Original Article

Pattern Regions As A Basis For Logical Inference In Columns-Based Intelligent Systems

Chesnokov A.M.[1]

[1]Cand. Sc., Institute of Control Sciences of RAS, Moscow, Email: alex-ches@yandex.ru

Address for correspondence: Chesnokov A.M., Cand. Sc., Institute of Control Sciences of RAS, Moscow, Email: alex-ches@yandex.ru

DOI: 10.47750/pnr.2023.14.S01.46

Abstract

In columns-based intelligent systems, basic problems serve as the basis for solving all other problems. The article deals with the solution of basic problems for the pattern regions, which originally arose as a natural means of ensuring operation in the presence of interference. Basic concepts and definitions are given. The formulation of basic problems for pattern regions is described and their solution is given using a general universal method based on an element-by-element comparison of patterns. Further, the solution of basic problems for pattern regions using a more efficient intersection method is considered. The solution scheme and an explanatory example are described in detail. In conclusion, it is shown that memorizing pattern regions under certain names means memorizing the IF-THEN rules, in the conditional part of which connectives AND, OR, NOT and variables with a universal quantifier are used.

Keywords: artificial intelligence, columns-based intelligent systems, column, logical inference

1. Introduction

Pattern regions have emerged as a natural means of solving basic problems under interference conditions. In columns-based intelligent systems, basic problems serve as the basis for solving all other problems.

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

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

In the set of names U non-overlapping subsets are distinguished, which are called name domains. The number of allocated name domains 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, typing). One of the most important reasons is the need to ensure that there are no accidental coincidence of names in different parts of a large-scale system.

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

The patterns of any set of patterns P can be renumbered using the names of some name domain U_p:

$P = \{p_i \mid i \in U_p\}$.

An ordered pair (i, p_i) is called a column. The column is denoted as $(i \mid p_i)$, where i – is the column name, p_i – is the column pattern. The notation $i \to p_i$ is also used. In this case, the column name i is said to be a reference or pointer to a column pattern p_i. In turn, it will be said about the pattern p_i itself that this pattern has a name i or is known by the name i. (It should be noted that the element $p_i \in P$, obviously corresponds to the pair (i, p) and column $(i \mid p)$. The notation $(i \mid p_i)$ is redundant and is used for convenience.)

The mapping $\varphi : i \to p_i$ is called name mapping.

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

The name i, which has not yet been used to name patterns is called a pure or empty name.

By default, the name mapping is considered to be one-to-one. All cases where this is not the case are considered separately.

So, if the same pattern p has several names i_k, then the so-called pattern factorization takes place. To name mapping $\varphi : i \to p$ all names i_k of the pattern p form an equivalence class $[i] = \{i_k \in U_p \mid \varphi(i_k) = p\}$. The one-to-one correspondence between the pattern name and the pattern is preserved, but in the form $\varphi : [i] \to p$, where the name mapping $\varphi : [i] \to p$ is defined as $\varphi([i]) = \varphi(i_k)$ for $\forall i_k \in [i]$. As a result, several columns $i_k \to p$ can be replaced with a single column $[i] \to p$:

$$\begin{array}{ccc} p & \ldots & p \\ \uparrow & \ldots & \uparrow \\ i_1 & \ldots & i_m \end{array} \Rightarrow \begin{array}{c} p \\ \uparrow \\ [i] = \{i_1, \ldots, i_m\} \end{array}$$

If several patterns p_k have the same name i, then factorization by name takes place. In this case, we consider the inverse mapping $\varphi^{-1} : p \to i$. All patterns p_k form an equivalence class $[p] = \{p_k \in P \mid \varphi^{-1}(p_k) = i\}$. The one-to-one correspondence between the pattern and its name is preserved, but in the form $\varphi^{-1} : [p] \to i$, where the inverse mapping $\varphi^{-1} : [p] \to i$ is defined as $\varphi^{-1}([p]) = \varphi^{-1}(p_k)$ for $\forall p_k \in [p]$. As a result, several columns $i \to p_k$ can be replaced by a single column $i \to [p]$:

$$\begin{array}{ccc} p_1 & \ldots & p_m \\ \uparrow & \ldots & \uparrow \\ i & \ldots & i \end{array} \Rightarrow \begin{array}{c} [p] = \{p_1, \ldots, p_m\} \\ \uparrow \\ i \end{array}$$

Moreover, if each of these patterns p_k consists of only one name $p_k = \{i_k\}$, then as a result of factorization by name, several columns $i \to \{i_k\}$ can be replaced by one column $i \to \{i_1, \ldots, i_m\}$, since $[p] = \{i_1, \ldots, i_m\}$:

$$\begin{array}{ccc} \{i_1\} & \ldots & \{i_m\} \\ \uparrow & \ldots & \uparrow \\ i & \ldots & i \end{array} \Rightarrow \begin{array}{c} \{i_1, \ldots, i_m\} \\ \uparrow \\ i \end{array}$$

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 addition and subtraction and are denoted by + and –. In the example below, to the index A is added l columns $(i_k \mid p_k)$:

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

If in the above example the patterns p_k are equal and consist of one name $p_k = \{i\}$, then due to factorization by the pattern, the above sum can be written as:

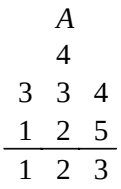
$$A + \sum_{k=1}^{l} (i_k \mid \{i\}) = A + (p \mid \{i\}),$$

where the pattern $p = \{i_1, \ldots, i_l\}$.

Obviously, the index can be represented as a table consisting of entries of the form "column name – names included in the column pattern". Such a table in vertical form consists of columns of variable height. In the bottom row of the table, under the line, are the names of the columns. Above the name of each column, all the names included in the column pattern are listed. By default, column names and names in patterns are considered 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 the simplest example, there is an index A with patterns in the form of unordered sets, consisting of three columns $(1 \mid \{1, 3\})$,

$(2|\{2,3,4\})$ and $(3|\{4,5\})$.



If the patterns are ordered, then the names in the column patterns are written in a certain order, for example, from bottom to top, i.e. the first pattern name is in the first row above the line, the second one is in the second row, and so on. Such an order of notation is adopted in this article and in other works devoted to column-based intelligent systems.

A columns-based intelligent system consists of one or more indexes and a mechanism that works with them, called a column engine. Receiving information about the outside world in the form of patterns, the column engine forms new columns, modifies existing ones, removes unnecessary ones, and performs all other operations with columns.

Knowledge in the systems under consideration is represented by columns, and the process of accumulation of knowledge is based on the memorization of new patterns under certain names. Obviously, the elementary basic problems, without the solution of which the functioning of such a system is impossible, are the direct problem (given 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 the direct problem, an unnamed 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 $(i|p_i)$. At the same time, this does not mean that data will be stored in this form inside the system. The internal representation of the stored data is determined only by the method for solving basic problems and the way this method is implemented. The internal representation may differ significantly from the formal description in the form of a column $(i|p_i)$. An example of a method for which the formal description coincides with the internal representation of data is the method based on element-by-element pattern comparison. In other cases, the formed column $(i|p_i)$, will most likely be stored in an implicit form, and the solution of basic problems will not only show its existence, but also allow you to get its pattern by the column name, and its name by the pattern.

Solving basic problems, the column engine actually implements in the direct problem the transition from an pattern to a name by reference $p_i \to i$ and in the inverse problem the transition from a name to an pattern by reference $i \to p_i$. This provides the basis on which the solution of all other problems is built. The solution to any such problem is a sequence of references until the result is obtained.

