Advances in Science, Technology and Engineering Systems Journal
Vol. 5, No. 4, 686-699 (2020)
www.astesj.com
Special Issue on Multidisciplinary Innovation in Engineering Science & Technology

ASTES Journal
ISSN: 2415-6698

Fuzzy Recognition by Logic-Predicate Network

Tatiana Kosovskaya[*]

Department of Mathematics and Mechanics, St. Petersburg State University, St. Petersburg, 199034, Russia

ARTICLE INFO

Article history:
Received: 28 May, 2020
Accepted: 04 August, 2020
Online: 28 August, 2020

Keywords:
Predicate calculus
logic-predicate approach to AI
predicate network
fuzzy predicate network

ABSTRACT

The paper presents a description and justification of the correctness of fuzzy recognition by a logic-predicate network. Such a network is designed to recognize complex structured objects that can be described by predicate formulas. The NP-hardness of such an object recognition requires to separate the learning process, leaving it exponentially hard, and the recognition process itself. The learning process consists in extraction of groups of features (properties of elements of an object and the relations between these elements) that are common for objects of the same class. The main result of a paper is a reconstruction of a logic-predicate recognition cell. Such a reconstruction allows to recognize objects with descriptions not isomorphic to that from a training set and to calculate a degree of coincidence between the recognized object features and the features inherent to objects from the extracted group.

1 Introduction

This paper is an extension of work originally presented in conference CSIT-2019 [1].

The term "logical approach to solving Artificial Intelligence (AI) problems" is usually understood as data notation in the form of a binary string that defines the values of some object properties under study. In this case, if the object is represented as a set of its elements that have some properties and are connected by given relations, then when describing it as a binary string, firstly, the structure of the object itself is lost, and secondly, you have to store a large number of "unnecessary" information that the element a_j does not have the properties $p_{i_1}, \ldots, p_{i_k}$, that the elements a_{j_1}, a_{j_2} are not in the relations $q_{i_1}, \ldots, q_{i_l}$, etc.

Consider objects composed of smaller elements with preassigned properties, and some smaller object having preassigned relations. In such a case predicate formulas are an adequate description language.

The use of predicate calculus and automatic proof of a theorem for AI problems solving was offered by many authors [2]-[4] in the 70-th years of the XX century. At the same time the notion of NP-completeness was introduced and began to be actively used by the scientific community [5].

In 2003 the use of predicate calculus and automatic proof of a theorem for AI problems solving (without complexity bounds) was described in [6].

In particular, in that book it is shown that if a binary string simulates a description of an economic problem in which interaction agents have given properties and are in given relations then the notation length of such a string is exponential of the length of the description of the same problem input data by setting properties of these agents and relations between them, i.e. in fact, using the language of predicate calculus.

In 2007 the author of this paper has proved NP-completeness of recognition problems described in the terms of predicate formulas and upper bounds of number of steps for two solving algorithms [7]. When such problems are solved by an exhaustive search algorithm, their computational complexity coincides with the length of their input data encoding using a binary string [6].

A hierarchical level description of classes was suggested in [8] to decrease the computational complexity of these problems. Formulas which are isomorphic to "frequently appeared" sub-formulas of "small complexity" are extracted from class descriptions in order to construct such a level description. This allows to decompose the main problem into a series of similar problems with input data with the smaller length. An instrument for this extraction is partial deduction of a predicate formula [9],[10].

Later, the author noticed that the level description is actually a recognition network, which, after training (having exponential computational complexity), can quickly solve the recognition problem.

The concept of "network" is increasingly used both in theoretical and in practical research, particularly, when solving pattern recognition problems. Research and practical applications of neural networks [11],[12], Bayesian networks [13]-[15], technical networks [16] are widespread. The inputs of such networks are usually signals characterizing the properties of the studied objects or pro-

[*]Corresponding Author: Tatiana Kosovskaya, St. Petersburg State University, St. Petersburg, 199034, Russia, +7 921 323 23 07 & kosovtm@gmail.com

cesses, and expressed, as a rule, by numerical characteristics, which can be encoded by binary strings. The processing of binary strings in the most cases is carried out in linear or polynomial (usually quadratic) number of steps under the length of these strings. A convenient model for processing such strings is propositional logic (Boolean functions).

As noted above, the use of binary strings in the recognition of complex structured objects has disadvantages. To recognize such objects, the author offered the notion of a logic-predicate network using level description of classes [17]. Such a network contains two blocks: a training block and a recognition block. The training block run is based on the extraction from a class of objects description such fragments, which are inherent in many objects of the class. Recognition block run is reduced to the sequential solution of problems of the same type with a smaller length of the input data.

The disadvantage of such a recognition network is that it is able to recognize only objects that differ from those included in the training set (TS) by renaming the elements of the object (objects with isomorphic descriptions). But such a network can be retrained by adding unrecognized objects to the TS. This was the reason for modification of the logic-predicate network permitting to recognize objects, that not coincide but only similar to the objects from a training set. Such a modification was offered in [1, 18]. The degree of coincidence for description of an object part and the formula, satisfiability of which is checking in the cell, is calculated. Such a fuzzy network allows not only quickly enough to recognize objects isomorphic to those represented in the TS, but also to calculate the degree of coincidence that a "new" object belongs to one or another class. If necessary, it can be retrained using such a "new" object.

The structure of the paper is as follows.

Section 2 "Logic-predicate approach to AI problems" contains 2 subsections: "Problem Setting" and "Important Definitions". This section describes the results previously obtained by the author and defines the terminology previously introduced by the author, which are necessary for understanding the further presentation.

Section 3 "Level description" presents the basic idea of constructing a level description of classes. It contains two sub-sections "Construction of a level description of classes" and "The use of level description of a class". They give algorithms for constructing a level description of classes and the use of such a description for object recognition. These algorithms are further used in constructing the logic-predicate network. It is shown that the computational complexity of an object recognition decreases while using a level description of a class.

Section 4 "Logic-predicate network" describes the structure of such a network.

Section 5 "Example of logic-predicate network" describes an example illustrating the formation of a network, recognizing of an object with the use of a network, and its retraining. It consists of 4 sub-sections "Training Block Run: Extraction of Sub-Formulas", "Training Block Run: Forming a Level Description", "Recognition Block Run" and "Retraining the Network".

Section 6 "Fuzzy recognition by a logic-predicate network" describes the changes that need to be made to the logic-predicate network in order to recognize objects that are not isomorphic to those presented in the TS. A description of the contents of a fuzzy network cell is given. A cell of this type replaces each cell in the logic-predicate network in which the logical sequence of the target formula from the description of an object is checked. In this case, the degree of coincidence is calculated that the fragment being tested partially (and to what extent) satisfies the target formula.

Section 7 "Model example of fuzzy recognition" presents an example of fuzzy recognition of a "new" object.

In comparison with [1], the setting of problems that can be solved in the framework of logic-predicate approach are described in the presented paper in more details. A justification for the introduction of a level description is given. Algorithms for a level description construction and a level description use are described. Scheme 1 of a common up to the names of arguments sub-formulas extraction is added. A section 5 has been added with an example of constructing and modifying a logic-predicate network. It describes in details how a network is formed by the training set, a new object is recognized, and the network is rebuilt. The scheme of a fuzzy logic-predicate network cell is presented. An example of fuzzy recognition is described in more details.

2 Logic-Predicate Approach to AI Problems

A detailed presentation of the logic-predicate approach to solving AI problems is available in [19]. In this paper only a general setting of problems and main methods for their solving are formulated.

2.1 Problem Setting

As it was mentioned in the Introduction, if an object to be recognized is a complex structured one, then predicate formulas are a convenient language for its description and recognition. Let an investigated object ω be represented as a set of its elements $\omega = \{\omega_1, \ldots, \omega_t\}$ and be characterized by predicates $p_1, \ldots, p_n$ which define some properties of its elements or relations between them. The description $S(\omega_1, \ldots, \omega_t)$ of the object ω is a set of all constant literals (atomic formulas or their negations) with predicates $p_1, \ldots, p_n$ which are valid on ω.

Let the set of all investigated objects Ω be is divided into classes $\Omega_1, \ldots, \Omega_K$ such that $\Omega = \cup_{k=1}^{K} \Omega_k$. Logical description of the class Ω_k is such a formula $A_k(\overline{x})$[1] that if the formula $A_k(\overline{\omega})$ is true then $\omega \in \Omega_k$. The class description may be represented as a disjunction of elementary conjunctions of atomic formulas.

Many AI problems may be formulated as follows with the use of such descriptions.

Identification problem. To pick out all parts of the object ω which belong to the class Ω_k

$$S(\omega) \Rightarrow \exists \overline{x}_{k\neq} A_k(\overline{x}_k)^2. \tag{1}$$

Classification problem. To find all such class numbers k that $\omega \in \Omega_k$

$$S(\omega) \Rightarrow \vee_{k=1}^{K} A_k(\overline{x}_k). \tag{2}$$

[1]Here and below the notation $\overline{x}$ is used for an ordered list of the set x.

[2]To denote that there exist distinct values for variables from the list $\overline{x}$ the notation $\exists \overline{x}_{\neq} A_k(\overline{x})$ is used.

Analysis problem. To find and classify all parts τ of the object ω which may be classified

$$S(\omega) \Rightarrow \vee_{k=1}^{K} \exists \overline{x}_{k\neq} A_k(\overline{x}_k). \tag{3}$$

In fact, instead of the quantifier $\exists$ (*whether exists*) a symbol ? (*for what*) must be written. But if one uses a constructive method of its proof (an exhaustive search algorithm or logical methods such as derivation in a sequential predicate calculus or resolution method for predicate calculus) then not only existence of such arguments would be proved but the values of them would be found.

It was proved that the problems (1) and (3) are NP-complete [7] and the problem (2) is GI-complete [20], i.e. it is polynomially equivalent to the "open" problem of Graph Isomorphism, for which it is not proved its NP-completeness and a polynomial in time algorithm is not found.

As the formulas $A_k(\overline{x}_k)$ may be represented as a disjunction of elementary conjunctions of atomic formulas, the checking of each of the problems (1), (2) and (3) may be reduced to sequential checking of the formula

$$S(\omega) \Rightarrow \exists \overline{x}_{\neq} A(\overline{x}), \tag{4}$$

where $A(\overline{x})$ is an elementary conjunction. Note that in the case of the problem (2) the number of variables in $\overline{x}$ equals to the number of constants in ω.

