Turkish Journal of Computer and Mathematics Education Vol.12 No.2 (2021), 2771-2780

Research Article

Incomplete Information And Columns-Based Intelligent Systems

Chesnokov A.M

[1]Cand. Sc., Institute of Control Sciences of RAS, Moscow
[1]alex-ches@yandex.ru
Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract*:* The paper considers columns-based intelligent systems that work under conditions of incomplete information, that is, input patterns are represented by their sub-patterns. The definition of direct and inverse problems under incomplete information is given. The solution of these problems is shown using the method of element-wise comparison of patterns and the intersection method. A relation between system's ability to work under incomplete information and prediction is shown.
Keywords: artificial intelligence, columns-based intelligent systems, column, incomplete information, prediction.

1. Introduction

Column-based intelligent systems are an example of systems whose main function is to memorize patterns. Such systems have not become widespread, since at first glance it seems that such systems, apart from memorizing patterns, cannot solve any other problems. Column-based intelligent systems show that this is not the case at all.

Column-based intelligent systems are the systems considered within the following model [1, 2].

There is, albeit a very large, yet *a finite set of names* U, intended for naming objects of arbitrary nature. Without loss of generality, the set of names U is considered to be a subset of the set of integers Z.

In a set of names U disjoint subsets are distinguished, called *name domains*. The number of name domains allocated is not constant. New name domains can be introduced at any time, and additional elements can be added to any name domain. The allocation of name domains in real subject areas can be caused by various reasons (for example, typification). One of the main reasons is the need to make sure that there are no random name conflicts in different parts of a large-scale system.

Any finite set of names belonging to one or another name domain is called a *pattern*.

Patterns of any set of patterns P can be renumbered using the names of some name domain U':

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

An ordered pair (i, p_i) is called a *column*. A column is designated as $(i \mid p_i)$, where i is the column name, p_i is the column pattern. Also used notation $i \to p_i$. In this case, it says that the column name i is *a reference* or *a pointer* to a column pattern p_i. In turn, about the pattern p_i itself it will be said that this pattern has a name i or known by the name i.

The mapping $\varphi : i \to p_i$ is called *name mapping*. By default, the name mapping is considered to be one-to-one. All cases when this is not the case are discussed separately.

To name an pattern p with a name i, or assign a name i to the pattern p, means that the definition of the name mapping φ an addition is made such that $\varphi(i) = p$.

A name i, that has not yet been used for naming patterns is called a *pure*, or *empty* name. It can be thought of as a column with an empty pattern, i.e. a column $(i \mid \varnothing)$ or $i \to \varnothing$, where $\varnothing$ is an empty set.

Column patterns include other column names as well as pure names. Therefore, a column pattern is entirely composed of the names of other columns, each of which serves as a pointer to the corresponding pattern, possibly empty. In turn, any name from a non-empty pattern also points to its own pattern, etc. The result is a complex column structure.

An index is any finite set of columns. The composition of any index can be changed by adding or removing columns. These operations are called index *addition* and *subtraction* and are denoted by + and –. In the example below, to the index A, l columns $(i_k \mid p_k)$ are added:

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

Obviously, the index can be represented in the form of a table consisting of records of the form "column name – names included in the column pattern". Such a table in vertical form is composed of columns of variable height. In the bottom row of the table, under the line, are the names of the columns. All names included in the column pattern are listed above the name of each column. By default, column names and names in patterns are assumed to belong to different name domains.

If the patterns are unordered sets of names, then the order of the names in the column patterns can be arbitrary. Below, as a simple example, an index A with patterns in the form of unordered sets, consisting of three columns $(1 \mid \{1, 3\})$, $(1 \mid \{2, 3, 4\})$ and $(3 \mid \{4, 5\})$, is given.

	A	
	4	
3	3	5
1	2	4
1	2	3

If the patterns are ordered, then the names in the column patterns are recorded in a certain order, for example, from the bottom up, i.e. the first name of the pattern in the first row above the line, the second in the second row, etc. This writing order is adopted in this article and in other works devoted to column-based intelligent systems.

An column-based intelligent system is one or more indexes and a mechanism that operates on them, called a *column engine*. Receiving information about the external world in the form of patterns, the column engine forms new columns, modifies existing ones, deletes unnecessary ones, and performs all other operations with columns.