Since everything is finite in the model under consideration, the solution of basic problems always exists. At the same time, the already mentioned method based on element-by-element pattern comparison is a general universal method for solving them. From the point of view of theory, this is enough to evaluate the possibilities of solving various problems using columns-based intelligent systems. However, if we talk about the practical application of such systems, then more efficient methods for solving basic problems, especially problems of large dimensions, are needed.

One of the possible methods for more efficient solution of basic problems is the intersection method. The idea of the intersection method goes back to the book index. In it, for each rubric, there are many pointers to those pages of the book where this rubric can be found. A query from several headings obviously corresponds to the intersection of sets of pointers for these headings.

In the early 2000s A.M. Mikhailov showed that the intersection method can be used to pattern identification [8, 9]. Within the framework of the emerging direction, called the index approach, the intersection method is used mainly in solving problems of pattern recognition [10, 11].

Based on the results of [8, 9], the author proposed a variant of the intersection method intended for research in the field of columns-based intelligent systems [1, 2]. It was used to solve basic problems for patterns in the form of unordered finite sets, for patterns in the form of finite sequences or vectors [7], and also for patterns in the form of finite multisets [3]. It was used to show the possibility of implementation in columns-based intelligent systems of arbitrary functions $f:U^n \to U^m$, including arbitrary Boolean functions $f:B^n \to B^m$, where $B=\{0,1\}$ [4]. In addition, the possibility of implementing arbitrary relations (predicates) $r \subset U^n$ in such systems was shown [5], as well as the possibility of solving basic problems under incomplete

information, when for one reason or another only a part of the original pattern enters the system [6]. Under interference conditions, pattern comparison is performed with an accuracy that is specified using some metric $\rho(p, p')$. As a result, the formulation of the basic problems changes, which in this case must be solved for pattern regions $\Delta(p) = \{p' \in P \mid \rho(p, p') \le r\}$. In [7], the solution of basic problems for pattern regions is given, which are specified using the Hamming metric. Since the use of a metric is not always possible, in this paper we consider a more general case when pattern regions are a direct product of the ranges for each of the coordinates. For these pattern regions, the solution of basic problems is given and the possibility of implementing a logical inference with variables is shown.

The next section describes the formulation of basic problems for pattern regions. The solution of these basic problems with the help of a general universal method based on element-by-element pattern comparison is considered. Then the solution of basic problems is given using the intersection method for the pattern regions in the form of a direct product of the ranges of values. Finally, in conclusion, it is shown that these pattern regions serve as the basis for implementation a logical inference with variables.

2. Basic problems for pattern regions

The need for regions of patterns arises, for example, in the presence of interference. In this case, the comparison of patterns is performed with an accuracy that is specified using some metric $\rho(p', p)$. It is believed that if for an unknown pattern p and pattern p_i by name i is $\rho(p, p_i) \le r$, then an unknown pattern p – this is pattern by name i (with precision $\rho(p, p_i) \le r$). In this regard, the concept of a new pattern is changing. As a new pattern is considered only that pattern p, for which there is no pattern p_i by the name i such that $\rho(p, p_i) \le r$.

As a result, if the system has memorized some pattern p_i named i, then in fact the name i is given to all patterns of the pattern region $\Delta_i = \{p \in P \mid \rho(p, p_i) \le r\}$. The name i can be interpreted as a region Δ_i name. The corresponding name mapping $\varphi_\Delta : i \to \Delta_i$ is equal to $\varphi_\Delta = \phi_\Delta \circ \varphi_p$, where $\phi_\Delta : p_i \to \Delta_i$ – is a one-to-one correspondence between the pattern p_i and the region Δ_i (for a given r), $\varphi_p : i \to p_i$ – is the name mapping for patterns p, composition of mappings $(f_1 \circ f_2)(x) = f_1(f_2(x))$.

Based on the foregoing, the basic problems for pattern regions are formulated as follows.

In order to solve the direct problem for any pattern p, you need to get the names of all known pattern regions Δ such that $p \in \Delta$. If such regions of patterns do not exist, then it is necessary to create and memorize under some name $i \in U_\Delta$ such an pattern region Δ_i, that $p \in \Delta_i$. Here U_Δ – name domain for naming pattern regions.

In order to solve the inverse problem for any name $i \in U_\Delta$, it is necessary to get the pattern region Δ by name i. If the name i is pure and has not yet been used to name pattern regions, an appropriate conclusion must be made.

3. Solving basic problems for pattern regions using the method of element-by-element pattern comparison

A general universal method for solving basic problems is a method based on element-by-element pattern comparison. It is very simple and with it you can solve basic problems for any type of patterns, for which element-by-element comparison generally makes sense. Under noise conditions, element-by-element comparison of patterns is performed with a certain specified accuracy r, where r – is an integer, $r > 0$.

Further, patterns will be considered in the form of finite sequences or vectors $p = (i_1, ..., i_m)$, $1 \le m \le n$, which belong to the set of patterns

$$P = \bigcup_{m=1}^{n} P^m, \quad P^m = U_1 \times ... \times U_m,$$

where U_k – name domain of the k-th coordinate. The dimension of a sequence or vector $p = (i_1, ..., i_m)$ will be denoted by m_p

, i.e. $m_p = m$. For naming pattern $p \in P$ and corresponding pattern regions $\Delta = \phi(p)$ name domain U_Δ will be used.

Element-by-element, or more precisely, coordinate-wise comparison of patterns $p = (i_1, ..., i_m) \in P^m$ and $p' = (i_1', ..., i_m') \in P^m$ with accuracy $r > 0$ leads to regions of patterns

$\Delta_\infty(p, r) = \{p' \in P^m \mid \rho_\infty(p, p') \le r\}$,

which are set using the metric $\rho_\infty(p, p') = \max_k |i_k - i_k'|$, where $m_p = m$.

Indeed, for any pattern $p' = (i_1', ..., i_m') \in \Delta_\infty(p, r)$ is satisfied $|i_k - i_k'| \le r$, $k = 1, ..., m$. Conversely, if for any $k = 1, ..., m$ is satisfied $|i_k - i_k'| \le r$, then the pattern $p' \in \Delta_\infty(p, r)$. Therefore, $p' = (i_1', ..., i_m') \in \Delta_\infty(p, r)$ if and only if, when $|i_k - i_k'| \le r$, $k = 1, ..., m$.

The scheme for solving basic problems for pattern regions $\Delta_\infty(p, r)$ using coordinate-wise comparison with accuracy r has the following form.

One index A_Δ is used which consists of columns $(i \mid p_i)$, where p_i – pattern for pattern region $\Delta_\infty(p_i, r)$, is known by the name i.

Initially $A_\Delta = \varnothing$, where $\varnothing$ – is the empty set.

Direct problem solution. Let any pattern $p \in P$ be given, for which it is necessary to solve the direct problem. The pattern p is coordinate-by-coordinate compared (with accuracy r) with the patterns a_i of all columns $(i \mid a_i) \in A_\Delta$. Thanks to this, for all known pattern regions $\Delta_\infty(a_i, r)$ membership $p \in \Delta_\infty(a_i, r)$ is checked and a set of names $S_\Delta = \{i \in U_\Delta \mid p \in \Delta_\infty(a_i, r)\}$ is formed.

If the set of names $S_\Delta \ne \varnothing$, then it is a solution to the direct problem, since it contains the names of all known pattern regions $\Delta_\infty(p_i, r)$ such that $p \in \Delta_\infty(p_i, r)$.