The formula (4) may be checked, for example, by an exhaustive algorithm and an algorithm based on the derivation in sequential calculus or on the use of resolution method.

The upper bound of number of steps for an exhaustive algorithm is $O(t^m)$[3], where t is the number of elements in ω, m is the number of variables in $A(\overline{x})$ [7]. This upper bound coincides with the upper bound of a binary string simulating input data in the form of $S(\omega)$ and $A(\overline{x})$.

The upper bound of the number of steps for a logical algorithm is $O(s^a)$ where s is the maximal number of atomic formulas in $S(\omega)$ with the same predicate symbol having occurrences in $A(\overline{x})$, a is the number of atomic formulas in the elementary conjunction $A(\overline{x})$ [21].

Obviously, these estimates depend exponentially on the parameters of the formula $A(\overline{x})$. That's why it will be useful to break the solution of the problem into a series of sub-problems of the type (4) with a shorter right-hand side.

2.2 Important Definitions

The objects and notions satisfying the following definitions will be used later in the text.

Definition 1: Elementary conjunctions $P(a_1, \ldots, a_m)$ and $Q(b_1, \ldots, b_m)$ are called **isomorphic**

$$P(a_1, \ldots, a_m) \sim Q(b_1, \ldots, b_m),$$

if there are an elementary conjunction $R(x_1, \ldots, x_m)$ and substitutions of arguments $a_{i_1}, \ldots, a_{i_m}$ and $b_{j_1}, \ldots, b_{j_m}$ instead of the variables $x_1, \ldots, x_m$ such that the results of these substitutions $R(a_{i_1}, \ldots, a_{i_m})$ and $R(b_{j_1}, \ldots, b_{j_m})$ coincide with formulas $P(a_1, \ldots, a_m)$ and $Q(b_1, \ldots, b_m)$, respectively, up to the order of literals.

The substitutions $(a_{i_1} \to x_1, \ldots, a_{i_m} \to x_m)$ and $(b_{j_1} \to x_1, \ldots, b_{j_m} \to x_m)$ are called **unifies** of $R(x_1, \ldots, x_m)$ with $P(a_1, \ldots, a_m)$ and $Q(b_1, \ldots, b_m)$ and are denoted as $\lambda_{R,P}$ and $\lambda_{R,Q}$, respectively. [1], [20]

Note that concept of isomorphism of elementary conjunctions of atomic predicate formulas differs from the concept of equivalence of these formulas, because they can have significantly different arguments. In fact, for isomorphic formulas there are permutations of their arguments such that they define the same relationship between their arguments.

Definition 2: Elementary conjunction $C(x_1, \ldots, x_n)$ is called a **common up to the names of arguments sub-formula** of two elementary conjunctions $A(a_1, \ldots, a_m)$ and $B(b_1, \ldots, b_k)$ if it is isomorphic to some sub-formulas $A'(a'_1, \ldots, a'_n)$ and $B'(b'_1, \ldots, b'_n)$ of $A(a_1, \ldots, a_m)$ and $B(b_1, \ldots, b_k)$, respectively.

The unifiers of $C(x_1, \ldots, x_n)$ with $A'(a'_1, \ldots, a'_n)$ and $B'(b'_1, \ldots, b'_n)$ will be denoted as $\lambda_{C,A}$ and $\lambda_{C,B}$, respectively [19].

For example, let

$$A(x, y, z) = p_1(x) \,\&\, p_1(y) \,\&\, p_1(z) \,\&\, p_2(x, y) \,\&\, p_3(x, z),$$

$$B(x, y, z) = p_1(x) \,\&\, p_1(y) \,\&\, p_1(z) \,\&\, p_2(x, z) \,\&\, p_3(x, z).$$

The formula

$$P(u, v) = p_1(u) \,\&\, p_1(v) \,\&\, p_2(u, v)$$

is their common up to the names of variables sub-formula with the unifiers $\lambda_{P,A} = (x \to u, y \to v)$ and $\lambda_{P,B} = (x \to u, z \to v)$ because

$$P(x, y) = p_1(x)\&p_1(y)\&p_2(x, y)$$

is a sub-formula of $A(x, y, z)$ and

$$P(x, z) = p_1(x)\&p_1(z)\&p_2(x, z)$$

is a sub-formula of $B(x, y, z)$.

Definition 3: Elementary conjunction $C(x_1, \ldots, x_n)$ is called a **maximal common up to the names of arguments sub-formula** of two elementary conjunctions $A(a_1, \ldots, a_m)$ and $B(b_1, \ldots, b_k)$ if it is their common up to the names of arguments sub-formula and after adding any literal to it, it ceases to be one [19].

For further presentation, the concept of partial sequence [9] is important.

The problem of checking if the formula $A(\overline{x})$ or some its sub-formula $A'(\overline{y})$ is a consequence of the set of formulas $S(\omega)$ is under consideration in [9]. Here the list of arguments $\overline{y}'$ is a sub-list of the list of arguments $\overline{y}$.

Definition 4: Let $A(\overline{x})$ and $B(\overline{y})$ be elementary conjunctions. If $A(\overline{x}) \Rightarrow \exists \overline{y}_{\neq}\, B(\overline{y})$ is not valid but for some sub-formula $B'(\overline{y}')$ of $B(\overline{y})$ the sequence $A(\overline{x}) \Rightarrow \exists \overline{y}'_{\neq}\, B'(\overline{y}')$ is true, we will say that $B(\overline{y})$ is a **partial sequence** from $A(\overline{x})$ and denote this by $A(\overline{x}) \Rightarrow_P \exists \overline{y}_{\neq}\, B(\overline{y})$ [1].

A constructive algorithm for a proof of partial sequence is in [22]. While using a constructive algorithm to prove $A(\overline{x}) \Rightarrow_P \exists \overline{y}_{\neq}\, B(\overline{y})$ we can find the maximal sub-formula $B'(\overline{y}')$ such that $A(\overline{x}) \Rightarrow \exists \overline{y}'_{\neq}\, B'(\overline{y}')$ and such values $\overline{x}'$ ($\overline{x}'$ is a permutation of a sub-string of $\overline{x}$) for $\overline{y}'$ that $B'(\overline{x}')$ is a sub-formula of $A(\overline{x})$. It means that we find a maximal common up to the names of arguments

[3] $f(x) = O(g(x))$ means that there is such a constant C that for every x the inequality $f(x) \leq C \cdot g(x)$ is valid.

sub-formula $B'(\overline{y}')$ of two elementary conjunctions $A(\overline{x})$ and $B(\overline{y})$ and its unifier with $B'(\overline{x}')$ (a sub-formula of $A(\overline{x})$).

Further, for fuzzy recognition of an object we need an extension of the partial sequence concept.

Every sub-formula $A'(\overline{x}')$ of the formula $A(\overline{x})$ is called its **fragment** [1].

Let a and a' be the numbers of atomic formulas in $A(\overline{x})$ and $A'(\overline{x}')$, respectively, m and m' be the numbers of objective variables in $\overline{x}$ and $\overline{x}'$, respectively.

Numbers q and r are calculated by the formulas $q = \frac{a'}{a}$, $r = \frac{m'}{m}$ and characterize the degree of coincidence between $A(\overline{x})$ and $A'(\overline{x}')$. For every elementary conjunction $A(\overline{x})$ and its fragment $A'(\overline{x}')$ it is true that $0 < q \leq 1$, $0 < r \leq 1$. Besides, $q = r = 1$ if and only if $A'(\overline{x}')$ coincides with $A(\overline{x})$.

Under these notations, the formula $A'(\overline{x}')$ will be called a (q,r)**-fragment of the formula** $A(\overline{x})$ [1].

If $S(\omega) \Rightarrow \exists\overline{x}_{\neq}\, A(\overline{x})$ is not valid but for some (q,r)-fragment $A'(\overline{x}')$ $(q \neq 1)$ of $A(\overline{x})$ the sequence $S(\omega) \Rightarrow \exists\overline{x}'_{\neq}\, A'(\overline{x}')$ is true, we will say that $S(\omega) \Rightarrow_P \exists\overline{x}_{\neq}\, A(\overline{x})$ is a **partial** (q,r)**-sequence** for description $S(\omega)$ [1].

A (q,r)-fragment $A'(\overline{x}')$ of the formula $A(\overline{x})$ with maximal value of q satisfying $S(\omega) \Rightarrow \exists\overline{x}'_{\neq}\, A'(\overline{x}')$ will be called a **maximal fragment of the formula** $A(\overline{x})$ for description $S(\omega)$ [1].

As for a maximal fragment of the formula $A(\overline{x})$ the checking whether $S(\omega) \Rightarrow \exists\overline{x}'_{\neq}\, A'(\overline{x}')$ may be done by some constructive method then such values $\overline{\tau}$ $(\tau \subseteq \omega)$ for the list of variables $\overline{x}'$ that $S(\omega) \Rightarrow A'(\overline{\tau})$ will be found.

Definition 5: Conjunction of literals from $A(\overline{x})$ which are not in $A'(\overline{x}')$ is called a **complement** of $A'(\overline{x}')$ up to $A(\overline{x})$ [1].

A complement of $A'(\overline{x}')$ up to $A(\overline{x})$ will be denoted by $C_{A(\overline{x})}A'(\overline{x}')$.

Definition 6: A (q,r)-fragment $A'(\overline{x}')$ of the formula $A(\overline{x})$ is called **contradictory** to the description $S(\omega)$ on the list of constants $\overline{\tau}$ if $S(\omega)$ and $C_{[A(\overline{x})]^{\overline{x}'}_{\overline{\tau}}}A'(\overline{\tau})$ lead to the contradiction, i.e., $S(\omega) \Rightarrow \neg C_{[A(\overline{x})]^{\overline{x}'}_{\overline{\tau}}}A'(\overline{\tau})$ [1].

Here the denotation $[A(\overline{x})]^{\overline{x}'}_{\overline{\tau}}$ is used for the result of substitution of the constants from the list $\overline{\tau}$ instead of the corresponding variables from the list $\overline{x}'$.

