAS, Moscow

ABSTRACT
The article considers the possibility of implementing relations (predicates) in column-based intelligent systems and introduces the main notions and definitions. The author suggests the presentation of relations in such systems, formulates the basic problems of implementing relations. They also suggest the solution for implementation of relations through the element-by-element comparison method and the intersection method.

ARTICLE HISTORY
Received February 23, 2020,
Accepted March 17, 2020
Published June 25, 2020

KEYWORDS
artificial intelligence, column-based intelligent systems, column, relation.

INTRODUCTION

Column-based intelligent systems are the systems considered within the following model [3, 5].

There is a relatively small, but *a finite set of names* U designated for naming random objects. Without restricting the generality, it is considered that the set of names U is a subset of the integer set Z.

In the set of names U there are non-intersecting subsets called *name domains*. The number of selected name domains is not constant. At any moment new name domains can be introduced, and more elements can be added into any name domain. Selecting name domains in real data domains can be caused by a number of reasons. For example, it can be connected with the target purpose of the names or their typification. One of the main reasons is the need to make sure that there are no random name conflicts in various parts of a big system.

Any finite set of names that belongs to a certain name domain is called a *pattern*.

The patterns of any pattern set P can be renumbered using names of a certain name domain U':

$$P = \{p_i \mid i \in U'\}.$$

An ordered pair (i, p_i) is dubbed *a column*. A column is designated as $(i \mid p_i)$, where i is the column name, p_i is the pattern contained within the column. Also, the designation $i \to p_i$ is used. In this case the column name i is said to be *a reference* or *a pointer* to the pattern contained in the column p_i. In its turn, the pattern p_i is said to have a name i or to be known by the name i.

Mapping $\varphi : i \to p_i$ is called *name mapping*. By default it is understood that name mapping is bijective (one-to-one). All cases when this is not true are specified.

To give a pattern p *a name* i or *to assign a name to a pattern* p i means to φ make an addition that $\varphi(i) = p$ to the name mapping.

A name i that was not used for pattern naming yet is called *an empty* name. It can be presented as a column with an empty pattern, i.e. a view column $(i \mid \varnothing)$ or $i \to \varnothing$, where $\varnothing$ is an empty set.

Column patterns contain names of other columns and empty names. Consequently, a column pattern completely consists of the names of other columns, each of which points to a corresponding pattern, possibly, an empty pattern. In its turn, any name from a non-empty pattern also points to its pattern, etc. As a result, a complex structure of columns is formed.

An index is any finite set of columns. The composition of any pattern can change due to

* **Contact** Aleksandr Mikhailovich Chesnokov V. A. Trapeznikov Institute of Control Sciences of RAS, Moscow
alex-ches@yandex.ru alex-ches@yandex.ru

adding or deleting columns. These operations are called *addition* and *deduction of indexes* and are designated as + and–. In the example below l of the columns $(i_k \mid p_k)$ is added to the index A:

$$A + \sum_{k=1}^{l} (i_k \mid p_k).$$

Obviously, the index can be presented as a table containing records "column name – names contained in the column pattern". Such table vertically consists of columns of variable size. In the bottom line of the table there are column names. Above each column name there are all names contained within the column pattern. By default it is assumed that column names and names in patterns belong to different name domains.

If patterns are unordered name sets, then the order of name recording in column patterns can be random. If the patterns are ordered, then the name recording in column patterns is carried out in a certain order, for example, from the bottom up, i.e. the first name of the pattern is in the first line above the line, second, in the second line, etc. Below we provide a simple example of an index A with patterns in the form of unordered sets consisting of three columns $(1|\{1, 3\})$ $(1|\{2, 3, 4\})$ and $(3|\{4, 5\})$.

A		
4		
3	3	5
1	2	4
1	2	3

A column-based intelligent system comprises one or several indexes and a mechanism that works with them, which is called *a column machine*. Receiving information about the external environment in the form of patterns, a column machine forms new columns, changes the existing ones, deletes the unnecessary ones and performs all other operations with the columns.

Knowledge in the considered systems is presented in columns, and the process of knowledge accumulation is based on memorization of new patterns under certain names. Thus, *the elementary basic problems* that needs to be solved for proper system functioning are, obviously, the *direct problem* (to obtain a pattern name by its pattern) and *the inverse problem* (to obtain a pattern by its name). Memorization of new patterns is carried out as a part of the direct problem. If when solving this problem a nameless pattern is found, then the column machine assigns a certain name to it and saves the corresponding data.