If $S_\Delta = \varnothing$, then such pattern regions do not exist. The pattern p is new and needs to be memorized. The column engine chooses any pure name $i_\Delta \in U_\Delta$ for it and performs the addition $A_\Delta + (i_\Delta \mid p)$, i.e. to the index A_Δ a column $(i_\Delta \mid p)$ is added. Memorizing an pattern p means creating a new pattern region $\Delta_\infty(p, r)$ named i_Δ. If in the future for any pattern $p' \in \Delta_\infty(p, r)$ will be solved the direct problem, then the set of names S_Δ will be contained at least one name i_Δ.

Solution of the inverse problem. To solve the inverse problem for any name $i \in U_\Delta$, it suffices to take the pattern a_i of the column $(i \mid a_i) \in A_\Delta$. The desired pattern region is equal to $\Delta_\infty(a_i, r)$. If in the index A_Δ there is no column named i, then i – is a pure name.

4. Solving basic problems for pattern regions using the intersection method

As was shown, a coordinate wise pattern comparison with accuracy r leads to pattern regions $\Delta_\infty(p, r)$. Moreover, any pattern $p' \in \Delta_\infty(p, r) \subset P^m$ if and only if when $|i_k - i_k'| \le r$, $k = 1, ..., m$, $m_p = m$. Therefore, the pattern region $\Delta_\infty(p, r)$ can be represented as a direct product

$\Delta_\infty(p, r) = \delta_1 \times ... \times \delta_m$,

where $\delta_k = \{i_k' \in U_k \mid |i_k - i_k'| \le r\}$, $k = 1, ..., m$. Set of names $\delta_k \subset U_k$ is named the range of values of k-th coordinate.

Further, the general case will be considered, when any range of values δ_k is an arbitrary non-empty name set $\delta_k \subset U_k$. This

is due to the fact that the use of metrics to define pattern regions is possible, as a rule, only at those levels of the system that are directly related to the receipt and processing of quantitative information. For example, these can be patterns containing the parameters of the object's movement. At higher levels of the intelligent system, each pattern can contain extremely heterogeneous information, both quantitative and qualitative. And here the assignment of pattern regions with the help of a metric either loses its meaning, or even becomes impossible.

So, for patterns in the form of finite sequences or vectors $p = (i_1, ..., i_m) \in P$, $1 \le m \le n$, the pattern regions in the form of a direct product $\Delta = \delta_1 \times ... \times \delta_m$ will be considered, where $\delta_k \neq \varnothing$ – range of values of k-th coordinate, $\delta_k \subset U_k$, U_k – name domain of the k-th coordinate.

Ranges of values and the membership problem

Any range of values δ_k is an pattern in the form of an unordered name set. Therefore, the direct and inverse problems are formulated and solved for it in the usual way. In order to solve the direct problem for the range of values δ_k, you need to obtain its name. In order to solve the inverse problem for any name $i_{\delta k} \in U_{\delta k}$, it is necessary to obtain the range of values δ_k, known by name $i_{\delta k}$. Here $U_{\delta k}$ – name domain for naming ranges of values $\delta_k \subset U_k$.

However, in order to solve the basic problems for the pattern regions $\Delta = \delta_1 \times ... \times \delta_m$, we need another inverse problem, which is called the membership problem.

In order to solve the membership problem for any name $i_k \in U_k$ it is necessary to obtain the names of all known ranges of values $\delta_k \subset U_k$ such that $i_k \in \delta_k$.

Let us first consider the simplest case, when the ranges of values are specified in advance and an index $A_\delta = \{A_{\delta 1}, ..., A_{\delta n}\}$ for the membership problem and an index $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\}$ for the inverse problem for the ranges δ_k are formed.

For each name $i_k \in U_k$ to the index $A_{\delta k}$ a column $(i_k \mid a_{\delta k})$ is added, the pattern of which $a_{\delta k} \neq \varnothing$ contains the names of all ranges $\delta_k \subset U_k$ such that $i_k \in \delta_k$. Obviously, in order to solve the membership problem for any name $i_k \in U_k$, it suffices to take the pattern $a_{\delta k}$ of the column $(i_k \mid a_{\delta k}) \in A_{\delta k}$.

The index $B_{\delta k}$ consists of columns $(i_{\delta k} \mid b_k)$, where b_k – is the range of values $b_k \subset U_k$ by name $i_{\delta k}$. Let any name be given $i_{\delta k} \in U_{\delta k}$, for which it is necessary to solve the inverse problem for the ranges of values. The range of values δ_k by name $i_{\delta k}$ is equal to the pattern b_k of the column $(i_{\delta k} \mid b_k) \in B_{\delta k}$. If in the index $B_{\delta k}$ there is no column named $i_{\delta k}$, then $i_{\delta k}$ – pure name.

Here you can clearly see the differences between the two inverse problems. The inverse problem for ranges of values is the transition $i_{\delta k} \to \delta_k$ from the name of the range of values to the range of values. The membership problem is also an inverse problem. But when solving it, a transition $i_k \to a_{\delta k}$ is made from the name $i_k \in U_k$ to the pattern $a_{\delta k}$, where the pattern $a_{\delta k}$ contains the names of all ranges of values $\delta_k \subset U_k$ such that $i_k \in \delta_k$.

In the general case, ranges of values of a given type are created during system operation. In this case, the membership problem is formulated as follows.

In order to solve the membership problem for any name $i_k \in U_k$ it is necessary to obtain the names of all known ranges of values $\delta_k \subset U_k$ such that $i_k \in \delta_k$. If such ranges do not exist, it is necessary to create and memorize under some name $i_{\delta k} \in U_{\delta k}$ such a range of values δ_k, that $i_k \in \delta_k$. The name $i_{\delta k} \in U_{\delta k}$ is in this case the solution to the membership problem.

Memorizing a new range of values under a specific name means that a direct problem must be solved for it. As a result, each stored range of values δ_k will have a single unique name $i_{\delta k} \in U_{\delta k}$. In this case, when solving basic problems for ranges of values, the intersection method is most convenient [7]. First, the basis of the intersection method is that the pattern $a_{\delta k}$ of any

column $(i_k \mid a_{\delta k}) \in A_{\delta k}$ contains the names of all memorized value ranges δ_k such that $i_k \in \delta_k$. That is, the index of the direct problem for ranges simultaneously provides a solution to the membership problem. Secondly, any new range of values δ_k is indeed new, since by construction it differs from all known value ranges of the k-th coordinate. Therefore, there is no need to check its novelty and calculate the intersection $\eta(A_{\delta k}, \delta_k)_{\{i\}}$ [7].

To solve the membership problem using intersection method we will use: indices $A_\delta = \{A_{\delta 1}, ..., A_{\delta n}\}$ and $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\}$, where $A_{\delta k}$ and $B_{\delta k}$ – are the direct and inverse problem indices for the ranges of the k-th coordinate, $k = 1, ..., n$.

The scheme for solving the membership problem has the following form.

In the initial state $A_\delta = \{A_{\delta 1}, ..., A_{\delta n}\} = \varnothing$, $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\} = \varnothing$.

Let for any name $i_k \in U_k$ it is necessary to solve the membership problem. For this, a pattern $a_{\delta k}$ of the column $(i_k \mid a_{\delta k}) \in A_{\delta k}$ is taken. The pattern $a_{\delta k}$ is a solution to the membership problem since it contains the names of all known ranges of values $\delta_k \subset U_k$ such that $i_k \in \delta_k$.