3 Level Description

As noted above, estimates of the number of steps of the proof of (4) exponentially depends on the number of variables or on the number of literals of the formula $A(\overline{x})$. Moreover, the proof of formulas (1), (2) and (3) in which $A_k(\overline{x}_k) = A_{k,1}(\overline{x}_{k,1}) \vee \cdots \vee A_{k,m_k}(\overline{x}_{k,m_k})$, with the available algorithms for solving (4), can be reduced to a sequential proof of (4) with $A_{k,1}(\overline{x}_{k,1}), \ldots, A_{k,m_k}(\overline{x}_{k,m_k})$ on the right side.

Due to the unprovable, but repeatedly confirmed in practice, statement of Einstein "God is subtle, but he is not malicious" elementary conjunctions included in the description of one class must have common up to the names of arguments sub-formulas. Therefore, it is natural to break the proof of this formula into a series of formulas of the same kind, but with lower values of the essential parameters. This may be done by means of the following procedure.

– Extract "frequently occurred" common up to the names of arguments sub-formulas $P^1_i(\overline{y}^1_i)$ $(i = 1, \ldots, n_1)$ of goal formulas $A_{k,1}(\overline{x}_{k,1}), \ldots, A_{k,m_k}(\overline{x}_{k,m_k})$ with "small complexity". Simultaneously we find unifiers of $P^1_i(\overline{y}^1_i)$ and sub-formulas of $A_k(x_1, \ldots, x_m)$.

– For every sub-formula $P^1_i(\overline{y}^1_i)$ a new 1st level predicate p^1_i with one 1st level variable y^1_i, defined by the equivalence $p^1_i(y^1_i) \Leftrightarrow P^1_i(\overline{y}^1_i)$, is introduced. This 1st-level variable y^1_i is a variable for a string of initial variables.

– Replace each occurrence of a formula, isomorphic to $P^1_i(\overline{y}^1_i)$ $(i = 1, \ldots, n_1)$, into $A_{k,j}(\overline{x}_{k,j})$ $(j = 1, \ldots, m_k)$ with a literal $p^1_i(y^1_i)$. The earlier found unifier gives us the particular string of initial variables which are in this occurrence of y^1_i. Denote the received from $A_{k,j}(\overline{x}_{k,j})$ formula by means of $A^1_{k,j}(\overline{x}^1_{k,j})$.

For $l = 1, \ldots, L-1$ repeat this procedure for $A^l_{k,1}(\overline{x}^l_{k,1}), \ldots, A^l_{k,m_k}(\overline{x}^l_{k,m_k})$ and receive formulas $A^{l+1}_{k,1}(\overline{x}^{l+1}_{k,1}), \ldots, A^{l+1}_{k,m_k}(\overline{x}^{l+1}_{k,m_k})$. The process will end because the number of literals in $A^l_{k,j}(\overline{x}^l_{k,j})$ decreases with increasing l.

Level description of formulas $A_{k,1}(\overline{x}_{k,1}), \ldots, A_{k,m_k}(\overline{x}_{k,m_k})$ has the following form [8].

$$\left\{\begin{array}{ccc} & A^L_{k,j}(\overline{x}^L_{k,j}) & (j = 1, \ldots, m_k) \\ p^1_1(y^1_1) & \Leftrightarrow & P^1_1(\overline{y}^1_1) \\ & \vdots & \\ p^1_{n_1}(y^1_{n_1}) & \Leftrightarrow & P^1_{n_1}(\overline{y}^1_{n_1}) \\ & \vdots & \\ p^l_i(y^l_i) & \Leftrightarrow & P^l_i(\overline{y}^l_i) \\ & \vdots & \\ p^L_{n_L}(y^L_{n_L}) & \Leftrightarrow & P^L_{n_L}(\overline{y}^L_{n_L}) \end{array}\right. . \quad (5)$$

The described procedure deals with such intuitive notions as "frequently occurred" and having "small complexity" sub-formula. In the following algorithms it is supposed that "frequently occurred" means that a sub-formula appears at least twice and "small complexity" means that the number of a sub-formula literals is less then such a number of at least one of the initial formulas.

3.1 Construction of a Level Description of Classes

An algorithm for construction of a level description of classes is presented in [19], [23]. This algorithm contains two parts: 1) extraction of common up to the names of arguments sub-formula from the class description, 2) forming a level description. Here it is presented with more accuracy.

1) Extraction of common up to the names of arguments sub-formulas from the class description

Let the description of a class be $A(\overline{x}) = A_1(\overline{x}_1) \vee \cdots \vee A_K(\overline{x}_K)$, where $A_1(\overline{x}_1), \ldots, A_K(\overline{x}_K)$ are elementary conjunctions of literals.

• For every i and j $(i < j)$ check whether $A_i(\overline{x}_i) \Rightarrow_P \exists\overline{x}_{j\neq}\, A_j(\overline{x}_j)$.

Using the notion of partial sequence we can obtain a maximal common up to the names of arguments sub-formula $Q^1_{i,j}(\overline{z}^1_{i,j})$ of two elementary conjunctions $A_i(\overline{x}_i)$ and $A_j(\overline{x}_j)$ and unifiers $\lambda_{Q^1_{i,j},A_i}$ and $\lambda_{Q^1_{i,j},A_j}$. Let s_i be the lists of indices of $Q^l_{s_i}(\overline{z}^l_{s_i})$.

• For $l = 1, \ldots, L-1$ do.

• For every s_i and s_j $(s_i \neq s_j)$ do.

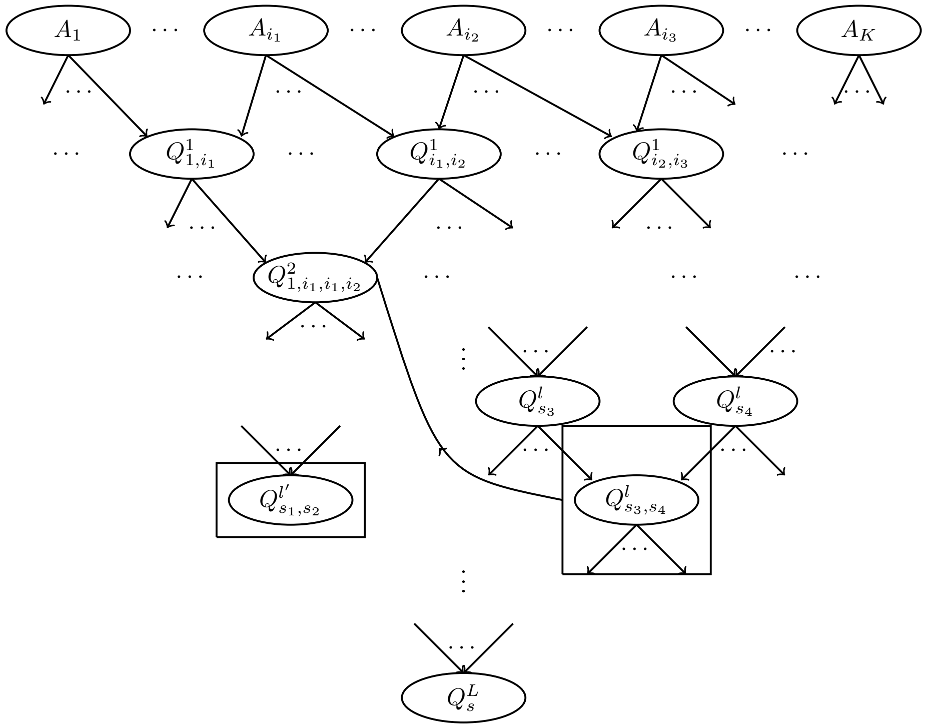


Figure 1: Scheme of common up to the names of arguments sub-formulas extraction.

– In the similar way find a maximal common up to the names of arguments sub-formula $Q^{l+1}_{s_i,s_j}(\overline{z}^{l+1}_{s_i,s_j})$ of the formulas $Q^l_{s_i}(\overline{z}^l_{s_i})$ and $Q^l_{s_j}(\overline{z}^l_{s_j})$ and unifiers $\lambda_{Q^{l+1}_{s_i,s_j},Q^l_{s_i}}$ and $\lambda_{Q^{l+1}_{s_i,s_j},Q^l_{s_j}}$.

– If a formula $Q^{l+1}_{s_i,s_j}(\overline{z}^{l+1}_{s_i,s_j})$ is isomorphic to some previously obtained $Q^{l'}_s(\overline{z}^{l'}_s)$ for some s and $l' \leq l+1$ then it is deleted from the set of the $(l+1)$th level formulas and we wright down unifiers $\lambda_{Q^{l'}_s,Q^l_{s_i}}$ and $\lambda_{Q^{l'}_s,Q^l_{s_j}}$.

Note that the length of $Q^{l+1}_{s_i,s_j}(\overline{z}^{l+1}_{s_i,s_j})$ decreases with increasing l. That is why the process would stop.

A scheme of the extraction is presented in Figure 1. In order not to overload the scheme, the arguments of the formulas are not written on it and instead of $Q^l_{s_i}(\overline{z}^l_{s_i})$ it is written $Q^l_{s_i}$.

In this scheme it is supposed that highlighted together with its connections formula $Q^l_{s_3,s_4}$ is isomorphic to some previously obtained formula, for example, to Q^2_{1,i_1,i_1,i_2} and that's why it is deleted from the set of the lth level formulas and all its connections are rewrited to Q^2_{1,i_1,i_1,i_2}.

A formula $Q^{l'}_{s_1,s_2}$ that does not have a common sub-formula with any other formulas, is highlighted on it.

2) Forming a level description.

Further, we assume that $Q^0_i(\overline{z}^0_i)$ is $A_i(\overline{x}_{k,i})$.

• Every formula of the type $Q^l_s(\overline{z}^l_s)$ which has no sub-formula of the type $Q^{l+1}_{s,s_j}(\overline{z}^{l+1}_{s,s_j})$ for any s_j is declared as $P^1_i(\overline{y}^1_i)$ $(i = 1, \dots, n_1)$.[4]

A new 1st level predicate defined as $p^1_i(y^1_i) \Leftrightarrow P^1_i(\overline{y}^1_i)$ is introduced and $p^1_i(y^1_i)$ is substituted (using the correspondent unifies) instead of $P^1_i(\overline{y}^1_i)$ into all formulas of the type $Q^l_s(\overline{z}^l_s)$ $(l \geq 0)$ such that there is an oriented path from $Q^l_s(\overline{z}^l_s)$ to $P^1_i(\overline{y}^1_i)$.

• For $l = 1, \dots, L-1$ do.