Knowledge in the systems under consideration is represented using columns, and the process of accumulating knowledge is based on memorizing new patterns under certain names. Obviously, elementary *basic problems*, without the solution of which the functioning of such a system is impossible, are the *direct problem* (given a pattern, obtain its name) and *the inverse problem* (given a name, obtain the corresponding pattern). 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 engine assigns a certain name to it and saves the corresponding data.

From a formal point of view, memorizing any pattern under a certain name always means the formation of the corresponding column $(i \mid p_i)$. At the same time, this does not mean that the data will be stored in this form inside the system. The internal representation of the stored data is determined only by the method of solving basic problems and the way of its implementation. Internal representation may differ significantly from the formal column description $(i \mid p_i)$. An example of a method for which the formal description coincides with the internal data representation is the method based on element-wise comparison of patterns [1, 2]. In other cases a formed column $(i \mid p_i)$ most probably will be stored in an implicit form, and solving basis problems will not only show its existence, but will also help to receive its pattern by the column name, and a column name, by its pattern.

By solving basic problems, the column engine actually implements the following links $p_i \to i$ in the direct problem and $i \to p_i$ in the inverse problem. This provides the basis on which to build the solution to all other problems. The solution to any such problem is essentially a chain of links until the result is obtained.

Since in the considered model everything is finite, the solution to 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 for more efficient solution of basic problems is the *intersection method*. The idea behind the intersection method goes back to the book index. In it, for each heading, there are many pointers to those pages of the book where this heading can be found. A query from several headings obviously corresponds to intersection of pointer sets for these headings.

In the early 2000s. A.M. Mikhailov showed that the intersection method can be used to work with patterns [6, 7]. Within the framework of the emerging direction, called the index approach, the intersection method is used mainly in solving problems of pattern recognition [8-10].

Based on the results of [6, 7], the author proposed a variant of the intersection method intended for research in the field of column-based intelligent systems [1, 2]. For this version, necessary and sufficient conditions for the existence of a solution to the direct problem were obtained, the fulfillment of which has little effect on the universality of the method. This variant of the intersection method is also characterized by the complete absence of the need for element-wise comparison of patterns.

It should be emphasized that the intersection method is not a necessary component of column-based intelligent systems. This is just one of the possible methods for solving basic problems. Instead of it, any other methods and means can be used, in particular, software-hardware, providing high efficiency in solving basic problems of one type or another.

In works [1-3] the solution of 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 was considered. In [1, 2] it was proved the possibility of implementing arbitrary Boolean functions $f: B^n \to B$, where $B = \{0, 1\}$. In the article [4] a much stronger result was proved – arbitrary functions of the kind $f: U^n \to U^m$ can be implemented in the systems under consideration, including arbitrary Boolean functions $f: B^n \to B^m$. In addition, it was shown in [5] that in column-based intelligent systems, arbitrary relations (predicates) $r \subset U^n$ can be realized, where r is a *n*-ary relation over a set U, $U^n = U \times ... \times U$ is a finite Cartesian power of a set U.

As mentioned earlier, in column-based intelligent systems, basic problem solving serves as the basis for solving all other problems. The solution to any such problem is a chain of links until the result is obtained. When solving the problem of implementing functions, such a chain consists of three links – two links of the direct problem and one link of the inverse problem [4]. When solving the problem of realizing relations, the chain of transitions is even shorter. It consists of only two links – one direct problem link and one inverse problem link [5]. The simplicity of the solution scheme allows the formation of functions and relationships dynamically during the functioning of the column-based intelligent system. As a result, as knowledge accumulates, such a system can form an increasingly accurate and complex internal representation of those functional dependencies and relationships that exist in the real world.

Working in the real world means, among other things, that the system can operate with incomplete information, when, for one reason or another, only a part of the original pattern (sub-pattern) enters the system. Therefore, in order for the system to work in the real world, it must solve basic problems under conditions of incomplete information. This work is devoted to this.

The next section describes the definition of basic problems under incomplete information. Further, the existence of a solution to these basic problems for all types of patterns is shown. Then, for patterns in the form of finite unordered sets and patterns in the form of finite sequences or vectors, the solution of basic problems under incomplete information is considered using the intersection method. Finally, in conclusion, the relation between system's ability to work under incomplete information and prediction is shown.