Technically, memorization of any pattern under a certain name always means forming a corresponding column $(i \mid p_i)$. At the same time, it does not mean that the data will be stored in this form in the system. The inner presentation of memorized data is defined only by the method of solving basic problems and the way of its implementation and can significantly differ from its formal description as a column $(i \mid p_i)$. An example of method for which formal description coincides with inner presentation of data is the method of solving basic problems on the basis of the element-by-element comparison of patterns [3–5]. In other cases a formed column $(i \mid p_i)$ most probably will be stored in a random form, and solving basis problems will not only show its existence, but will also help to receive its patterns by the column name, and a column name, by its pattern.

By solving basic problems, a column machine basically follows links $p_i \to i$ in the direct and $i \to p_i$ in the inverse problem. This provides the basis for solving all other problems. Solving any kind of such problem basically presents a chain of link following until the result is obtained.

Since the considered model is finite, the solution of basic problems always exists. Thus, a universal method of solving them is the afore-mentioned method of element-by-element pattern comparison. From the theoretical standpoint, this is enough to estimate the possibilities of solving different problems with the help of column-based intelligent systems. However, if we are talking about practical application of such systems, we need more efficient methods of solving basic problems, especially high dimension problems.

One of the possible methods of more efficient solving of basic problems is the *intersection method*. The idea of the intersection method goes back to book indexing. For each section there is a set of pointers to the pages of the book where you can find this section. A request relating to several sections, obviously, corresponds to intersection of sets of pointers for these sections. Such methods for pointer tables (indexes) are used in search engines [6].

In early 2000s A.M. Mikhailov demonstrated that the intersection method can be used for working with patterns [8, 9]. Within the new approach dubbed the index approach the intersection method is applied mainly when solving problems of recognition [1, 2, 10].

On the basis of the results [8, 9] a version of the intersection method was proposed to investigate column-based intelligent systems [3, 5]. For this version there were obtained necessary and sufficient conditions for solutions of the direct problem, solving which does not cut down on the universality of the method. This version of the intersection method is also characterized by the

complete absence of the necessity to compare patterns element by element.

It is important to emphasize that the intersection method is not an essential component of column-based intelligent systems. This is only one of many possible methods of solving basic problems. Instead any other methods and means can be used, in particular, hardware-software methods that assure high efficiency of solving basic problems of a certain type.

The researches [3–5] considered solving various basic problems for patterns in the form of unordered finite sets, for patterns in the form of vectors or finite sequences, as well as for patterns in the form of finite multisets [7]. They showed the ability of column-based intelligent systems to operate with incomplete information [4]. It also turned out that forecasting is an inherent property of such systems. [3, 5] proved the possibility of implementing arbitrary Boolean functions in the form $f : B^n \to B$, where $B = \{0, 1\}$. This article is dedicated to the possibility of implementing relations (predicates) $r \subset A^n$, where $A^n = A \times ... \times A$ is a finite Cartesian power of a set A. In the following chapter the author provides main definitions and considers presentation of relations in column-based intelligent systems. Then basic problems and the implementation problem are formed. Later a solution for implementing relations through the element-by-element comparison method and the intersection method is presented.

PRESENTATION OF RELATIONS

A subset $r \subset A^n$ is called *a n-local* or *n-ary relation* in a set A, where A^n is a finite Cartesian power $A^n = A \times ... \times A$. The number n is called *rank* or *arity* of the relation r. A subset $r \subset A^n$ is also called a *n-local* or *n-ary predicate* in the set A.

Let us consider a name set U and a finite Cartesian power $U^n = U \times ... \times U$. Let us assume that $r \subset U^n$ is a certain *n*-ary relation in U. It consists of a finite number of patterns $p = (i_1, ..., i_n) \in U^n$. By specifying a corresponding name domain for each coordinate in patterns $p \in r$, we can define the relation in U.

A subset $r \subset U_1 \times ... \times U_n$ is called a *n*-ary relation (predicate) in U, where U_j is a name domain of the *j*-coordinate of the patterns $p \in r$ $j = 1, ..., n$.