For every $i = 1, \dots, n_l$ if $P^l_i(\overline{y}^l_i)$ is a maximal common up to the names of arguments sub-formula of formulas $Q^{l'}_{s_{i'}}(\overline{z}^{l'}_{s_{i'}})$ and $Q^{l'}_{s_{j'}}(\overline{z}^{l'}_{s_{j'}})$ then these formulas are declared as $P^{l+1}_{i''}(\overline{y}^{l+1}_{i''})$ and $P^{l+1}_{i''+1}(\overline{y}^{l+1}_{i''+1})$ $(i'' = 1, \dots, n_{l+1}-1)$.

New $(l+1)$th level predicates $p^{l+1}_{i''}(y^{l+1}_{i''})$ and $p^{l+1}_{i''+1}(y^{l+1}_{i''})$ defined as $p^{l+1}_{i''}(y^{l+1}_{i''}) \Leftrightarrow P^{l+1}_{i''}(\overline{y}^{l+1}_{i''})$ and $p^{l+1}_{i''+1}(y^{l+1}_{i''+1}) \Leftrightarrow P^{l+1}_{i''+1}(\overline{y}^{l+1}_{i''+1})$ are introduced and every of them is substituted (using the correspondent unifies) instead of $P^{l+1}_{i''}(\overline{y}^{l+1}_{i''})$ and $P^{l+1}_{i''+1}(\overline{y}^{l+1}_{i''+1})$, respectively, into all formulas of the type $Q^{l'}_s$ such that there is an oriented path from Q^l_s to at least one of them.

If $P^l_i(\overline{y}^l_i)$ corresponds to the cell marked as Q^2_{1,i_1,i_1,i_2} in Figure 1 then $p^{l+1}_{i''}(y^{l+1}_{i''})$ is substituted instead of $P^{l+1}_{i''}(\overline{y}^{l+1}_{i''})$ into the formulas corresponding the cells marked as Q^1_{1,i_1}, A_1 and A_{i_1}. The literal $p^{l+1}_{i''+1}(y^{l+1}_{i''+1})$ is substituted instead of $P^{l+1}_{i''+1}(\overline{y}^{l+1}_{i''+1})$ into the formulas corresponding the cells marked as $Q^1_{i_1,i_2}$, A_{i_1} and A_{i_2}.

The words "using the correspondent unifies" may be explained in such a way. If $P^l_i(\overline{y}^l_i)$ corresponds to the cell marked as Q^2_{1,i_1,i_1,i_2} in Figure 1 then $p^{l+1}_{i''}(y^{l+1}_{i''})$ is substituted instead of $P^{l+1}_{i''}(\overline{y}^{l+1}_{i''})$ into the formulas corresponding the cells marked as Q^1_{1,i_1}, A_1 and A_{i_1}. The literal $p^{l+1}_{i''+1}(y^{l+1}_{i''+1})$ is substituted instead of $P^{l+1}_{i''+1}(\overline{y}^{l+1}_{i''+1})$ into the formulas corresponding the cells marked as $Q^1_{i_1,i_2}$, A_{i_1} and A_{i_2}.

It may be easily proved that the problem of level description

[4]These are formulas Q^L_s and $Q^{l'}_{s_1,s_2}$ in Figure 1.

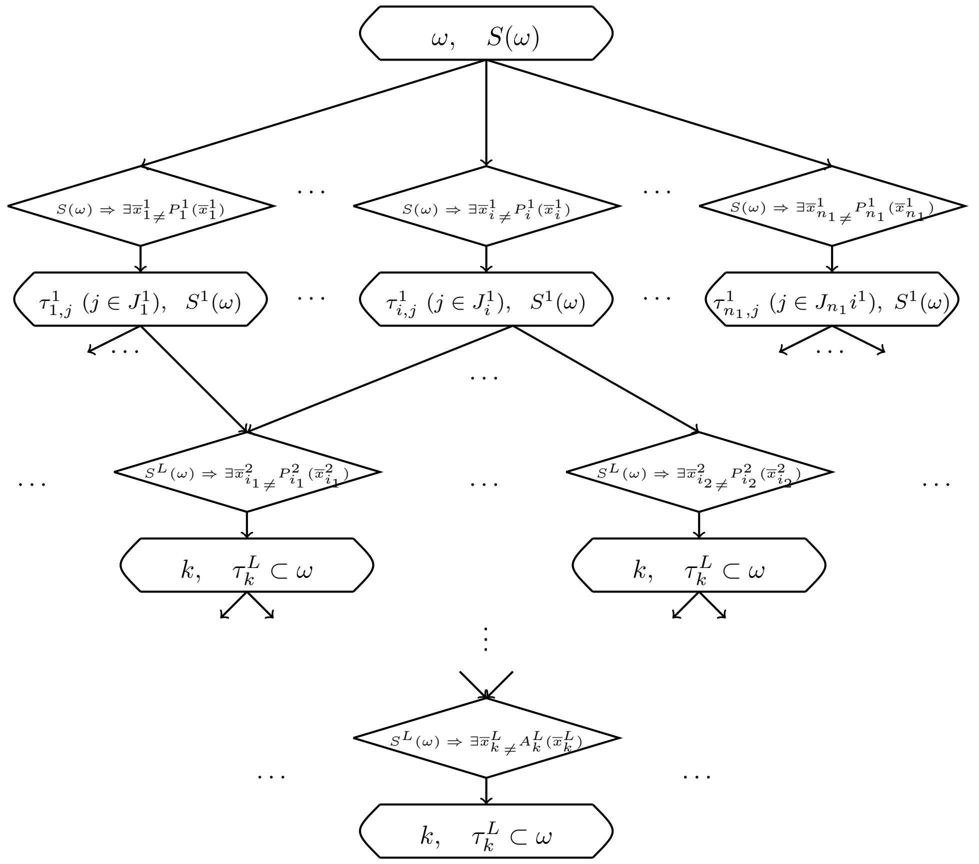

$\omega, \quad S(\omega)$
$S(\omega) \Rightarrow \exists \overline{x}^1_{1\neq} P^1_1(\overline{x}^1_1)$
$S(\omega) \Rightarrow \exists \overline{x}^1_{i\neq} P^1_i(\overline{x}^1_i)$
$S(\omega) \Rightarrow \exists \overline{x}^1_{n_1\neq} P^1_{n_1}(\overline{x}^1_{n_1})$
$\tau^1_{1,j}\ (j \in J^1_1),\ \ S^1(\omega)$
$\tau^1_{i,j}\ (j \in J^1_i),\ \ S^1(\omega)$
$\tau^1_{n_1,j}\ (j \in J_{n_1}i^1),\ S^1(\omega)$
$S^L(\omega) \Rightarrow \exists \overline{x}^2_{i_1\neq} P^2_{i_1}(\overline{x}^2_{i_1})$
$S^L(\omega) \Rightarrow \exists \overline{x}^2_{i_2\neq} P^2_{i_2}(\overline{x}^2_{i_2})$
$k, \quad \tau^L_k \subset \omega$
$k, \quad \tau^L_k \subset \omega$
$S^L(\omega) \Rightarrow \exists \overline{x}^L_{k\neq} A^L_k(\overline{x}^L_k)$
$k, \quad \tau^L_k \subset \omega$

Figure 2: Scheme of level recognition.

construction is NP-hard. In particular, the problem of checking if there exists a level description is NP-complete and the upper bounds of number of steps for traditional algorithms exponentially depends on parameters of $A_1(\overline{x}_1), \dots, A_K(\overline{x}_K)$. At the same time such a description must be constructed only once and then may be used as many times as you like.

3.2 The Use of Level Description of a Class

Checking for what k the sequence $S(\omega) \Rightarrow \exists \overline{x}_{k\neq} A_k(\overline{x}_k)$ is valid and if "yes" then finding for what value of $\overline{x}_k$ it is valid with the use of level description of a class, may be done according to the following algorithm.

• For $i = 1, \dots, n_1$ check $S(\omega) \Rightarrow \exists \overline{y}^1_{i\neq} P^1_i(\overline{y}^1_i)$ and find all lists of values $\overline{\tau}^1_{i,j}$ for the variables $\overline{y}^1_i$ such that $S(\omega) \Rightarrow P^1_i(\overline{\tau}^1_{i,j})$.[5]

Let $\tau^1_{i,j}$ be a notation for the list $\overline{\tau}^1_{i,j}$. All atomic formulas of the form $p^1_i(\tau^1_{i,j})$ with $\tau^1_{i,j}$ a notation for the list $\overline{\tau}^1_{i,j}$ are added to $S(\omega)$ to obtain $S^1(\omega)$ – a 1st level description of ω.

• For $l = 1, \dots, L-1$ do.

– For $i = 1, \dots, n_l$ check $S^l(\omega) \Rightarrow \exists \overline{y}^l_{i\neq} P^l_i(\overline{y}^l_i)$ and find all lists of values $\overline{\tau}^l_{i,j}$ for the variables $\overline{y}^l_i$ such that $S^l(\omega) \Rightarrow P^l_i(\overline{\tau}^l_{i,j})$.

– All atomic formulas of the form $p^l_i(\tau^l_{i,j})$ with $\tau^l_{i,j}$ a notation for the list $\overline{\tau}^l_{i,j}$ are added to $S^l(\omega)$ to obtain $S^{1+1}(\omega)$ – a lth level description of ω.

• – For $k = 1, \dots, K$ check $S^L(\omega) \Rightarrow \exists \overline{y}^L_{k\neq} A^L_k(\overline{y}^L_k)$ and find all lists of values $\overline{\tau}^L_{k,j}$ for the variables $\overline{y}^L_k$ such that $S^L(\omega) \Rightarrow A^L_k(\overline{\tau}^L_{k,j})$.

The object ω satisfies such formulas $A_k(\overline{x}_k)$ for which the corresponding sequence $S^L(\omega) \Rightarrow \exists \overline{y}^L_{k\neq} A^L_k(\overline{y}^L_k)$ is valid.

The scheme of level recognition is presented in Figure 2.

It is proved in [8] that for a two-level description ($L = 1$) the number of steps with the use of 2-level description decreases. For an exhaustive algorithm the number of steps decreases from $O(t^{m_k})$ to $O(n_1 \cdot t^r + t^{\delta^1_k + n_1})$, where r is a maximal number of arguments in the 1st level predicates, n_1 is the number of such predicates, δ^1_k is the number of initial variables which are presented in $A_k(\overline{x}_k)$ and are absent in $A^1_k(\overline{x}^1_k)$. Both parameters r and $\delta^1_k + n_1$ are less than m_k.