2. Basic problems under incomplete information

We denote by p_0 the original full pattern, and by p the pattern that came into the system and is only a part of the original pattern p_0 (sub-pattern). For patterns in the form of finite unordered sets, what has been said means that $p_0 \subset p$.

In what follows, we will assume that the column engine has additional information with which it can distinguish between complete and incomplete patterns. For patterns in the form of finite unordered sets, the number of elements of the original pattern $n_0 = | p_0 |$ can be used, where $|\cdot|$ – number of elements (cardinality) of a set. The sign of completeness in this case will be the equality $| p | = n_0$. For other types of patterns, it is most simple to distinguish between complete and incomplete patterns by replacing the missing pattern elements with special service names. An obvious sign that the pattern p is complete is the absence of the specified service names in its composition.

A simple definition of basic problems will be considered, in which parts of patterns are not memorized.
Consider first the *direct problem* under incomplete information.

Let be p – a complete pattern, i.e. $p = p_0$. In this case, the usual direct problem is solved [1, 2]. Need to get the name i of the pattern p. If the column engine manages to do this, then the name i is a solution to the direct problem. Otherwise, the pattern p is new and memorized under a certain name i_p, which in this case will be the solution to the direct problem.

Let the pattern p – now be an incomplete pattern. In this case, the column engine must specify the names of all those known patterns, of which the pattern p is a part. If it succeeds, then the set of names of such patterns S_0 is a solution to the direct problem. Otherwise, the pattern under consideration p is a part of some pattern unknown to the system.

The *inverse problem* under conditions of incomplete information remains unchanged – by the name of the pattern i it is necessary to obtain an pattern p known by this name.

3. Solving basic problems under incomplete information using the method of element-wise comparison

A general universal method for solving basic problems is a method based on element-wise comparison of patterns. To solve basic problems under incomplete information, one index A is used, which consists of columns $(i \mid p_i)$, where p_i is the pattern known by the name i.

Initially, $A = \varnothing$.

Let us denote by P the set of patterns with which the system works.

Let be $p \in P$ – some *complete* pattern for which it is necessary to solve the direct problem. It is element-wise compared against the patterns of all columns of the index A. If a match is found, then the name i_a of the column $(i_a \mid a) \in A$ is such that $p = a$, is the name of the pattern p. If no match is found, then the pattern p is new and must be memorized. Any pure name $i_p \in U_p$ is chosen for it, where U_p – name domain for patterns P, and addition is performed

$$A + (i_p \mid p),$$

that is to the index A a column $(i_p \mid p)$ is added. The name i_p is the solution to the direct problem and is the name by which the pattern p will now be known. If the pattern p reappears at the system input, then the element-wise comparison will give a name i_p for it.

Now let the pattern $p \in P$ – be some *incomplete* pattern, i.e. it is only a part of some original pattern p_0. The solution to the direct problem in this case will be a set of names S_0, which consists of the names of all complete patterns known to the system, of which the pattern p is a part. When solving the direct problem, first the set S_0 is set to the initial state $S_0 = \varnothing$. Then the incomplete pattern p is element-wise compared with the patterns of all columns of the index A. If all elements of the pattern p match the corresponding elements of the pattern a_k of the column $(i_k \mid a_k) \in A$, then name i_k added to the name set S_0. After elementwise comparison finished, set S_0 is a direct problem solution for the pattern p. If after all comparisons it turns out that $S_0 = \varnothing$, then the pattern p – it is part of some pattern unknown to the system.

The inverse problem is solved in the same way as in the case of complete information. For $\forall i \in U_p$ the pattern known by this name is equal to the pattern a_i of the column $(i \mid a_i) \in A$. If the index A there is no column with that name, then i – a pure name.

4. Solving basic problems under incomplete information using the intersection method

Intersection method for patterns in the form of finite unordered sets

We will assume that the patterns of the set P are finite unordered sets of names $p = \{i_1, ..., i_m\}$, $m \geq 1$, where $i_k \in U'$, U' – is a certain name domain.

In addition, we will assume that the true cardinality m_0, is known for each input pattern p i.e. the cardinality that the complete pattern p_0 has. Because $p \subset p_0$, then $|p| = m \leq m_0$. Obviously, the sign of the completeness of the pattern p is the equality $m = m_0$.