If in the index $A_{\delta k}$ there is no column named i_k, this means that there are no ranges of values $\delta_k \subset U_k$ such that $i_k \in \delta_k$. The column engine creates a new value range $\delta_k \subset U_k$ of the given type, for which $i_k \in \delta_k$. Moreover, this will be a really new range of values, since this is the only range of values that contains the name i_k. To memorize the new range δ_k, the column engine chooses any pure name $i_{\delta k} \in U_{\delta k}$ for it and performs additions [7]:

$$A_{\delta k} + (\delta_k \mid \{i_{\delta k}\}) = A_{\delta k} + \sum_{i_k \in \delta_k} (i_k \mid \{i_{\delta k}\}),$$

$$B_{\delta k} + (i_{\delta k} \mid \delta_k),$$

i.e. to the index $A_{\delta k}$ are added $|\delta_k|$ columns $(i_k \mid \{i_{\delta k}\})$, and to the index $B_{\delta k}$ a column $(i_{\delta k} \mid \delta_k)$ is added, where $|S|$ – number of elements (cardinality) of the set S.

The name $i_{\delta k}$ is a solution to the direct problem and the membership problem.

If in the future other new value ranges δ_k are memorized that contain the name i_k, then, due to factorization by name, a pattern $a_{\delta k}$ of the column $(i_k \mid a_{\delta k}) \in A_{\delta k}$ the names of these value ranges δ_k will also be added. Thus, for any name $i_k \in U_k$ the solution to the membership problem $a_{\delta k} \neq \varnothing$ and $|a_{\delta k}| \geq 1$.

The inverse problem for ranges of values is solved in the usual way. Let any name $i_{\delta k} \in U_{\delta k}$ be given. Range of values δ_k by name $i_{\delta k}$ is equal to the pattern b_k of the column $(i_{\delta k} \mid b_k) \in B_{\delta k}$. If in the index $B_{\delta k}$ there is no column named $i_{\delta k}$, then $i_{\delta k}$ – is a pure name.

Basic problems for pattern regions in the form of a direct product of value ranges

The formulation of the basic problems for the considered pattern regions remains unchanged.

In order to solve the direct problem for any pattern $p = (i_1, ..., i_m) \in P$, $1 \leq m \leq n$, it is necessary to obtain the names of all known pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ such that $p \in \Delta$.

In order to solve the inverse problem for any name $i \in U_\Delta$, it is necessary to obtain the pattern region Δ by name i. If the name i is a pure name, then an appropriate conclusion must be made.

Consider any pattern $p = (i_1, ..., i_m) \in P$, $1 \leq m \leq n$. By solving the membership problem for all coordinates of the pattern p, you can get a pattern set $P_\Delta(p) = a_{\delta 1} \times ... \times a_{\delta m}$, where $a_{\delta k} \neq \varnothing$ – is the solution of the membership problem for the name i_k, which is the value of the k-th coordinate of the pattern p.

It is easy to show that any pattern $p_\Delta \in P_\Delta(p)$ corresponds one-to-one to such a pattern region Δ, that $p \in \Delta$.

Indeed, let $p_\Delta = (i_{\delta 1}, ..., i_{\delta m}) \in P_\Delta(p)$, where $i_{\delta k} \in a_{\delta k}$, $k = 1, ..., m$. Obviously, the pattern p_Δ corresponds one-to-one to the pattern region $\Delta = \delta_1 \times ... \times \delta_m$, where δ_k – is the range of values by name $i_{\delta k}$. In this case, the name i_k, which is the k-th coordinate of the pattern $p = (i_1, ..., i_m)$, belongs to the value range δ_k by name $i_{\delta k}$, i.e. $p \in \Delta = \delta_1 \times ... \times \delta_m$

The name of pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m})$ can be interpreted as the name of an pattern region $\Delta = \delta_1 \times ... \times \delta_m$. The corresponding name mapping $\varphi_\Delta : i \to \Delta_i$ equals $\varphi_\Delta = \phi \circ \varphi_{p\Delta}$, where $\phi : p_\Delta \to \Delta$ – is a one-to-one correspondence between the pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m})$ and the pattern region $\Delta = \delta_1 \times ... \times \delta_m$, $\varphi_{p\Delta} : i \to p_{\Delta i}$ – is the name mapping for patterns p_Δ, $i \in U_\Delta$, U_Δ – is the name domain for patterns p_Δ and pattern regions $\Delta = \phi(p_\Delta)$.

Therefore, to find the name of the pattern region $\Delta = \phi(p_\Delta)$, it is sufficient to solve the direct problem for the pattern p_Δ.

To solve basic problems for patterns p_Δ using the intersection method, we will use: an index $A_\Delta = \{A_{\Delta 1}, ..., A_{\Delta n}\}$ for the direct problem, an index B_Δ for the inverse problem, as well as given by the set of ordered pairs (i, m_i) function $m_\Delta(i)$, which will store the dimensions of known patterns p_Δ [7].

In addition, name set $S_\Delta \subset U_\Delta$ will be used.

The scheme for solving basic problems for pattern regions $\Delta = \delta_1 \times ... \times \delta_m$, $1 \le m \le n$, has the form.

Initially $A_\Delta = \varnothing$, $B_\Delta = \varnothing$ and $m_\Delta(i) = \varnothing$.

Direct problem solution. Let an pattern be given $p = (i_1, ..., i_m) \in P$, $1 \le m \le n$, for which it is necessary to solve the direct problem for pattern regions.

For a pattern p is set $S_\Delta = \varnothing$.

For all names i_k, that are pattern coordinates $p = (i_1, ..., i_m)$, the membership problem is solved and a set of patterns $P_\Delta(p) = a_{\delta 1} \times ... \times a_{\delta m}$ is formed, where $a_{\delta k}$ – is the solution of the membership problem for the name i_k. Since $a_{\delta k} \neq \varnothing$ for any k, then the set of patterns $P_\Delta(p) \neq \varnothing$. Moreover, if exists k, for which $|a_{\delta k}| > 1$, then $|P_\Delta(p)| > 1$. This means that the pattern p belongs to several pattern regions at once.

Each pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m}) \in P_\Delta(p)$ corresponds to a pattern region $\Delta = \phi(p_\Delta)$, that $p \in \Delta$. To find the name of this pattern region, the direct problem for the pattern p_Δ is solved.

For the pattern p_Δ the intersection is calculated

$$\eta(A_\Delta, p_\Delta)_{(i)} = \bigcap_{k=1}^{m} a_{\Delta k},$$

where $a_{\Delta k}$ – is the pattern of the column $(i_{\delta k} \mid a_{\Delta k}) \in A_{\Delta k}$, $i_{\delta k}$ – name that is the k-th coordinate of the pattern p_Δ [7].

If $\eta(A_\Delta, p_\Delta)_{(i)} \neq \varnothing$ and there is at least one name $i \in \eta(A_\Delta, p_\Delta)_{(i)}$ such that $m_\Delta(i) = m$, then i – the only name that is the name of the pattern p_Δ and the corresponding pattern region Δ such that $p \in \Delta$. The name i is added to name set S_Δ.