Number of steps for an algorithm based on the derivation in predicate calculus decreases from $O({s_k}^{a_k})$ to $O({s^1}^{a^1_k} + \sum_{j=1}^{n_1} s^{\rho^1_j})$, where a_k and a^1_k are maximal numbers of literals in $A_k(\overline{x}_k)$ and $A^1_k(\overline{x}^1_k)$, respectively, s and s^1 are the numbers of literals in $S(\omega)$ and $S^1(\omega)$, respectively, ρ^1_j is the numbers of literals in $P^1_i(\overline{y}^1_i)$. Both parameters a^1_k and ρ^1_j are less than a_k.

[5]Note that for every predicate P^1_i there may be many lists of values from ω satisfying it.

So, the recognition of an object with the use of a level description requires less time than that with the use of initial description.

4 Logic-Predicate Network

The inputs of a traditional neuron network are binary or many-valued characteristics of an object. The neuron network configuration is fixed that does not correspond to biological networks.

Below a logic-predicate network is described. Such a network inputs are atomic formulas setting properties of the elements composing an investigated object and relations between them [17], [19]. Further it will be shown that its configuration may be changed during its retraining.

A level description of a class of objects permits quickly enough to recognize an object which description is isomorphic to the description of an element of a training set.

A logic-predicate network consists of two blocks: a training block and a recognition block. Training block forms a level description of a class of objects according to a training set (or some formalized description of the class). Its content is presented in Figure 1.

Recognition block implements the algorithm of the use of level description of a class. It must be marked that the level description may be represented as an oriented graph presented in Figure 2.

Scheme of logic-predicate network is presented in Figure 3. Here inputs and outputs are situated in ovals and checking the conditions are situated in rhombus.

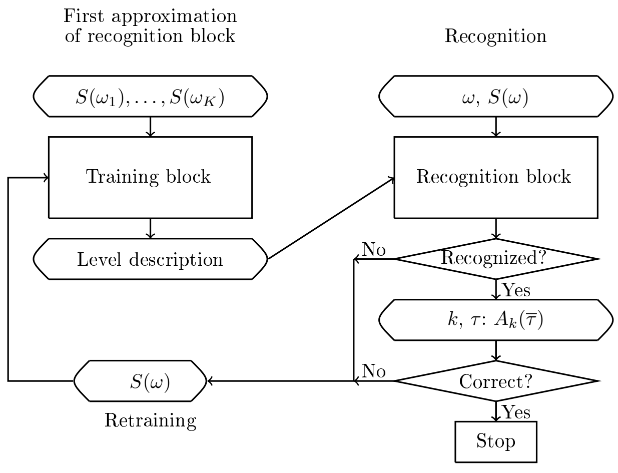


Figure 3: Scheme of logic-predicate network.

Let a training set of objects $\omega^1, \dots, \omega^K$ be given. An initial variant of the class description may be formed by replacement of every constant ω^k_j in $S(\omega^k)$ by a variable x^k_j ($k = 1, \dots, K$, $j = 1, \dots, t^k$) and substitution of the sign & between the atomic formulas. Construct a level description for these goal formulas. The first approximation to the recognition block is formed.

If after the recognition block run an object is not recognized or has wrong classification, then it is possible to retrain the network. The description of the "wrong" object must be added to the input set of the training block. The training block extracts common up to the names sub-formulas of this description and previously received formulas forming a new recognition block. Some sub-formulas, the number of layers and the number of formulas in every layer may be changed in the level description. Then the recognition block is reconstructed.

5 Example of Logic-Predicate Network Construction, its Running and Retraining

Let a set of contour images of "boxes" be to recognize. Every image may be described in the terms of two predicates V and L defined as it is presented in Figure 4.

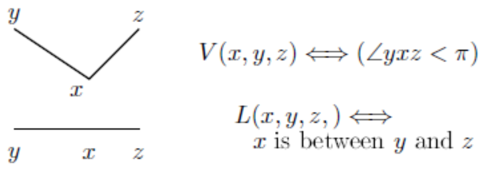

y
z
x
$V(x, y, z) \iff (\angle yxz < \pi)$
$L(x, y, z,) \iff$
x is between y and z
y
x
z

Figure 4: Initial predicates.

A training set $\{\omega_1, \omega_2, \omega_3\}$ of such images is presented in Figure 5. Here $\omega_1 = \{a_1, \dots, a_{10}\}$, $\omega_2 = \{b_1, \dots, b_8\}$, $\omega_3 = \{c_1, \dots, c_{10}\}$.

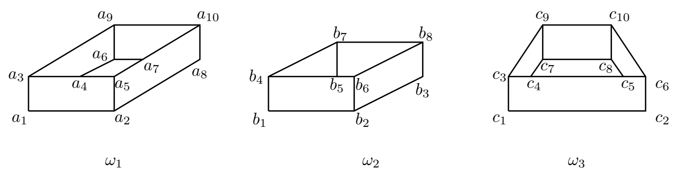


Figure 5: Training set.

In order not to overload the text, we write down only the formula $A_2(y_1, \dots, y_8)$, which is obtained from the description $S(\omega_2)$ by replacing the names b_i with the variables y_i and placing the sign & between the literals.

$A_2(y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8) =$
$V(y_1, y_4, y_2)$ & $V(y_2, y_1, y_6)$ & $V(y_2, y_1, y_3)$ & $V(y_2, y_6, y_3)$ &
$V(y_3, y_2, y_8)$ & $V(y_4, y_7, y_6)$ & $V(y_4, y_7, y_5)$ & $V(y_4, y_7, y_1)$ &
$V(y_4, y_5, y_1)$ & $V(y_4, y_6, y_1)$ & $V(y_5, y_4, y_7)$ & $V(y_5, y_4, y_6)$ &
$V(y_5, y_7, y_6)$ & $L(y_5, y_4, y_6)$ & $V(y_6, y_4, y_8)$ & $V(y_6, y_5, y_8)$ &
$V(y_6, y_8, y_2)$ & $V(y_6, y_2, y_5)$ & $V(y_6, y_2, y_4)$ & $V(y_7, y_4, y_5)$ &
$V(y_7, y_5, y_8)$ & $V(y_7, y_4, y_8)$ & $V(y_8, y_3, y_6)$ & $V(y_8, y_6, y_7)$ &
$V(y_8, y_3, y_7)$

Further, for clarity, we will write down not formulas, but drawings.

5.1 Training Block Run: Extraction of Sub-Formulas

For constructing a level description, find pairwise maximal common up to the names of arguments sub-formulas $Q_i^1(\overline{y}_i^1)$ of formulas $A_1(\overline{x}), A_2(\overline{y}), A_3(\overline{z})$ and their unifiers. See Figure 6.

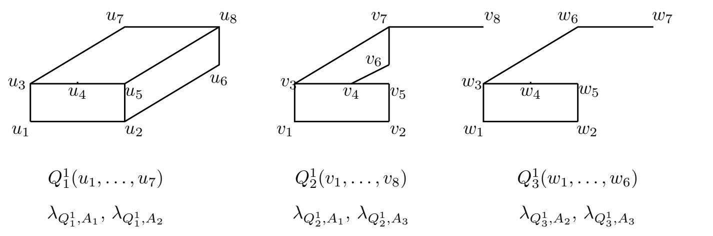


Figure 6: First extraction of pairwise maximal common up to the names of arguments sub-formulas.

Here the following unifier are formed.

Let $u^1 = (u_1,\dots,u_8)$, $v^1 = (v_1,\dots,v_8)$, $w^1 = (w_1,\dots,w_7)$.

$\lambda_{Q_1^1,A_1} = (u_1 \to x_1, u_2 \to x_2, u_3 \to x_3, u_4 \to x_4, u_5 \to x_5, u_6 \to x_8, u_7 \to x_9, u_8 \to x_{10}) = (u^1 \to x_1^1)$,

$\lambda_{Q_1^1,A_2} = (u_1 \to y_1, u_2 \to y_2, u_3 \to y_4, u_4 \to y_5, u_5 \to y_6, u_6 \to y_3, u_7 \to y_7, u_8 \to y_8) = (u^1 \to y_1^1)$,

$\lambda_{Q_2^1,A_1} = (v_1 \to x_1, v_2 \to x_2, v_3 \to x_3, v_4 \to x_4, v_5 \to x_5, v_6 \to x_6, v_7 \to x_9, v_8 \to x_{10}) = (v^1 \to x_2^1)$,

$\lambda_{Q_2^1,A_3} = (v_1 \to z_1, v_2 \to z_2, v_3 \to z_3, v_4 \to z_4, v_5 \to z_6, v_6 \to z_7, v_7 \to z_9, v_8 \to z_{10}) = (v^1 \to z_1^1)$,

$\lambda_{Q_3^1,A_2} = (w_1 \to y_1, w_2 \to y_2, w_3 \to y_4, w_4 \to y_5, w_5 \to y_6, w_6 \to y_7, w_7 \to y_8) = (w^1 \to y_2^1)$,

$\lambda_{Q_3^1,A_3} = (w_1 \to z_1, w_2 \to z_2, w_3 \to z_3, w_4 \to z_4, w_5 \to z_6, w_6 \to z_9, w_7 \to z_{10}) = (w^1 \to z_2^1)$.

Pairwise extractions of maximal common up to the names of arguments sub-formulas of $Q_1^1(u_1,\dots,u_8)$, $Q_2^1(v_1,\dots,v_8)$ and $Q_3^1(w_1,\dots,w_6)$ give only just obtained formula $Q_3^1(w_1,\dots,w_6)$. That's why we have a situation presented in Figure 7.

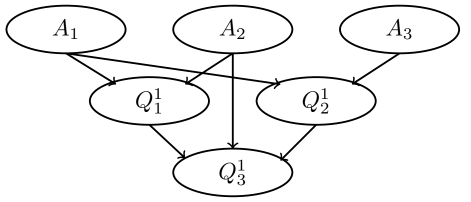


Figure 7: Connections between the extracted formulas.