To solve basic problems using the intersection method, indices A, B will be used and given as a set of ordered pairs (i, m_i) function $m(i)$, which contains the cardinalities of the complete patterns known to the system.

In the initial state $A = \varnothing$, $B = \varnothing$ and $m(i) = \varnothing$.

For any $p = \{i_1, ..., i_m\} \in P$ denote by $\eta(A, p)$ intersection

$$\eta(A, p) = \bigcap_{k=1}^{m} a_k,$$

where a_k is the pattern of the column $(i_k \mid a_k) \in A$ for all names $i_k \in p$.

Let be $p \in P$ – some *complete* pattern, i.e. $|p| = m_0$. For it, the direct problem is solved in the usual way [1, 2].

If the intersection $\eta(A, p) \neq \varnothing$ and there is at least one name $i \in \eta(A, p)$, for which $m(i) = |p|$, then such a name is unique and is the name by which the pattern p is known.

In all other cases, the complete pattern $p = \{i_1, ..., i_m\}$ is new and must be memorized. The column engine chooses some pure name $i_p \in U_p$ for it, where U_p – name domain for naming patterns P, and performs additions:

$$A + (p \mid \{i_p\}) = A + \sum_{i_k \in p} (i_k \mid \{i_p\}),$$

$$B + (i_p \mid p),$$

i.e. to index *A* added m columns $(i_k \mid \{i_p\})$ for all names $i_k \in p$, and to the index B column $(i_p \mid p)$. In addition, in the function definition $m(i)$ pair (i_p, m) is added. The name i_p is the direct problem solution and is the name by which the complete pattern p will now be known.

Let now p – *incomplete* pattern, i.e. $|p| < m_0$. In this case, it is necessary to find a set of names of complete patterns known to the system, of which the pattern p is a part. Obviously, for any non-empty subset p, included in some known pattern, the intersection $\eta(A, p) \neq \varnothing$. Therefore, if $\eta(A, p) = \varnothing$, then incomplete pattern p is a subset of the unknown pattern.

If $\eta(A, p) \neq \varnothing$ and $m(i) \neq m_0$ for $\forall i \in \eta(A, p)$, then the pattern p is also part of an unknown pattern.

Finally let $\eta(A, p) \neq \varnothing$ and there is at least one name $i \in \eta(A, p)$ such that $m(i) = m_0$. In this case, the name set

$$S_0 = \{ i \in \eta(A, p) \mid m(i) = m_0 \}$$

is a solution to the direct problem. For $\forall i \in S_0$ the pattern known by this name contains p as a subset and has cardinality m_0.

The inverse problem is solved in the usual way [1, 2]. For any name $i \in U_p$ the pattern that is known by this name is equal to the pattern b_i of the column $(i \mid b_i) \in B$. If the index B does not have a column with this name, then i – a pure name.

5. Intersection method for patterns in the form of finite sequences or vectors

Let us now consider the application of the intersection method for solving basic problems under incomplete information for patterns in the form of finite sequences or vectors $p = (i_1, i_2, ..., i_m)$, $1 \le m \le n$. The pattern p may be known only partially and contain a special service name i_{0k} or simply 0 instead of some coordinates. The name 0 is interpreted as the absence of data on the corresponding coordinate. For example, $(0, i_2, ..., i_m)$ or $(0, i_2, ..., i_{m-1}, 0)$. Thus, any pattern p belongs to the set of patterns

$$P = \bigcup_{k=1}^{n} P^k,$$

where $P^k = U_1^0 \times ... \times U_k^0$, $U_k^0 = U_k \bigcup \{0\}$, U_k – k-th coordinate name domain.

To solve basic problems using the intersection method, the index $A = \{A_1, A_2, ..., A_n\}$ will be used, where A_k – k-th coordinate index, index B, and also given by a set of ordered pairs (i, m_i) function $m(i)$, which contains the dimensions of the complete patterns known to the system.

Initially, $A = \varnothing$, $B = \varnothing$, $m(i) = \varnothing$.

Let be $p = (i_1, i_2, ..., i_m) \in P$ – an arbitrary pattern for which it is necessary to solve the direct problem

Let us denote by $\eta'(A, p)$ intersection

$$\eta'(A, p) = \bigcap_{\substack{k=1 \\ i_k \ne 0}}^{m} a_k,$$

where a_k is the pattern of the column $(i_k \mid a_k) \in A_k$, i_k is the name that is the k-th coordinate of the pattern $p = (i_1, i_2, ..., i_m)$.