In all other cases, the pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m})$ is new and must be memorized. The corresponding region of patterns $\Delta = \delta_1 \times ... \times \delta_m$ is also new, where δ_k – is the range of values by name $i_{\delta k}$. The column engine chooses for the pattern p_Δ any pure name $i_\Delta \in U_\Delta$ and performs additions [7]:

$A_\Delta + (p_\Delta \mid \{i_\Delta\}) = \{A_{\Delta 1} + (i_{\delta 1} \mid \{i_\Delta\}), ..., A_{\Delta m} + (i_{\delta m} \mid \{i_\Delta\})\}$,

$B_\Delta + (i_\Delta \mid p_\Delta)$,

$m_\Delta(i) \bigcup (i_\Delta, m)$.

The name i_Δ is the solution to the direct problem and is the name of the pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m})$ and the corresponding pattern region $\Delta = \delta_1 \times ... \times \delta_m$ such that $p \in \Delta$. The name i_Δ is added to the name set S_Δ.

After the direct problem has been solved for all patterns $p_\Delta = (i_{\delta 1}, ..., i_{\delta m}) \in P_\Delta(p)$, a set of names $S_\Delta \neq \varnothing$ will be obtained. The set of names S_Δ is the solution to the direct problem for pattern regions, since it contains the names of all known pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ such that pattern $p = (i_1, ..., i_m) \in \Delta$.

Solution of the inverse problem. To solve the inverse problem for any name $i \in U_\Delta$, is taken a pattern b_i of the column $(i \mid b_i) \in B_\Delta$. If in the index B_Δ there is no column by name i, then i – is a pure name.

If in the index B_Δ is a column $(i \mid b_i)$, then $p_\Delta = (i_{\delta 1}, ..., i_{\delta m}) = b_i$. The desired pattern region is equal to $\Delta = \delta_1 \times ... \times \delta_m$, where δ_k – range of values by name $i_{\delta k}$. For ranges of values δ_k the inverse problem is solved. Range of values δ_k by the name $i_{\delta k}$ is equal to the pattern $b_{\delta k}$ of the column $(i_{bk} \mid b_{\delta k}) \in B_{\delta k}$, $k = 1, ..., m$, where $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\}$ – index for the inverse problem for value ranges.

EXAMPLE. Let $n = 3$ and for all name domains $|U_k| = 7$. For simplicity, we will assume that the value ranges are set in advance, and in each name domain U_k there are three ranges: range of values $\delta_1 = \{1, 2, 3\}$ by name 1, range of values $\delta_2 = \{3, 4, 5\}$ by name 2 and range of values $\delta_3 = \{5, 6, 7\}$ by name 3. Therefore, the indices $A_{\delta k}$ and $B_{\delta k}$ for ranges of values are the same for all k and have the following form:

$A_{\delta k}$

		2		3		
1	1	1	2	2	3	3
1	2	3	4	5	6	7

$B_{\delta k}$

3	5	7
2	4	6
1	3	5
1	2	3

Let also the indices A_Δ, B_Δ and the function $m_\Delta(i)$ have the form:

$A_{\Delta 1}$

		3
1		2
1	2	3

$A_{\Delta 2}$

3		
2		1
1	2	3

$A_{\Delta 3}$

	3	2
1	2	3

B_Δ

	3	2
3	1	1
1	3	3
1	2	3

$m_\Delta(i)$

i	1	2	3
m	2	3	3

It is easy to see that are remembered: under the name 1 pattern $p_{\Delta 1} = (1, 3)$ and pattern region $\Delta_1 = \delta_1 \times \delta_3$, under the name 2 pattern $p_{\Delta 2} = (3, 1, 3)$ and pattern region $\Delta_2 = \delta_3 \times \delta_1 \times \delta_3$, under the name 3 pattern $p_{\Delta 3} = (3, 1, 2)$ and pattern region $\Delta_3 = \delta_3 \times \delta_1 \times \delta_2$.

Let the direct problem for the pattern regions be solved for the pattern $p=(2,6)$. It solves the membership problem. For $i_1=2$ pattern $a_{\delta 2}$ of the column $(2\,|\,a_{\delta 2})\in A_{\delta k}$ is equal $a_{\delta 2}=\{1\}$, i.e. coordinate $i_1=2$ of the pattern p belongs to range of values by name 1. Similarly, for $i_2=6$ pattern $a_{\delta 6}$ of the column $(6\,|\,a_{\delta 6})\in A_{\delta k}$ is equal $a_{\delta 6}=\{3\}$, i.e. coordinate $i_2=6$ of the pattern p belongs to the range of values by name 3. Therefore, the set of patterns $P_\Delta=\{1\}\times\{3\}$ consists of one pattern $p_\Delta=(1,3)$. For the pattern $p_\Delta=(1,3)$ the direct problem is solved. The intersection $\eta(A_\Delta,\,p_\Delta)_{(i)}=\{1\}$ and $m_\Delta(1)=2$, i.e. the pattern $p_\Delta=(1,3)$ – is the pattern by name 1. Therefore, the pattern $p=(2,6)$ belongs to the pattern region Δ, which is known by name 1.

If the pattern $p=(6,2,5)$, then the solution of the membership problem for all its coordinates will give a set of patterns $P_\Delta=\{3\}\times\{1\}\times\{2,3\}=\{p_{\Delta 1},\,p_{\Delta 2}\}$, where $p_{\Delta 1}=(3,1,2)$, $p_{\Delta 2}=(3,1,3)$. Therefore pattern $p=(6,2,5)$ belongs to two pattern regions at once. Solving for patterns $p_{\Delta 1}$ and $p_{\Delta 2}$ direct problem, we get $\eta(A_\Delta,\,p_{\Delta 1})_{(i)}=\{3\}$, $m_\Delta(3)=3$ and $\eta(A_\Delta,\,p_{\Delta 2})_{(i)}=\{2\}$, $m_\Delta(2)=3$. This means that the pattern $p=(6,2,5)$ belongs to pattern regions by name 3 and 2.

Let $p=(7,1)$. For it the pattern $p_\Delta=(3,1)$, the intersection $\eta(A_\Delta,\,p_\Delta)_{(i)}=\{2,3\}$, but $m_\Delta(2)=m_\Delta(3)\neq 2$. Therefore, the pattern $p_\Delta=(3,1)$ and the corresponding region $\Delta=\delta_3\times\delta_1$ are new. It needs to be memorized. The column engine chooses a pure name 4 for them and performs additions $A_\Delta+(p_\Delta\,|\,\{4\})$, $B_\Delta+(4\,|\,p_\Delta)$, $m_\Delta(i)\bigcup(4,2)$. As a result, we get:

$A_{\Delta 1}$

		4	
		3	
1		2	
1	2	3	

$A_{\Delta 2}$

4			
3			
2			1
1	2	3	

$A_{\Delta 3}$

		3	2
1	2	3	

B_Δ

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

$m_\Delta(i)$

i	1	2	3	2
m	2	3	3	4

If for the pattern $p=(7,1)$ again solve the direct problem for the pattern regions, then again we get $p_\Delta=(3,1)$. The intersection $\eta(A_\Delta,\,p_\Delta)_{(i)}=\{2,3,4\}$ and $m_\Delta(4)=2$. This means that the pattern $p=(7,1)$ belongs to the pattern region Δ named 4.

Suppose now it is necessary to solve the inverse problem for the pattern region Δ by name 3. For it, the pattern p_Δ is equal to the pattern b_3 of the column $(3\,|\,b_3)\in B_\Delta$, i.e. $p_\Delta=(3,1,2)$. It follows that the pattern region Δ is a direct product $\Delta=\delta_3\times\delta_1\times\delta_2$ of the value ranges named 3, 1 and 2. Solving using the index $B_{\delta k}$ the inverse problem, we obtain $\delta_3=\{5,6,7\}$, $\delta_1=\{1,2,3\}$ and $\delta_2=\{3,4,5\}$.