Let us assume that $r \subset U_1 \times ... \times U_n$ is a certain *n*-ary relation in U. All patterns of this relation $p_{rk} \in r$ can be named using names $i_{rk} \in U_p$, where U_p is a name domain for patterns p_{rk}. As a result of substituting patterns with their names the relation r becomes $r' = \{i_{r1}, ..., i_{rl}\}$, where i_{rk} is the name of the pattern p_{rk}, $k = 1, ..., l$, $l = |r|$, $|\cdot|$ is the number of elements (potency) of the set. It is obvious that the relation as a pattern $r' = \{i_{r1}, ..., i_{rl}\}$ bijectively corresponds to the relation r. Bijective mapping $\phi_r : r' \to r$ is defined as $\phi_r(r') = \{\varphi_p(i_{r1}), ..., \varphi_p(i_{rl})\}$, where $\varphi_p : i \to p$ is mapping of pattern naming $p \in U_1 \times ... \times U_n$.

To set a relation r, we need to single out only patterns $p_{rk} \in r$ $k = 1, ..., l$ out of all patterns. Obviously, it is enough to name a pattern $r' = \{i_{r1}, ..., i_{rl}\}$ by taking any pure name $i_R \in U_R$, where U_R is the name domain for relations:

$$i_R \to \underbrace{\{i_{r1}, \ldots, i_{rl}\}}_{R} \quad \begin{matrix} p_{r1} & p_{rl} \\ \uparrow & \uparrow \end{matrix}$$

Here R indicates the index that consists of columns $(i_{rk} \mid p_{rk})$, $k = 1, ..., l$.

The name i_R of the relation r' can also be interpreted as the name of the relation r. The corresponding bijective mapping $\varphi_R : i_R \to r$ equals $\varphi_R = \phi_r \circ \varphi_r$, where $\varphi_r : i_R \to r'$ is the mapping of pattern naming r', composition of mappings $(f_1 \circ f_2)(x) = f_1(f_2(x))$.

THE PROBLEM OF RELATION IMPLEMENTATION

Suppose R_n is a set of relations of the arity m, $1 \le m \le n$ and is such that all patterns p contained within these relations belong to the set of patterns $P = \bigcup_{m=1}^{n} P^m$, where $P^m = U_1 \times ... \times U_m$, U_j is the name domain of the *j*--coordinate, $j = 1, ..., n$.

Since for any $r \in \mathrm{R}_n$ the corresponding relation $r' = \{i_{r1}, ..., i_{rl}\}$ is a pattern in the form of an unordered name set, basic problems for them are formed and solved in a regular way[3–5]. Thus, *in the direct problem* for relation $r' = \{i_{r1}, ..., i_{rl}\}$ we

need to find its name i_R. In *the inverse problem* for $\forall i_R \in U_R$ we need to find the relation r'.

What is more important for estimating the possibilities of column-based intelligent systems is *the problem of relation implementation*. It is formulated in the following way.

Suppose a system memorized some number of relations (predicates) $r \in \mathrm{R}_n$. When solving the implementation problem, first for $\forall p \in P$ we need to find the names of all relations $r \in \mathrm{R}_n$ known to the system that are $p \in r$.

Solving the problem of relation implementation through the element-by-element comparison

The relation representation that we obtained above defines the solving algorithm of the implementation problem. To solve it with the help of the method of element-by-element comparison [3–5], the system uses indexes A, A_r and A_R. The index A memorizes patterns p_{rk}, the index A_r stores patterns r', and the index A_R contains names for patterns p_{rk}.

In the original state $A = \varnothing$, $A_r = \varnothing$ and $A_R = \varnothing$. Suppose there is a need to memorize a certain relation $r = \{p_{r1}, ..., p_{rl}\} \in \mathrm{R}_n$. First, we solve the direct problem for all patterns $p_{rk} \in r$. Each pattern p_{rk} is compared element by element with patterns from all columns of the index A. If a match is found, then the name i_p of the column $(i_p \mid a_p) \in A$ is such that $p_{rk} = a_p$ is the name of the pattern p_{rk}. If no match is found, then the pattern p_{rk} is new. Any pure name $i_{rk} \in U_p$ is chosen for it and a column $(i_{rk} \mid p_{rk})$ is added to the index A. The name i_{rk} is the solution of the direct problem and is the name that the pattern p_{rk} will be known under.