Obviously, if the pattern p is complete and does not contain zero coordinates, then $\eta'(A, p)$ – this is a common intersection

$$\eta(A, p) = \bigcap_{k=1}^{m} a_k.$$

Therefore, for complete patterns, the usual intersection method can be used [1, 2].

So, let it be necessary to solve the direct problems for the *complete* pattern $p = (i_1, ..., i_m) \in P$, which has no zero coordinates.

If $\eta'(A, p) \ne \varnothing$ and there is at least one name $i \in \eta'(A, p)$ such that $m(i) = m$, then such a name is unique and is the name by which the pattern p is known.

In all other cases, the pattern p – unknown complete pattern and must be memorized. To do this, the column engine chooses any pure name $i_p \in U_p$, where U_p – name domain for naming patterns P, and performs additions:

$$A + (p \mid \{i_p\}) = \{A_1 + (i_1 \mid \{i_p\}), ..., A_m + (i_m \mid \{i_p\})\},$$
$$B + (i_p \mid p),$$

where i_k is the name that is the *k*-th coordinate of the pattern $p = (i_1, i_2, ..., i_m)$. In addition, in the function definition $m(i)$ pair (i_p, m) is added. The name i_p is the solution to the direct problem and is the name by which the complete pattern p will now be known.

Now let the direct problem be solved for an *incomplete* pattern $p = (i_1, i_2, ..., i_m) \in P$, which has coordinates containing name 0.

First of all, it should be noted that for any complete pattern of the dimension m known to the system and any nonempty subset, $K \subset \{1, ..., m\}$ the intersection

$$\eta'(A, p) = \bigcap_{k \in K} a_k \neq \varnothing .$$

Therefore, if for an incomplete pattern p intersection $\eta'(A, p) = \varnothing$, then the incomplete pattern p is part of the unknown pattern.

If $\eta'(A, p) \neq \varnothing$ and $m(i) \neq m$ for $\forall i \in \eta'(A, p)$, then incomplete pattern p also part of an unknown pattern.

Finally, if $\eta'(A, p) \neq \varnothing$ and there is at least one name $i \in \eta'(A, p)$ such that $m(i) = m$, then the set of names

$$S_0 = \{ i \in \eta'(A, p) \mid m(i) = m \}$$

is a solution to the direct problem. This set contains the names of all dimension m known patterns, for which the corresponding coordinates coincide with the nonzero coordinates of the pattern p.

The inverse problem is solved in the usual way [1, 2]. For any name $i \in U_p$ the pattern that is known by this name is equal to the pattern b_i of the column $(i \mid b_i) \in B$. If in the index B there is no column with that name, then i – a pure name.

Obviously, the results obtained are valid for time sequences $p_t = (i_1, i_2, ..., i_m)$, where i_k – this is the name of the pattern that describes the state of some process at the *k*-th moment of time. The most important consequence of this is that a system that is able to solve basic problems under incomplete information has the ability to make a prediction. According to the part of the time sequence available at the *k*-th moment of time $(i_1, ... i_k, 0, ..., 0)$ the system can receive $|S_0|$ future process development options. Each of these options is a complete known sequence $p_t = (i_1, ..., i_k, i_{k+1}, ..., i_m)$, whose name is $i_{pt} \in S_0$. It should be noted that in the part of the sequence available at the *k*-th moment of time $i_1, ..., i_k$ there can also be names equal to 0. Thus, in this case we are talking about a prediction under conditions of incomplete information.

As already mentioned, in column-based intelligent systems, the solution to any problem is a chain of links until the result is obtained. To make a prediction, such a chain consists of only two links – one direct problem link and one inverse problem link.

$$(i_1, ..., i_k, 0, ..., 0) \xrightarrow{\text{dir}} \overset{S_0}{\left\{ \begin{matrix} i_{pt} \\ ... \end{matrix} \right\}} \xrightarrow{\text{inv}} (i_1, ..., i_k, i_{k+1}, ..., i_n)$$

Example. Let be $n = 3$ and there are indices shown below $A = \{A_1, A_2, A_3\}$, B and function $m(i)$.