5. Logical inference

Initially, pattern regions $\Delta=\delta_1\times\ldots\times\delta_m$ arose as a natural tool that provides the possibility of solving problems in the presence of interference. However, it immediately turned out that the presence of pattern regions means the presence in the system of local classification and local generalization. In the first case, this makes it possible for the system to solve complex classification

problems. In the second, it provides the ability to form multi-level structures with more and more common names (concepts) when building invariant representations. The pattern regions play the most important role in the implementation of inference with variables.

In intelligent systems with forward-chaining (data-directed inference), reasoning proceeds from data to possible outcomes. An inference engine uses facts and knowledge base rules to generate new facts. Usually, IF-THEN rules are used, which consist of a left and a right part. The left side or prerequisite describes the conditions under which the rule can be executed. The right part of the rule describes the actions that are performed when the rule fires.

Consider now an columns-based intelligent system. In it, the transition from the pattern to its name $p \to i_p$ can be seen as a step of the data-directed inference. In this case, the names included in the pattern play the role of conditions in the premise of the rule, and the name of the pattern i_p is the result of the execution of the rule.

Indeed, let there be an pattern $p = (i_1, i_2, ..., i_m) \in P$ by name i_p. If for the pattern p solve the direct problem, then its name i_p will be obtained, i.e. if there is an pattern, the value of the first coordinate of which is equal to the name i_1, and the value of the second coordinate of which is equal to the name i_2,… and the value of the m-th coordinate of which is equal to the name i_m, then the name i_p will be obtained. In other words, there is a rule

IF i_1 AND i_2 … AND i_m THEN i_p.

The role of the inference engine in this case is played by the column engine, which solves the direct problem $p \to i_p$ and thus performs the step of the data-directed inference.

In the given example, the conditions in the premise, represented by the elements of the known pattern p, are connected using a connective $\wedge$ (and). There is no connective $\vee$ (or), negation $\neg$ and variables. In order to be able to talk about the implementation in columns-based intelligent systems the rules of the data-directed inference, it is necessary to show the possibility of implementing arbitrary conditions expressed using connectives $\wedge$, $\vee$, $\neg$ and variables.

Variables with a universal quantifier $\forall$ will be considered, which are quite often used in rule-based systems. If the normal condition in the prerequisite is satisfied when the names $i_k' = i_k$, then the condition with such a variable is considered to be satisfied for $\forall i_k' \in \delta_k$. The range of values $\delta_k \subset U_k$ is called the range of variable, and the name $i_{\delta k}$ of the range of values δ_k is called the variable name. Thus, in fact, a variable is a named value range or column $(i_{\delta k} \mid \delta_k)$.

Let any pattern region $\Delta = \delta_1 \times \delta_2 \times ... \times \delta_m$ be given such that for at least one range of values $|\delta_k| > 1$, i.e. the pattern region Δ contains more than one pattern. Let us show that the pattern region Δ corresponds to the general rule that uses variables with the universal quantifier.

Again consider the pattern $p = (i_1, i_2, ..., i_m) \in P$ named i_p. As shown, it corresponds to the rule

IF i_1 AND i_2 … AND i_m THEN i_p

or in another entry

$i_1 \wedge i_2 \wedge ... \wedge i_m \to i_p$.

Obviously, for an arbitrary fact represented by the pattern $p' = (i_1', i_2', ..., i_m') \in P$, the conditional part of this rule is satisfied only if all m of conditions $i_k' = i_k$ are met, i.e. if $p' = p$.

Now let under the name i_Δ the system remember the pattern region

$\Delta = i_1 \times \delta_2 \times i_3 \times ... \times i_m$,

where δ_2 – value range by name $i_{\delta 2}$, $|\delta_2| > 1$.

When solving the direct problem, the name i_{Δ} will be obtained for any pattern $p' \in \Delta$, i.e. if $i'_k = i_k$ for $k \neq 2$ and $\forall i'_2 \in \delta_2$, then i_{Δ}. The corresponding IF-THEN rule can be written as

$$i_1 \wedge i_{\delta 2} \wedge ... \wedge i_m \to i_{\Delta},$$

where $i_{\delta 2}$ – is a name of variable with a range of variable δ_2. In other words, the premise of this rule uses a variable by name $i_{\delta 2}$ with a universal quantifier in the range of variable δ_2, i.e. the second condition is met when substituted for the variable name $i_{\delta 2}$ of any name $i'_2 \in \delta_2$.

In this case, we have a general rule formed using a variable named $i_{\delta 2}$. The premise of this rule is the pattern region $\Delta = i_1 \times \delta_2 \times i_3 \times ... \times i_m$. Moreover, if the premise in the form of the pattern p uses the connective $\wedge$, then the premise in the form of an pattern region also uses the connective $\vee$.

Indeed, let the range of the variable $\delta_2 = \{i_{21}, ..., i_{2l}\} \subset U_2$. This means that the second condition of the prerequisite is satisfied if i'_2 is $i_{\delta 1}$ or $i_{\delta 2}$ … or $i_{\delta l}$. Therefore, the rule under consideration can be rewritten in the form

$$i_1 \wedge i_{\delta 2} \wedge ... \wedge i_m \to i_{\Delta} \;\Rightarrow\; i_1 \wedge (i_{21} \vee ... \vee i_{2l}) \wedge ... \wedge i_m \to i_{\Delta}$$

Obviously it replaces l rules:

$$i_1 \wedge i_{21} \wedge ... \wedge i_m \to i_{\Delta},$$

…

$$i_1 \wedge i_{2l} \wedge ... \wedge i_m \to i_{\Delta},$$

i.e. is a general rule.

The range of a variable can be specified in the same way as any other range of values. Complex ranges can be obtained using Boolean subset algebra operations:

$$\delta_{k1} \wedge \delta_{k2} = \delta_{k1} \bigcap \delta_{k2},$$

$$\delta_{k1} \vee \delta_{k2} = \delta_{k1} \bigcup \delta_{k2},$$

$$\neg \delta_{k1} = U_k \setminus \delta_{k1},$$

where $\delta_{k1}, \delta_{k2} \subset U_k$, complement $U_k \setminus \delta_{k1}$ is the set of names that are in U_k but not in δ_{k1}.

Of greatest interest to us is negation, since it allows us to introduce negation into the conditions of the rule.

Let there be some variable named $i_{\delta k}$ with a range of variable δ_k, i.e. column $(i_{\delta k} \mid \delta_k)$, where $i_{\delta k} \in U_{\delta k}$, $\delta_k \subset U_k$.

The negation of a variable is understood as a variable with a range of variable $U_k \setminus \delta_k$. By denoting the name of this variable $\neg i_{\delta k} \in U_{\delta k}$, we get

$$\neg(i_{\delta k} \mid \delta_k) = (\neg i_{\delta k} \mid U_k \setminus \delta_k),$$

or

$$\neg(i_{\delta k} \mid \delta_k) = (\neg i_{\delta k} \mid \neg \delta_k),$$

because $U_k \setminus \delta_k = \neg \delta_k$.

For the correct definition of the negation operation, it is necessary that the variable name satisfies the equality $\neg\neg i_{\delta k} = \neg(\neg i_{\delta k}) = i_{\delta k}$. Then $\neg\neg i_{\delta k} = i_{\delta k}$, $\neg\neg \delta_k = \delta_k$ and $\neg\neg(i_{\delta k} \mid \delta_k) = (i_{\delta k} \mid \delta_k)$.