After all names i_{rk} have been found, we solve the direct problem for the pattern $r' = \{i_{r1}, ..., i_{rl}\}$. The pattern r' is compared element by element with patterns from all columns of the index A_r. If a match is found, then the name i_r of the column $(i_r \mid a_r) \in A_r$ is such that $r' = a_r$ is the name of the relations r' and r. This means that the relation r is already known to the system under the name i_r.

If no match has been found, then r' is a new relation that needs to be memorized. Any pure name $i_R \in U_R$ is added to it, and a column $(i_R \mid r')$ is added to the index A_r. At the same time, index A_R adds l of the columns $(i_{rk} \mid \{i_R\})$.

All other relations $r \in \mathrm{R}_n$ are memorized the same way. Moreover, because of the factorization by the name [3, 5] a pattern a_{Ri} from any column $(i \mid a_{Ri}) \in A_R$ contains the names of all relations of the same arity that contain the pattern p known under the name i.

Suppose for a certain pattern $p \in P$ there is a need to solve the implementation problem and find the names of all relations known to the system that $r \in \mathrm{R}_n$ would be $p \in r$. At first, the name of the pattern p needs to be found through its element-by-element comparison with the patterns from all columns of the index A. If no match is found, then the pattern p is a new unknown pattern. This pattern cannot belong to relations known to the system. If a match is found, then the name i_p of the column $(i_p \mid a_p) \in A$ is such that $p = a_p$ it is also a name of the pattern p.

Then the column $(i_p \mid a_{Rp}) \in A_R$ is taken. If the index A_R does not have a column under such name, then the pattern p does not belong to any of the relations known to the system. If not, the index of the column a_{Rp} contains the names of all known relations r such as $p \in r$.

As was mentioned earlier, solving any kind of problem in column-based intelligent systems basically presents a chain of link following until the result is obtained. When solving the problem of relation implementation for any pattern $p \in P$, this chain comprises only two links – one link for the direct problem and another for the inverse problem:

$$p \xrightarrow{\text{пр}} i_p \xrightarrow{\text{обр}} a_{Rp} .$$

Solving the problem of relation implementation through the intersection method

The general algorithm of solving the implementation problem stays the same. It includes solving the direct problem for patterns $p_{rk} \in r$ and $r' = \{i_{r1}, ..., i_{rl}\}$, where i_{rk} is the name of the pattern p_{rk}, $k = 1, ..., l$, $l = |r|$. For patterns p_{rk} the system uses the index $A = \{A_1, ..., A_n\}$, where

A_j is the index for *j*-coordinate, and the function $m(i_p)$ defined through the set of ordered pairs (i_p, m) that stores dimension of the known patterns p_{rk}. For patterns $r' = \{i_{r1}, ..., i_{rl}\}$ we use the index A_R and the function $l(i_R)$ defined by the set of ordered pairs (i_R, l) that contains a number of elements in the known patterns r' [3–5].

Also further the indexes B and B_R will be considered, which is not needed for solving the implementation problem. With their help it can be demonstrated how for any found relation name i_R the inverse problem can be solved and the relations r' and r can be obtained.

In the original state $A = \varnothing$, $B = \varnothing$, $m(i_p) = \varnothing$, $A_R = \varnothing$, $B_R = \varnothing$ and $l(i_R) = \varnothing$.

Suppose there is a need to memorize a certain *m*-ary relation $r = \{p_{r1}, ..., p_{rl}\} \in \mathrm{R}_n$.

First, the direct problem is solved for all patterns $p_{rk} \in r$. For each pattern $p_{rk} = (i_{k1}, ..., i_{km})$ the coordinate-by-coordinate intersection is calculated

$$\eta(A, p_{rk}) = \bigcap_{j=1}^{m} a_{kj},$$

where a_{kj} is the pattern of the column $(i_{kj} \mid a_{kj}) \in A_j$, i_{kj} is the name that is the *j*-coordinate of the pattern p_{rk}, $j = 1, ..., m$ [3–5].

If $\eta(A, p_{rk}) \neq \varnothing$ and there is at least one name $i \in \eta(A, p_{rk})$ that $m(i) = m$, then such name is the only one and is the name under which the pattern p_{rk} is known.