5.2 Training Block Run: Forming a Level Description

The formula $Q_3^1(w_1,\dots,w_7)$ has no common sub-formulas with the other ones. Therefore, it is the 1st level predicate $P_1^1(w_1,\dots,w_7)$ of the network. A new 1st level predicate $p_1^1(w^1)$ defining as $p_1^1(w^1) \Leftrightarrow P_1^1(w_1,\dots,w_7)$ is introduced. Remind that w^1 is a new 1st level variable for the string of initial variables $(w_1, w_2, w_3, w_4, w_5, w_6, w_7)$.

Formula $P_1^1(w_1,\dots,w_7)$ (which corresponds to $Q_3^1(w_1,\dots,w_7)$) is on the end of edges from $A_2(y_1,\dots,y_8)$, $Q_1^1(u_1,\dots,u_8)$ and $Q_2^1(v_1,\dots,v_8)$ in Figure 7. The unifier of $P_1^1(w_1,\dots,w_7)$ with $A_2(y_1,\dots,y_8)$ coincides with $\lambda_{Q_3^1,A_2}$ obtained above.

The unifiers of $P_1^1(w_1,\dots,w_7)$ with $Q_1^1(u_1,\dots,u_8)$ and $Q_2^1(v_1,\dots,v_8)$ are the following.

$\lambda_{P_1^1,Q_1^1} = (w^1 \to u^2)$, where $u^2 = (u_1, u_2, u_3, u_4, u_5, u_7, u_8)$.

$\lambda_{P_1^1,Q_2^1} = (w^1 \to v^2)$, where $v^2 = (v_1, v_2, v_3, v_4, v_5, v_7, v_8)$.

The formulas $Q_1^1(u_1,\dots,u_8)$ and $Q_2^1(v_1,\dots,v_8)$ with the substitutions of $p_1^1(u^1)$ and $p_1^1(v^1)$, respectively, instead of sub-formulas isomorphic to $P_1^1(w_1,\dots,w_7)$ will form the 2nd level of the network and are denoted by $P_1^2(u_1,\dots,u_8;u^1)$ and $P_2^2(v_1,\dots,v_8;v^1)$.

It must be marked that if we know the value of the 1st level variable u^1 then we know the values of $(u_1, u_2, u_3, u_4, u_5, u_7, u_8)$. That's why the essential variables of $P_1^2(u_1,\dots,u_8;u^1)$ are $(u_6;u^1)$. Similarly the essential variables of $P_2^2(v_1,\dots,v_8;v^1)$ are $(v_6;v^1)$.

New 2nd level predicates $p_1^2(u^2)$ and $p_2^2(v^2)$ defining as $p_1^2(u^2) \Leftrightarrow P_1^2(u_1,\dots,u_8;u^1)$ and $p_2^2(v^2) \Leftrightarrow P_2^2(v_1,\dots,v_8;v^1)$ are introduced.

Taking into account only essential variables and earlier received unifiers we have the last layer formulas.

$A_{1,1}^2(x_7;x^1,x_1^2)$, where

$x^1 = (x_1, x_2, x_3, x_4, x_5, x_8, x_9, x_{10})$, $x_1^2 = (x_6;x^1)$.

$A_{1,2}^2(x_7;x^1,x_2^2)$, where

$x^1 = (x_1, x_2, x_3, x_4, x_5, x_8, x_9, x_{10})$, $x_2^2 = (x_6;x^1)$.

$A_{2,1}^2(;y^1,y_1^2)$, where

$y^1 = (y_1, y_2, y_4, y_5, y_6, y_7, y_8)$, $y_1^2 = (y_3, y_8;y^1)$.

$A_{2,2}^2(y_3y_7;y^1)$, where

$y^1 = (y_1, y_2, y_4, y_5, y_6, y_7, y_8)$.

$A_3^2(z_5, z_8;z^1,z^2)$, where

$z^1 = (z_1, z_2, z_3, z_4, z_6, z_9, z_{10})$, $z^2 = (z_4, z_7;z^1)$.

The contents of the last layer cells are the following:

$A_{1,1}^2(\overline{x};w^1,u^2) \vee A_{1,2}^2(\overline{x};w^1,v^2)$,

$A_{2,1}^2(\overline{y};w^1,u^2) \vee A_{1,2}^2(\overline{y};w^1)$,

$A_3^2(\overline{z};w^1,v^2)$.

5.3 Recognition Block Run

The recognition block is formed according to the level description constructed in the previous subsection. The scheme of recognition block is presented in Figure 8. In this scheme only names of a checking formula and results of checking are written down.

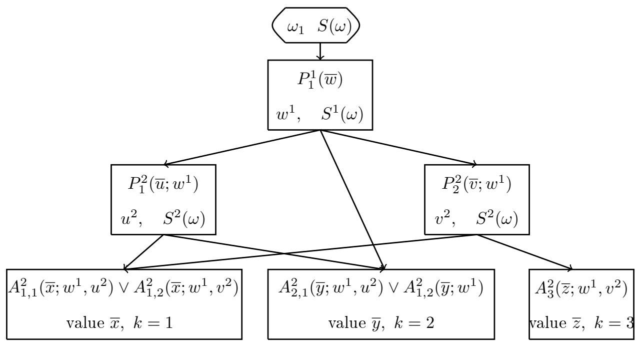

ω_1 $S(\omega)$
$P_1^1(\overline{w})$
w^1, $S^1(\omega)$
$P_1^2(\overline{u};w^1)$
u^2, $S^2(\omega)$
$P_2^2(\overline{v};w^1)$
v^2, $S^2(\omega)$
$A_{1,1}^2(\overline{x};w^1,u^2)\vee A_{1,2}^2(\overline{x};w^1,v^2)$
value $\overline{x}$, $k=1$
$A_{2,1}^2(\overline{y};w^1,u^2)\vee A_{1,2}^2(\overline{y};w^1)$
value $\overline{y}$, $k=2$
$A_3^2(\overline{z};w^1,v^2)$
value $\overline{z}$, $k=3$

Figure 8: Predicate network.

It is obvious that if an input object ω is isomorphic to one from the training set, then it will be recognized. It means that the number k and such a permutation $\overline{\omega}$ of ω that $S(\omega) \Rightarrow A_k(\overline{\omega})$ will be found.

Let a control object with the description not isomorphic to anyone from the training set is to be recognized. Such a control object $\{d_1, \ldots, d_8\}$ is presented in Figure 9.

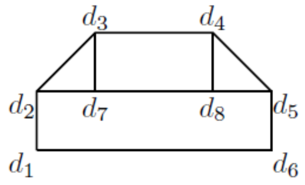


Figure 9: Control object.

Checking $S(\omega) \Rightarrow \exists\overline{w}_{\neq}P_1^1(\overline{w})$ in the first layer (see Figure 8) gives us $\overline{w} = (d_1, d_6, d_2, d_7, d_5, d_3, d_4)$. A new 1st level variable w^1 takes this list of constants as a value. Add new atomic formula to $S(\omega)$ and obtain $S^1(\omega) = S(\omega) \cup \{p_1^1(w^1)\}$.

Neither $P_1^2(\overline{u}, w^1)$ or $P_2^2(\overline{v}, w^1)$ are a consequence of $S^1(\omega)$ because, in fact, 7 variables in $\overline{u}$ and $\overline{v}$ are changed by constants from w^1.

This control object is not recognized.

5.4 Retraining the Network

Let's retrain the network with the object presented in Figure 9.

Replace every constant d_i $(i = 1, \ldots, 8)$ in its description by a variable x_i and substitute the sign & between the atomic formulas. The formula $A_4(\overline{x})$ is obtained.

Pairwise checking of partial sequence between $A_4(\overline{x})$ and $A_j(\overline{x}_j)$ $(j = 1, 2, 3)$ permits to obtain maximal common up to the names of arguments formulas Q_4^1, Q_5^1 and Q_6^1. The images of thees fragments are shown in Figure 10.

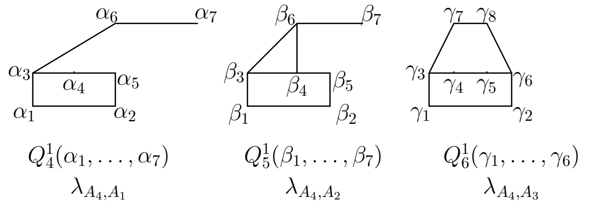


Figure 10: Pairwise extraction of maximal common up to the names of arguments sub-formulas between $A_4(\overline{x}_4)$ and $A_j(\overline{x}_j)$ $(j = 1, 2, 3)$.

One of the obtained formulas (formula $Q_4^1(\alpha_1, \ldots, \alpha_7)$) is isomorphic to the earlier obtained formula $Q_3^1(w_1, \ldots, w_7)$. That's why it would not be added to the 2nd layer of the retrained network. But all its connections with the cells of another layers would be saved.

Pairwise checking of partial sequence between Q_5^1, Q_6^1 and $Q_j^1(\overline{x}_j^1)$ $(j = 1, 2, 3)$ does not give any new formulas.

So, the 1st layer would have only the formula $Q_3^1(w_1, \ldots, w_7)$ renamed as $P_1^1(w_1, \ldots, w_7)$.

On the 2nd layer the cell with formula $P_4^2(\overline{\beta}; w^1)$ and output t^2 and the cell with formula $P_5^2(\overline{\gamma}; w^1)$ and output r^2 would be added on the end of the edge from the cell with $P_1^1(\overline{w})$. Edges from these cells to the cells with $A_j^2(\overline{x}_j; x_j^1, x_j^2)$ $(j = 1, 2, 3)$ would be added as it is shown in Figure 11.