A_1		
4		2
1		3
1	2	3

A_2		
2		
3	4	1
1	2	3

A_3		
		4
	3	2
1	2	3

B			
	3	2	2
3	1	1	2
1	3	3	1
1	2	3	4

$m(i)$

i	1	2	3	4
m	2	3	3	3

It is easy to see that the system has memorized 4 patterns: pattern $p_1 = (1, 3)$ under the name 1, pattern $p_2 = (3, 1, 3)$ under the name 2, pattern $p_3 = (3, 1, 2)$ under the name 3 and pattern $p_4 = (1, 2, 2)$ under the name 4.

Let the complete pattern be given $p = (3, 1)$. Intersection $\eta'(A, p) = \{2, 3\}$. Wherein $m(2) \neq 2$ and $m(3) \neq 2$. Therefore, the pattern $p = (3, 1)$ – this is an unknown complete pattern to be memorized. After memorizing it under the name 5 indices $A = \{A_1, A_2, A_3\}$, B and the function $m(i)$ will take the form.

A_1		
		5
4		2
1		3
1	2	3

A_2		
5		
2		
3	4	1
1	2	3

A_3		
		4
	3	2
1	2	3

B				
	3	2	2	
3	1	1	2	1
1	3	3	1	3
1	2	3	4	5

$m(i)$

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

If the full pattern $p = (3, 1)$ appears again at the entrance, the intersection $\eta'(A, p) = \{2, 3, 5\}$ and $m(5) = 2$, i.e. pattern $p = (3, 1)$ – this is an pattern by name 5.

Let an incomplete pattern $p = (1, 0, 3)$ be given. Because $\eta'(A, p) = \varnothing$, then incomplete pattern p is part of some pattern unknown to the system.

Now let an incomplete pattern $p = (3, 1, 0)$. Intersection $\eta'(A, p) = \{2, 3, 5\}$ and $m(2) = m(3) = 3$, i.e. set of names $S_0 = \{2, 3\}$. Therefore, an incomplete pattern $p = (3, 1, 0)$ nonzero coordinates coincides with the patterns of the same dimension that are known to the system under the name 2 and 3. Solving the inverse problem, we find that these patterns are equal to the patterns of the columns $(2 \mid b_2), (3 \mid b_3) \in B$, i.e. these are patterns $(3, 1, 3)$ and $(3, 1, 2)$. For time sequences, this will mean that for the sequence available at time 2 $p = (3, 1, 0)$ possible scenarios are sequences $(3, 1, 3)$ and $(3, 1, 2)$.

6. Results

In the real world, the system should be able to work under conditions of incomplete information, when, for one reason or another, only a part of the original pattern (sub-pattern) enters the system. In column-based intelligent systems, basic problems form the basis for all other problems. Therefore, in order for such systems to work in the real world, they must solve basic problems under incomplete information.

The paper proposes a simple definition of basic problems under conditions of incomplete information. It is assumed that the system, in addition to the received pattern, has additional information that makes it possible to distinguish between complete and incomplete patterns. In this case, incomplete patterns in the system are not memorized.

For complete patterns, the setting of basic problems does not change. In the *direct problem*, for a complete pattern p you need to get its name. If the pattern p – nameless full pattern, then it must be memorized. In the *inverse problem* for pattern name i you need to get a complete pattern known by this name.

For incomplete patterns, the direct problem changes. For an incomplete pattern p you need to get a set of names S_0, consisting of the names of all complete patterns known to the system, of which the pattern p is a part.

Using the method based on element-by-element comparison of patterns, a solution to basic problems under incomplete information was obtained for any types of patterns. Using a more efficient intersection method, we show the solution of basic problems under conditions of incomplete information for patterns in the form of finite unordered sets $p = \{i_1, ..., i_m\}$ and patterns in the form of finite sequences or vectors $p = (i_1, i_2, ..., i_m)$, $1 \le m \le n$.