There are various options for defining the name $\neg i_{\delta k}$ so that the equality $\neg\neg i_{\delta k} = i_{\delta k}$ carried out. In the simplest version, into

the name domain $U_{\delta k}$ paired negative names (negative integers) are added. Then for $\forall i_{\delta k} \in U_{\delta k}$ negation can be defined as $\neg i_{\delta k} = -i_{\delta k}$. As a result $\neg\neg i_{\delta k} = -(-i_{\delta k}) = i_{\delta k}$.

Obviously, negation applies not only to variables, but also to any elements of the premise. So, in the premise $\Delta = \delta_1 \times \delta_2 \times ... \times i_k \times ... \times \delta_m$ the negation of a name i_k is a variable by name $\neg i_k \in U_{\delta k}$ with a range of variable $\delta_k = U_k \setminus \{i_k\}$.

In conclusion, it should be noted that as soon as an columns-based intelligent system is able to remember patterns under interference conditions, at the same time it is able to remember general rules and perform inference with variables.

6. Results

Pattern regions have emerged as a natural tool that provides a solution to problems under interference conditions. In this case, the pattern comparison is performed with an accuracy that is specified using a certain metric $\rho(p', p)$. It is believed that if for an unknown pattern p and pattern p_i named i is $\rho(p, p_i) \le r$, then the unknown pattern p – is an pattern by name i (with precision $\rho(p, p_i) \le r$). Therefore as a new pattern only that pattern p is considered, for which there is no pattern p_i names i such that $\rho(p, p_i) \le r$.

As a result, if the system remembers some pattern p_i named i, that's actually the name i get all patterns from pattern region $\Delta_i = \{p \in P \mid \rho(p, p_i) \le r\}$. Therefore the name i of the pattern p_i can be interpreted as a region Δ_i name.

Proceeding from this, the basic problems for the pattern regions are formulated.

In order to solve the direct problem for any pattern p, it is necessary to obtain the names of all known pattern regions Δ such that $p \in \Delta$. If such regions of patterns do not exist, then it is necessary to create and memorize under some name $i \in U_\Delta$ such an pattern region Δ_i, that $p \in \Delta_i$.

In order to solve the inverse problem for any name $i \in U_\Delta$, it is necessary to obtain the pattern region Δ by name i. If the name i is pure, an appropriate conclusion must be made.

A general universal method for solving basic problems is a method based on element-by-element pattern comparison. With its help, you can solve basic problems for any type of patterns for which element-by-element comparison makes sense. The paper considers patterns in the form of finite sequences or vectors $p = (i_1, ..., i_m) \in P$, $1 \le m \le n$.

Under conditions of interference, coordinate-wise comparison of patterns $p, p' \in P$ is performed with a certain specified accuracy r. Such a comparison means the use of pattern regions $\Delta_\infty(p, r) = \{p' \in P \mid \rho_\infty(p, p') \le r\}$, where the metric $\rho_\infty(p, p') = \max_k |i_k - i_k'|$.

The scheme for solving basic problems using the method based on coordinate-by-coordinate comparison with accuracy r differs only in that when comparing coordinates, equality $|i_k - i_k'| = 0$ is replaced by inequality $|i_k - i_k'| \le r$. In this case, the pattern region $\Delta_\infty(p, r)$ can be represented as a direct product $\Delta_\infty(p, r) = \delta_1 \times ... \times \delta_m$, where $\delta_k = \{i_k' \in U_k \mid |i_k - i_k'| \le r\}$ is the range of values of the k-th coordinate.

Further in the paper, we consider the general case when the pattern regions are a direct product $\Delta = \delta_1 \times ... \times \delta_m$, where the range of values δ_k is an arbitrary non-empty name set $\delta_k \subset U_k$, $k = 1, ..., m$.

A more efficient method for solving basic problems is the intersection method. To use it for pattern regions $\Delta = \delta_1 \times ... \times \delta_m$, it is necessary to solve a special inverse problem, which is called the membership problem. It is formulated as follows.

To solve the membership problem for any name $i_k \in U_k$, you must specify the names of all known value ranges $\delta_k \subset U_k$ such

that $i_k \in \delta_k$.

In the simplest case, the ranges are predefined. An index $A_\delta = \{A_{\delta 1}, ..., A_{\delta n}\}$ is created, such that for $\forall i_k \in U_k$ pattern $a_{\delta k}$ of the column $(i_k \mid a_{\delta k}) \in A_{\delta k}$ contains the names of all ranges $\delta_k \subset U_k$ such that $i_k \in \delta_k$.

If ranges of a given type are created during system operation, then a simplified intersection method is used for the membership problem. The solution scheme has the form.

In the initial state $A_\delta = \{A_{\delta 1}, ..., A_{\delta n}\} = \varnothing$, $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\} = \varnothing$.

Solution of the membership problem. For $\forall i_k \in U_k$ pattern $a_{\delta k}$ of the column $(i_k \mid a_{\delta k}) \in A_{\delta k}$ is a solution to the membership problem, since it contains the names of all known value ranges $\delta_k \subset U_k$ such that $i_k \in \delta_k$.

If in the index $A_{\delta k}$ there is no column by name i_k, a new range $\delta_k \subset U_k$ of the given type is created such that $i_k \in \delta_k$. The column engine chooses any pure name $i_{\delta k} \in U_{\delta k}$ for it and performs additions $A_{\delta k} + (\delta_k \mid \{i_{\delta k}\})$, $B_{\delta k} + (i_{\delta k} \mid \delta_k)$. The name $i_{\delta k}$ is a solution to the membership problem.

As a result, always $a_{\delta k} \neq \varnothing$ and $|a_{\delta k}| \geq 1$.

Solution of the inverse problem for value ranges. For $\forall i_{\delta k} \in U_{\delta k}$ range of values by name $i_{\delta k}$ is equal to the pattern b_k of the column $(i_{\delta k} \mid b_k) \in B_{\delta k}$. If in the index $B_{\delta k}$ there is no column by name $i_{\delta k}$, then $i_{\delta k}$ – is a pure name.

Now let any pattern $p = (i_1, ..., i_m) \in P$ be given. For all coordinates i_k of the pattern p you can solve the membership problem and get a set of patterns $P_\Delta(p) = a_{\delta 1} \times ... \times a_{\delta m}$, $|P_\Delta(p)| \geq 1$, where $a_{\delta k}$ is the solution of the membership problem for the name i_k. It is shown that there is a one-to-one correspondence $\phi : p_\Delta \to \Delta$, which to any pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m}) \in P_\Delta$ matches the pattern region $\Delta = \delta_1 \times ... \times \delta_m$ such that $p \in \Delta$. Therefore, the name of the pattern p_Δ can be interpreted as the name of the pattern region $\Delta = \phi(p_\Delta)$. Moreover, to solve the direct problem for the pattern region $\Delta = \phi(p_\Delta)$ such that $p \in \Delta$, it suffices to solve this problem for the pattern p_Δ.

The scheme for solving basic problems for pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ has the form.

Initially $A_\Delta = \{A_{\Delta 1}, ..., A_{\Delta n}\} = \varnothing$, $B_\Delta = \varnothing$ and $m_\Delta(i) = \varnothing$.

Direct problem solution. Let any pattern be given $p = (i_1, ..., i_m) \in P$. For all coordinates of the pattern p the membership problem is solved and a set of patterns $P_\Delta(p) = a_{\delta 1} \times ... \times a_{\delta m} \neq \varnothing$ is formed.