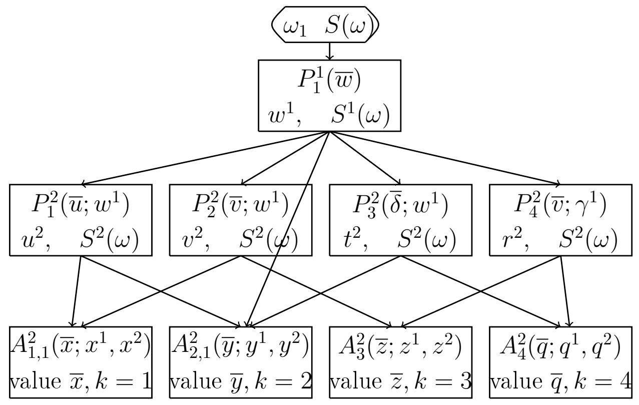

ω_1 $S(\omega)$
$P_1^1(\overline{w})$
w^1, $S^1(\omega)$
$P_1^2(\overline{u};w^1)$
u^2, $S^2(\omega)$
$P_2^2(\overline{v};w^1)$
v^2, $S^2(\omega)$
$P_3^2(\overline{\delta};w^1)$
t^2, $S^2(\omega)$
$P_4^2(\overline{v};\gamma^1)$
r^2, $S^2(\omega)$
$A_{1,1}^2(\overline{x};x^1,x^2)$
value $\overline{x}, k=1$
$A_{2,1}^2(\overline{y};y^1,y^2)$
value $\overline{y}, k=2$
$A_3^2(\overline{z};z^1,z^2)$
value $\overline{z}, k=3$
$A_4^2(\overline{q};q^1,q^2)$
value $\overline{q}, k=4$

Figure 11: Pairwise extraction of maximal common up to the names of arguments sub-formulas between $A_4(\overline{x}_4)$ and $A_j(\overline{x}_j)$ $(j = 1, 2, 3)$.

The network is retrained.

It should be noted that the examples of images discussed in the article are taken from [3]. The original predicates W, Y and T in that book for describing the images differ from the predicates V and L used here. This is due to the fact that W, Y and T are expressed in terms of the predicates V and L and do not significantly reduce the computational complexity of recognition when constructing a level description.

Both $\{W, Y, T\}$ and $\{V, L\}$ are not enough to describe real images of "boxes". So, for example, the description of the image in Figure 12 is isomorphic to the description of $A_2(y_1, \ldots, y_8)$, but you need to have a rich imagination to guess the box in it. The choice of initial predicates in each specific problem should be chosen very carefully.

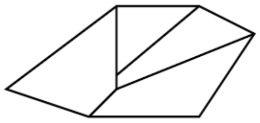


Figure 12: Is it a box?

Here we do not consider an example when, while retraining the network, not only additional cells in a layer appear, but the number of layers also changes. This could be done by retraining the network using the example of a mirror (relative to the vertical axis) image of an object from the training set. In this case, rectangle recognition will appear in the 1st layer. The contents of the remaining cells will also be changed.

6 Fuzzy Recognition by a Logic-Predicate Network

The disadvantage is that the proposed logic-predicate network does not recognize objects with descriptions that are not isomorphic to the descriptions that participate in its formation. This can be eliminated by slight modifications of the network.

Further the *i*th cell of the *l*th level will be denoted as $cell_i^l$.

Let's change the contents of the network cells presented in Figure 2 by replacing the checking of $S^{l-1}(\omega) \Rightarrow \exists \overline{x}_{i\neq}^l P_i^l(\overline{x}_i^l)$ in the $cell_i^l$ with the **partial** sequence checking $S^{l-1}(\omega) \Rightarrow_P \exists \overline{x}_{i\neq}^l P_i^l(\overline{x}_i^l)$. The degree of coincidence of a recognized object with the description of a class and the degree of recognition correctness certainty is calculated during partial sequence checking.

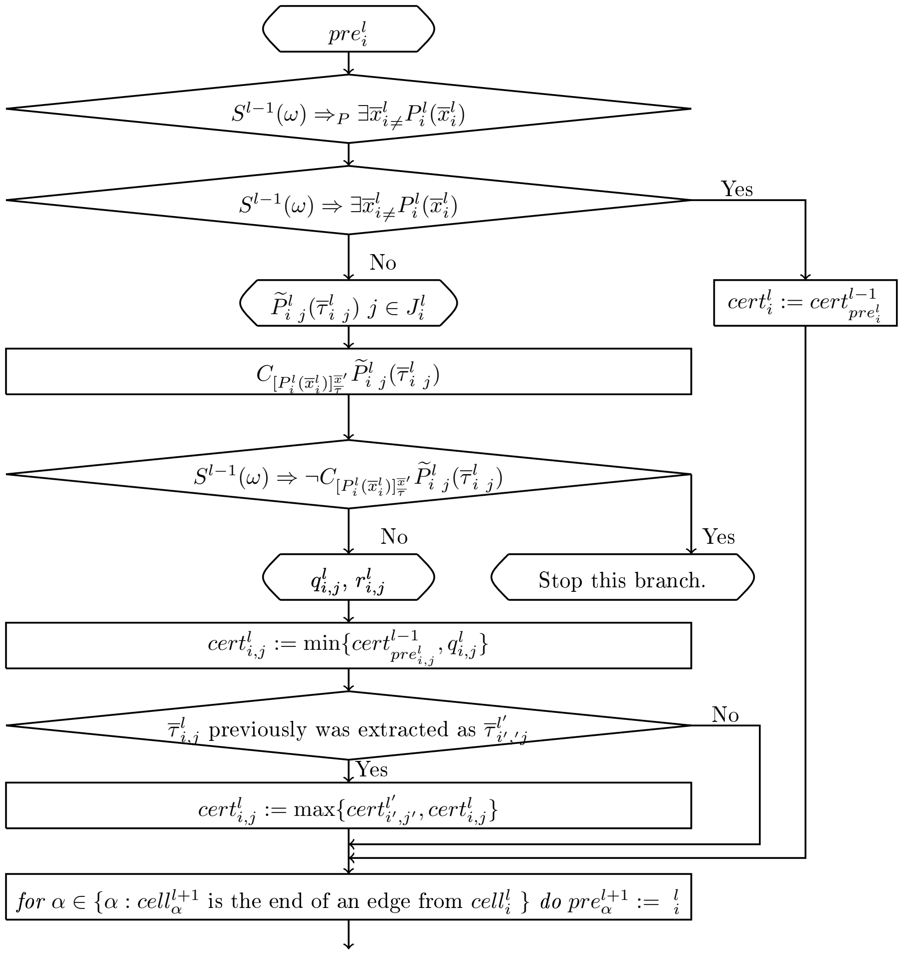

pre_i^l
$S^{l-1}(\omega) \Rightarrow_P \exists \overline{x}_{i\neq}^l P_i^l(\overline{x}_i^l)$
$S^{l-1}(\omega) \Rightarrow \exists \overline{x}_{i\neq}^l P_i^l(\overline{x}_i^l)$
Yes
No
$\widetilde{P}_{i\ j}^l(\overline{\tau}_{i\ j}^l)\ j \in J_i^l$
$cert_i^l := cert_{pre_i^l}^{l-1}$
$C_{[P_i^l(\overline{x}_i^l)]_{\overline{\tau}}^{\overline{x}'}} \widetilde{P}_{i\ j}^l(\overline{\tau}_{i\ j}^l)$
$S^{l-1}(\omega) \Rightarrow \neg C_{[P_i^l(\overline{x}_i^l)]_{\overline{\tau}}^{\overline{x}'}} \widetilde{P}_{i\ j}^l(\overline{\tau}_{i\ j}^l)$
No
Yes
$q_{i,j}^l,\ r_{i,j}^l$
Stop this branch.
$cert_{i,j}^l := \min\{cert_{pre_{i,j}^l}^{l-1}, q_{i,j}^l\}$
$\overline{\tau}_{i,j}^l$ previously was extracted as $\overline{\tau}_{i',j}^{l'}$
No
Yes
$cert_{i,j}^l := \max\{cert_{i',j'}^{l'}, cert_{i,j}^l\}$
$for\ \alpha \in \{\alpha : cell_\alpha^{l+1}$ is the end of an edge from $cell_i^l\}\ do\ pre_\alpha^{l+1} := \ _i^l$

Figure 13: The fuzzy network *i*th cell of the *l*th level contents.

For a fully identified object, both of these degrees equal to 1. For an object from a given class for which these degrees are essentially small (this is determined by an expert), you can still retrain the network.

A fuzzy network has two additional parameters:

$cert_i^l$ ($l = 0, \dots, L$, $i = 1, \dots, n_l$) – a **degree of certainty in the correctness of recognition** by the $cell_i^l$;

pre_i^l ($l = 1, \dots, L$, $i = 1, \dots, n_l$) – the **cell number of a cell preceding** the $cell_i^l$ in the current traversal of a network. The cell number has the form ${}^{l'}_{j}$, where l' is the number of a layer and j is the number of a cell in this layer

Initially, $cert_1^0 := 1$, $pre_i^1 := {}^0_1$ ($i = 1, \dots, n_1$).

The scheme of $cell_i^l$ contents of a fuzzy network is presented in Figure 13.

First of all, a partial sequence $S^{l-1}(\omega) \Rightarrow_P \exists \overline{x}^l_{i\neq} P_i^l(\overline{x}_i^l)$ is checked in this cell. If this sequence is not partial but total, then the parameter $cert_i^l$ is not changed and the calculation goes to the end of the cell.

Otherwise, all lists of constants $\overline{\tau}^l_{i\ j}$ and maximal not contradictory on $\overline{\tau}^l_{i\ j}$ with $S^{l-1}(\omega)$ sub-formulas $\widetilde{P}^l_{i\ j}(\overline{x}^l_{i\ j})$ of the formula $P_i^l(\overline{x}_i^l)$ are found while partial sequence checking.

If the sub-formulas $\widetilde{P}^l_{i\ j}(\overline{x}^l_{i\ j})$ is contradictory to the description $S(\omega)$ on the list of the found constants, then this recognition branch stops.

Remind that $S(\omega)$ and $C_{[P_i^l(\overline{x}_i^l)]^{\overline{x}'}_{\overline{\tau}}} \widetilde{P}^l_{i\ j}(\overline{\tau}^l_{i\ j})$ are in the contradiction means that $S(\omega) \Rightarrow \neg C_{[P_i^l(\overline{x}_i^l)]^{\overline{x}'}_{\overline{\tau}}} \widetilde{P}^l_{i\ j}(\overline{\tau}^l_{i\ j})$, where $C_{[P_i^l(\overline{x}_i^l)]^{\overline{x}'}_{\overline{\tau}}} \widetilde{P}^l_{i\ j}(\overline{\tau}^l_{i\ j})$ is a complement (conjunction of literals from $P_i^l(\overline{x}_i^l)$ which are not in $\widetilde{P}^l_{i\ j}(\overline{x}^l_{i\ j})$) of $\widetilde{P}^l_{i\ j}(\overline{x}^l_{i\ j})$ up to $P_i^l(\overline{x}_i^l)$.[6]

Parameters $q^l_{i\ j}$ and $r^l_{i\ j}$[7] are calculated with the use of the full form of formulas $P_i^l(\overline{x}_i^l)$ and $\widetilde{P}_i^l(\overline{y}_i^l)$, i.e., with the replacement in them of each atomic formula of the levels l' ($l' < l$) by the defining elementary conjunction.