In any other case the pattern p_{rk} is an unknown new pattern that needs to be memorized. For this the column machine selects any pure name $i_{rk} \in U_p$, where U_p is the name domain for patterns p_{rk} and performs addition:

$$A + (p_{rk} \mid \{i_{rk}\}) = \{A_1 + (i_{k1} \mid \{i_{rk}\}), ..., A_m + (i_{km} \mid \{i_{rk}\})\},$$

$$B + (i_{rk} \mid p_{rk}),$$

where i_{kj} is the name that is the *j*-coordinate of the pattern p_{rk}, $j = 1, ..., m$. Besides, the pair (i_{rk}, m) is added to the definition of the function $m(i_p)$. The name i_{rk} is the solution of the direct problem and is the name under which the pattern p_{rk} will be known.

After all patterns p_{rk} have their found names i_{rk}, the direct problem is solved for the relation $r' = \{i_{r1}, ..., i_{rl}\}$ that is a pattern in the form of an unordered name set. For this the intersection is calculated

$$\eta(A_R, r') = \bigcap_{k=1}^{l} a_k,$$

where a_k is the pattern of the column $(i_{rk} \mid a_k) \in A_R$.

If $\eta(A_R, r') \neq \varnothing$ and there is at least one name $i \in \eta(A_R, r')$ that $l(i) = l$, then this name is the only one and is the name under which the relation r' [3–5] is known.

In any other case the relation $r' = \{i_{r1}, ..., i_{rl}\}$ is new and needs to be memorized. For this the column machine selects any pure name $i_R \in U_R$, where U_R is the name domain for relations, and performs addition:

$$A_R + (r' \mid \{i_R\}) = A_R + \sum_{k=1}^{l} (i_{rk} \mid \{i_R\}),$$

$$B_R + (i_R \mid r').$$

Besides, in the function definition $l(i_R)$ the pair (i_R, l) is added.

All other relations $r \in \mathrm{R}_n$ are memorized the same way. Because of the factorization by the name [3–5] a pattern of any column $(i \mid a_{Ri}) \in A_R$ contains the names of all relations of the same arity that contains the pattern p known under the name i.

Suppose for a certain pattern $p = (i_1, ..., i_m) \in P$ the implementation problem needs to be solved and the names of all known relations r that $p \in r$ need to be found. First, for the pattern p the direct problem is

solved. For this the coordinate-by-coordinate intersection $\eta(A,\ p)$ is calculated. If $\eta(A,\ p)=\varnothing$ or $m(i)\neq m$ for $\forall i\in\eta(A,\ p)$, then the pattern p is the new pattern that cannot belong to the relations known to the system.

If $\eta(A,\ p)\neq\varnothing$ and there is at least one name $i_p\in\eta(A,\ p)$ that $m(i_p)=m$, then this name is the only one and is the name under which the pattern p is known. Then the column $(i_p\mid a_{Rp})\in A_R$ is considered. If the index A_R does not contain such column, then the pattern p under the name i_p does not belong to any of the known relations. If such column does exist, then its pattern a_{Rp} contains the names of all known *m*--ary relations r that $p\in r$.

For any name $i_R\in a_{Rp}$ a relation known under this name can be obtained. First, the inverse problem is solved for the relation r'. The relation $r'=\{i_{r1},\ldots,i_{rl}\}=b_R$, where b_R is the column pattern $(i_R\mid b_R)\in B_R$. Then the inverse problem is solved for all names $i_{rk}\in r'$. As a result, for the relation r under the name i_R we obtain $r=\{b_{r1},\ldots,b_{rl}\}$, wheree b_{rk} is the name of the column $(i_{rk}\mid b_{rk})\in B$, $k=1,\ldots,l$ [3–5].

Example. Suppose that for relations with arity $1\le m\le 3$ the system has already memorized two binary relations:

$r_1=\{(1,2),(2,2),(2,3)\}$,

$r_2=\{(1,1),(2,2),(3,3)\}$.

They correspond to the following relations:

$r_1'=\{1,2,3\}$,

$r_2'=\{2,4,5\}$,

where 1 – is the name of the pattern p_{rk} that equals $(1,2)$, 2 is the pattern name $(2,2)$, 3 is the pattern name $(2,3)$, 4 and 5 are the pattern names $(1,1)$ and $(3,3)$.