For every pattern $p_\Delta \in P_\Delta(p)$ the direct problem is solved. For this, the intersection $\eta(A_\Delta, p_\Delta)_{(i)}$ is calculated.

If $\eta(A_\Delta, p_\Delta)_{(i)} \neq \varnothing$ and there is at least one name $i \in \eta(A_\Delta, p_\Delta)_{(i)}$ such that $m_\Delta(i) = m$, then i – the only name that is the name of the pattern p_Δ and pattern region $\Delta = \phi(p_\Delta)$ such that $p \in \Delta$. The name i is added to the name set S_Δ.

In all other cases, the pattern p_Δ is new and needs to be memorized. The column engine chooses any pure name $i_\Delta \in U_\Delta$ for it and perform additions $A_\Delta + (p_\Delta \mid \{i_\Delta\})$, $B_\Delta + (i_\Delta \mid p_\Delta)$ and $m_\Delta(i) \bigcup (i_\Delta, m)$. The name i_Δ is the name of the pattern p_Δ and the corresponding pattern region $\Delta = \phi(p_\Delta)$ such that $p \in \Delta$. The name i_Δ is added to the name set S_Δ.

The set of names S_Δ, obtained after solving the direct problem for all patterns $p_\Delta \in P_\Delta(p)$, is the solution of the direct problem for pattern regions, since it contains the names of all known pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ such that the pattern $p = (i_1, ..., i_m) \in \Delta$.

Solution of the inverse problem. For $\forall i \in U_\Delta$ pattern p_Δ known by name i, is equal to the pattern b_i of the column $(i \mid b_i) \in B_\Delta$. If in the index B_Δ is no column by name i, then i – is a pure name.

Otherwise, the desired pattern region for the pattern $p_\Delta = (i_{\delta 1}, ..., i_{\delta m})$ is equal to $\Delta = \delta_1 \times ... \times \delta_m$, where δ_k – is the value range by name $i_{\delta k}$. The value range by name $i_{\delta k}$ is equal to the pattern $b_{\delta k}$ of the column $(i_{bk} \mid b_{\delta k}) \in B_{\delta k}$, where $B_\delta = \{B_{\delta 1}, ..., B_{\delta n}\}$ – index for the inverse problem for the value ranges.

Initially, pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ arose as a natural tool that provides the possibility of solving problems in the presence of interference. Later it turned out that they serve as the basis for implementation in columns-based intelligent systems of inference with variables.

In rule-based systems, the inference engine uses facts and IF-THEN rules to generate new facts. In an columns-based intelligent system, the transition from an pattern to its name $p \to i_p$ can be considered as a step of the data-directed inference. In this case, the names included in the pattern play the role of conditions in the premise of the rule, and the pattern name i_p is the result of the execution of the rule. The role of the inference engine is performed by the column engine, which solves the direct problem $p \to i_p$ and thus performs the step of the data-directed inference.

The pattern $p = (i_1, i_2, ..., i_m) \in P$ by name i_p corresponds to the IF-THEN rule $i_1 \wedge i_2 \wedge ... \wedge i_m \to i_p$. For an arbitrary fact represented by the pattern $p' = (i_1', i_2', ..., i_m') \in P$, the conditional part of this rule is satisfied if m of the conditions $i_k' = i_k$ are met.

Memorization of pattern regions $\Delta = \delta_1 \times ... \times \delta_m$ provides the possibility of implementing the rules under which connectives $\wedge$, $\vee$, $\neg$ and variables with the universal quantifier $\forall$ are used. Such a variable is understood as a range of values δ_k by name $i_{\delta k}$. If the usual condition in the prerequisite is met when the names $i_k' = i_k$, then the condition with the variable by name $i_{\delta k}$ is considered to be satisfied for $\forall i_k' \in \delta_k$.

It is shown that if there is any pattern region $\Delta = \delta_1 \times \delta_2 \times ... \times \delta_m$ by name i_Δ such that $|\Delta| > 1$, then this region corresponds to a general rule that uses variables with a universal quantifier.

So, for example, an pattern region $\Delta = i_1 \times \delta_2 \times i_3 \times ... \times i_m$ named i_Δ, where $\delta_2 = \{i_{21}, ..., i_{2l}\} \subset U_2$, matches the general rule $i_1 \wedge (i_{21} \vee ... \vee i_{2l}) \wedge ... \wedge i_m \to i_\Delta$, that replaces l rules $i_1 \wedge i_{2j} \wedge ... \wedge i_m \to i_\Delta$, $j = 1, ..., l$. The premise of this rule is satisfied for $i_k' = i_k$ ($k \neq 2$) and $\forall i_2' \in \delta_2$.

Complex ranges can be obtained using Boolean subset algebra operations: $\delta_{k1} \wedge \delta_{k2} = \delta_{k1} \bigcap \delta_{k2}$, $\delta_{k1} \vee \delta_{k2} = \delta_{k1} \bigcup \delta_{k2}$, $\neg \delta_{k1} = U_k \setminus \delta_{k1}$. The negation operator allows you to define the negation of a variable.

If there is a variable named $i_{\delta k}$ with a range of values δ_k, i.e. column $(i_{\delta k} \mid \delta_k)$, then the negation of a variable means a variable with a range of values $\neg \delta_k = U_k \setminus \delta_k$. If we designate the name of this variable $\neg i_{\delta k} \in U_{\delta k}$, then $\neg (i_{\delta k} \mid \delta_k) = (\neg i_{\delta k} \mid \neg \delta_k)$. For the correct definition of the negation operation, it is necessary that the variable name satisfies the equality $\neg \neg i_{\delta k} = \neg (\neg i_{\delta k}) = i_{\delta k}$. Then $\neg \neg i_{\delta k} = i_{\delta k}$, $\neg \neg \delta_k = \delta_k$ and $\neg \neg (i_{\delta k} \mid \delta_k) = (i_{\delta k} \mid \delta_k)$. In the simplest case, this can be achieved by using in $U_{\delta k}$ negative names (negative integers). Then negation $\neg i_{\delta k} = -i_{\delta k}$ and $\neg \neg i_{\delta k} = -(-i_{\delta k}) = i_{\delta k}$.

Memorizing patterns and pattern regions under certain names means memorizing the IF-THEN rules, in which conditions the connectives $\wedge$, $\vee$, $\neg$ and variables with the universal quantifier $\forall$ are used. Thus, inference is an intrinsic property inherent in columns-based intelligent systems.

7. Conclusion

Columns-based intelligent systems are systems that, despite their simplicity, have great potential. Arbitrary functions and

relations (predicates) can be implemented in them, they can solve problems of the classification type and work under conditions of incomplete information. Pattern regions arose as a natural tool that provides the possibility of solving problems in the presence of interference. At the same time, pattern regions provide the possibility of inference with variables, which once again demonstrates the versatility of columns-based intelligent systems.

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. Chesnokov A.M. Incomplete Information and Columns-Based Intelligent Systems // Turkish Journal of Computer and Mathematics Education, Vol.12 No.2 (2021), pp. 2771–2780.
7. Chesnokov A.M. Intersection Method – Mathematical Substantiation and Application in Columns-Based Intelligent Systems // Webology, 2022, Vol. 19, No. 1, pp.7516–7534.
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, p. 121.
10. Mikhailov A.M. Pattern Recognition by Indexing // Automation and Remote Control, 2012, Vol. 73, No. 4, pp. 717–724.
11. 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.