The presented methods for solving basic problems under conditions of incomplete information represent more general variants of methods [1, 2], which for incomplete patterns are supplemented by the construction of a set of names S_0. In this case, the structure of the solution remains practically unchanged. Element-wise comparison and intersection are performed for those elements of the incomplete pattern that are available. For example, in finite sequences or vectors $p = (i_1, i_2, ..., i_m)$, $1 \le m \le n$, the missing elements are replaced with the service name 0. To build the set S_0 the intersection

$$\eta'(A, p) = \bigcap_{\substack{k=1 \\ i_k \neq 0}}^{m} a_k ,$$

is used, which is part of the regular intersection $\eta(A, p)$ for available elements of pattern p, where a_k is the pattern of the column $(i_k \mid a_k) \in A_k$, i_k is the name that is the k-th coordinate of the pattern $p = (i_1, i_2, ..., i_m)$. Moreover, the set of names

$$S_0 = \{ i \in \eta'(A, p) \mid m(i) = m \},$$

where $m(i)$ is the dimension of the complete pattern, which is known by name i.

Critical implication for temporal sequences $p_t = (i_1, i_2, ..., i_m)$ is that a system that is able to solve basic problems under incomplete information has the ability to make a prediction. According to the part of the time sequence available at the k-th moment of time $(i_1, ... i_k, 0, ..., 0)$ the system can receive $|S_0|$ future scenarios. Each of these options is a known complete sequence $p_t = (i_1, ..., i_k, i_{k+1}, ..., i_m)$, whose name $i_{pt} \in S_0$. Moreover, we are talking about a prediction under conditions of incomplete information, since in the part of the sequence available at the k-th moment of time $i_1, ..., i_k$ there can also be names equal to 0.

The scheme for solving the prediction problem is very simple. The chain of following links when solving it consists of only two links – one link of the direct problem $(i_1, ... i_k, 0, ..., 0) \to i_{pt}$ and one link of the inverse problem $i_{pt} \to p_t = (i_1, ..., i_k, i_{k+1}, ..., i_m)$. As a result, as knowledge accumulates, an columns-based intelligent systems can not only form an ever more accurate and complex internal representation of those functional dependencies and relationships that exist in the real world, but also expand its predicting capabilities.

7. Conclusion

The methods for solving basic problems presented in the work represent a more general version of the methods from [1, 2], which includes the ability of the system to work under incomplete information. The most important consequence of this is that the ability of the system to work under incomplete information at the same time means the ability of the system to predict, since from the currently available part of the time sequence, the system can restore options for the development of events in the future. Moreover, the part of the sequence available to the present moment may be known only partially, i.e. we are talking about a prediction under incomplete information. Thus, an elementary baseline prediction is an intrinsic property of a system capable of solving basic problems under conditions of incomplete information. This demonstrates one of the main features of the column-based model – its versatility, where the same mechanism serves different purposes.

References

1. Chesnokov A.M. Column-Based Intelligent Systems (Intellektual'nye sistemy na osnove kolonok) // Upravlenie bol'shimi sistemami (Large-Scale Systems Control), 2013, No. 46, pp. 118–146.
2. Chesnokov A.M. Vvedenie v obshchuyu teoriyu kolonok (Introduction to General Columns Theory). – M.: IPU RAN publ., 2012.
3. Chesnokov A.M. Finite Multisets as Patterns in Column-Based Intelligent Systems // Automation and Remote Control, 2015, Vol. 76, No. 9, pp. 1681–1688.
4. Chesnokov A.M. Functions in Column-Based Intelligent Systems // International Journal of Engineering and Advanced Technology (IJEAT), Vol. 9 Issue 2, December 2019, pp. 5045-5051.
5. Chesnokov A.M. Implementation of Relations (Predicates) In Column-Based Intelligent Systems // Journal of Advanced Research in Dynamical and Control Systems (JARDCS), Vol. 12, Special Issue-02, 2020, pp. 518-526.
6. 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.
7. Mikhailov A. Biologically Inspired Artificial Neural Cortex and its Formalism // World Academy of Science, Engineering and Technology, August 2009, Vol. 56, p. 121.
8. Mikhailov A. Indexing-based Pattern Recognition // Advanced Materials Research. – 2012, Vols. 403–408, pp. 5254–5259.
9. Mikhailov A.M. Pattern recognition by indexing // Automation and Remote Control, 2012, Vol. 73, No. 4, pp. 717–724.
10. Mikhailov A.M. An indexing-based approach to pattern and video clip recognition // Automation and Remote Control, 2014, Vol. 75, No. 12, pp. 2201–2211.