At the same time the indexes A, B, A_R, B_R, of the functions $m(i)$ and $l(i)$ become :

A_1

4	3	6
1	2	5
1	2	3

A_2

6	2	5
4	1	3
1	2	3

A_3

1	2	3

B

2	2	3	1	3	1
1	2	2	1	3	3
1	2	3	4	5	6

$m(i)$

i	1	2	3	4	5	6
m	2	2	2	2	2	2

A_R

	2			
1	1	1	2	2
1	2	3	4	5

B_R

3	5	
2	4	
1	2	
1	2	3

$l(i)$

i	1	2
l	3	3

Suppose for the pattern $p=(1,3)$ we need to define the names of all known patterns r that $p\in r$. By solving the direct problem for the pattern $p=(1,3)$ we obtain $\eta(A,\ p)=\varnothing$, i.e. the pattern is new and cannot belong to the known relations.

Let the implementation problem be solved for the pattern $p=(3,1)$.Coordinate-by-coordinate intersection $\eta(A,\ p)=\{6\}$ and $m(6)=2$, i.e. this pattern p is known under the name 6. However, in the index A_R there is no column under the name 6.

Consequently, the pattern $p = (3, 1)$ does not belong to any of the known relations.

Finally, let the implementation problem be solved for the pattern $p = (2, 2)$. For it we obtain $\eta(A, p) = \{2\}$ and $m(2) = 2$, i.e. p is the pattern under the name 2. The pattern of the column 2 of the index A_R equals $\{1, 2\}$. This means that the pattern $p = (2, 2)$ belongs to binary relations known under the names 1 and 2.

These relations can be found. By solving the inverse problem for the patterns r' through the index B_R we obtain that the relation under the name 1 is the relation $r_1' = \{1, 2, 3\}$, and the relation under the name 2 is the relation $r_2' = \{2, 4, 5\}$. By solving the inverse problem for the patterns p through the index B it can be easily established that the pattern p under the name 1 is the pattern $(1, 2)$, the pattern under the name 2 is the pattern $(2, 2)$, etc. Consequently, the relation under the name 1 is a binary relation $r_1 = \{(1, 2), (2, 2), (2, 3)\}$, and the relation under the name 2 is the binary relation $r_2 = \{(1, 1), (2, 2), (3, 3)\}$.

REFERENCES

1. Mikhailov A. M. Recognition of patterns through their indexing. *Avtomatika i telemekhanika* [Automatics and telemechanics], 2012, No. 4, pp. 151–161. (in Russian)
2. Mikhailov A. M. Index approach to recognition of patterns and videos. *Avtomatika i telemekhanika* [Automatics and telemechanics], 2014, No. 12, pp. 139–152. (in Russian)
3. Chesnokov A.M. Column-based intelligent systems. *Upravleniye bolshimi sistemami* [Management of big systems], 2013, No. 46, pp. 118–146.(in Russian)
4. Chesnokov A.M. Column-based intelligent systems with incomplete information. *Upravleniye bolshimi sistemami* [Management of big systems], 2014, No. 50, pp. 84–98. (in Russian)
5. Chesnokov A.M. *Vvedeniye v obshchuyu teoriyu kolonok* [Introduction to the general column theory]. Moscow, Institute of Control Sciences of RAS Publ., 2012, 141 p.
6. Barroso L.A., Dean J., Hölzle U. Web Search for a Planet: The Google Cluster Architecture. IEEE Micro, 2003, Vol. 23, Iss. 2, pp. 22–28.
7. Chesnokov A.M. Finite Multisets as Patterns in Column-Based Intelligent Systems. Automation and Remote Control, 2015, Vol. 76, No. 9, pp. 1681–1688.
8. Mikhailov A., Pok Y.M. Artificial Neural Cortex. Proceedings of Artificial Neural Networks in Engineering Conference (ANNIE 2001), Nov. 4–7, 2001, St. Louis, Missouri, U.S.A.
9. Mikhailov A. Biologically Inspired Artificial Neural Cortex and its Formalism. World Academy of Science, Engineering and Technology, August 2009, Vol. 56, pp. 121.
10. Mikhailov A. Indexing-based Pattern Recognition. Advanced Materials Research, 2012, Vols. 403–408, pp. 5254–5259.