Except this, a degree of certainty $cert_i^l$ that the recognition would be valid is calculated in every cell. While first visit to the $cell_i^l$ the value of $cert_i^l$ will be $cert_i^l := \min\{cert^{1-1}_{pre_i^l}, q_i^l\}$. This corresponds to conjunction of degrees of certainty while successive passage along one branch of network traversal.

While next visit of the $cell_i^l$ the value of $cert_i^l$ will be $cert_i^l := \max\{cert_i^l, \min\{cert^{1-1}_{pre_i^l}, q_i^l\}\}$. This corresponds to disjunction of degrees of certainty while parallel passage along different branches of network traversal.

Why can we visit a cell not only once? For example, in Figure 11 the 3rd layer cell with A_1^2 may be visited by a path including the cells with P_1^1, P_1^2 and A_1^2 and by a path including the cells with P_1^1, P_2^2 and A_1^2.

In the last layer of the network, objects τ_k^L would be received and for each object the degree of certainty q_k^L, that it is the r_k^Lth part of an object satisfying the description $A_k(\overline{x}_k)$, would be calculated.

A flowchart of fuzzy recognition network can not be presented in the paper because it is very large. It consists of the "Scheme of logic-predicate network" shown in Figure 3, in which instead of the "Training block" the "Scheme of common up to the names of arguments sub-formulas extraction" (Figure 1) is inserted, and the "Scheme of level recognition" (Figure 2) is inserted instead of the "Recognition block". At the same time, in Figure 2, each element enclosed in a rhombus is replaced by "The fuzzy network ith cell of the lth level contents" (Figure 13).

7 Model Example of Fuzzy Recognition

Let's look how a fuzzy network (corresponding to that constructed in the section 6 and presented in Figure 8) recognizes the control object presented in Figure 9.

To recognize it, a partial sequence $S(d_1, d_2, \dots, d_8) \Rightarrow_P \exists w_1 \dots w_6 P_1^1(w_1, \dots, w_6)$ is checked in the **1st layer**. There is a total sequence and the 1st level variable $w^1 = (w_1, w_2, w_3, w_4, w_5, w_6, w_7)$ takes the value $(d_1, d_6, d_2, d_7, d_5, d_3, d_4)$.

$cert_1^1 := 1$, $pre_1^2 := {}^1_1$, $pre_2^2 := {}^1_1$, $pre_2^3 := {}^1_1$.

The result of the 1st layer fuzzy cell run is presented in Figure 14.[8]

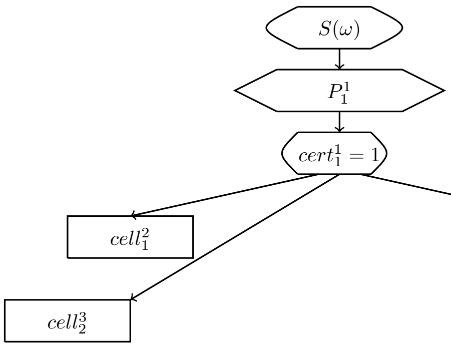


Figure 14: The result of the 1st layer fuzzy cell run.

In the **2nd layer**, partial sequence $S^1(d_1, d_2, \dots, d_8, w^1) \Rightarrow_P \exists u_1 \dots u_{7\neq} P_1^2(u_1 \dots u_8; w^1)$ is checked. It must be noted that in fact $S^1(d_1, d_2, \dots, d_8, w^1) \Rightarrow_P \exists u_6 P_1^2(d_1, d_6, d_2, d_7, d_5, u_6, d_3, d_4; w^1)$ is checked. That is, there is only one variable at the right-hand side of the formula.

Since we need to check formulas for consistency, we note that $V(x, y, z) \Rightarrow \neg V(x, z, y)$.

There is only a partial sequence. A formula $\tilde{P}_1^2(d_1, d_6, d_2, d_7, d_5, d_3, d_4; w^1)$ is the maximal sub-formula such that $S^1(d_1, d_2, \dots, d_8, w^1) \Rightarrow \tilde{P}_1^2(d_1, d_6, d_2, d_7, d_5, d_3, d_4; w^1)$. Its complement up to $P_1^2(d_1, d_6, d_2, d_7, d_5, d_3, d_4, u_6; w^1)$ contains three literals with predicate V, one of them is $V(u_6, d_4, d_6)$. This contradicts $S^1(d_1, d_2, \dots, d_8, w^1)$ containing $V(u_6, d_6, d_4)$. This branch of the network stops.

There is another cell in the 2nd layer on the end of an edge from $cell_1^1$. This is $cell_2^2$.

Check partial sequence $S^1(d_1, d_2, \dots, d_8, w^1) \Rightarrow_P \exists v_1 \dots v_8 P_2^2(v_1 \dots v_8; w^1)$. In fact, $S^1(d_1, d_2, \dots, d_8, w^1) \Rightarrow_P \exists v_6 P_2^2(d_1, d_6, d_2, d_7, d_5, v_6, d_3, d_4; w^1)$ is checked.

[6] See the definitions of a complement and contradictory formulas in the section 2.

[7] Here $q^l_{i\ j}$ and $r^l_{i\ j}$ are ratios of the number of literals and variables, respectively, in $\widetilde{P}^l_{i\ j}(\overline{x}^l_{i\ j})$ and $P_i^l(\overline{x}_i^l)$. See definition of (q, r) fragment in the section 2.

[8] Not to overload the following schemes, only the name of the formula is written in the cell instead of the formula of corresponding partial seguence.

There is only partial sequence. A formula $\tilde{P}_2^2(d_1, d_6, d_2, d_7, d_5, v_6, d_3, d_4; w^1)$ is the maximal sub-formula such that $S^1(d_1, d_2, \ldots, d_8, w^1) \Rightarrow \tilde{P}_2^2(d_1, d_6, d_2, d_7, d_5, v_6, d_3, d_4; w^1)$. Its complement up to $P_2^2(d_1, d_6, d_2, d_7, d_5, v_6, d_3, d_4; w^1)$ contains 5 literals with predicate V

$V(v_6, d_7, d_3)$, $V(d_3, v_6, d_2)$, $V(d_7, v_6, d_2)$, $V(d_7, d_8, v_6)$, $V(d_7, d_5, v_6)$

and 3 literals with predicate L

$L(d_7, d_2, d_8)$, $L(d_8, d_2, d_5)$, $L(d_8, d_7, d_5)$, but no one of them is in contradiction with $S^1(d_1, d_2, \ldots, d_8, w^1)$. The value of v^2 is $(v_6; w^1)$. It means that one 1st level variable v_6 does not take value.

The number of literals in $P_2^2(v_1 \ldots v_8; w^1)$ (taking into account its full form in which the 1st level predicate is changed by the defining formula only with initial variables) is $a_2^2 = 16$. The number of literals in $\tilde{P}_2^2(d_1, d_6, d_2, d_7, d_5, v_6, d_3, d_4; w^1)$ is $\tilde{a}_2^2 = 8$.

That's why $q_2^2 = \frac{8}{16} = \frac{1}{2}$ and $cert_2^2 := \min\{1, \frac{1}{2}\} = \frac{1}{2}$.

There are two edges from the cell with P_2^2 to the 3rd layer cells with $A_{1,2}^2$ and with A_3^2. That's why $pre_1^3 := \frac{2}{2}$, $pre_3^3 := \frac{2}{2}$.

The result of the 2nd layer fuzzy cells run is presented in Figure 15.

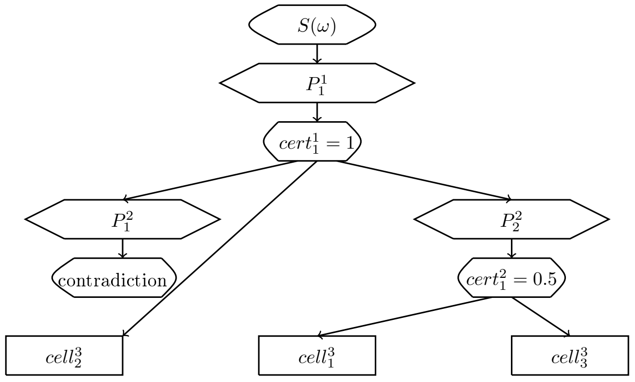


Figure 15: The result of the 2nd layer fuzzy cell run.

In the **3rd layer** while going from the cell with P_1^1 to the $cell_2^3$ with $A_{2,1}^2$ partial sequence $S^1(d_1, \ldots, d_8; w^1) \Rightarrow_P \exists y_1 \ldots y_{8\neq} A_{2,1}^2(y_1 \ldots y_8; v^1)$ is checked. In fact, a partial sequence $S^1(d_1, \ldots, d_8; w^1) \Rightarrow_P \exists y_3 A_{2,1}^2(d_1, d_6, y_3, d_2, d_7, d_5, d_3, d_4; v^1)$ with the only one initial variable y_3 is checked.

There is only a partial sequence. The same string of values for variables in v^2, as it was in the second layer, partially satisfies $A_{1,2}^2(d_1, d_6, y_3, d_2, d_7, d_5, d_3, d_4; v^1)$. But the description of this fragment has two additional literals $V(d_7, d_2, d_3)$ and $V(d_7, d_3, d_5)$.

The number of literals in $A_2(y_1 \ldots y_8; v^1)$ is $a_2 = 22$. The number of literals in $\tilde{A}_{1,2}^2(y_1 \ldots y_8; v^1, v^2)$ is $\tilde{a}_{1,2}^2 = 10$. That's why $q_2^3 = \frac{10}{22} \approx 0.455$ and $cert_2^3 := \min\{1, \frac{10}{22}\} \approx 0.455$.

While going from the cell with P_2^2 to the $cell_1^3$ with $A_{1,2}^2$ partial sequence $S^2(d_1, d_2, \ldots, d_8; v^1, v^2) \Rightarrow_P \exists v_1 \ldots v_{10\neq} A_{1,2}^2(v_1 \ldots v_{10}; v^1, v^2)$ is checked. In fact $S^2(d_1, d_2, \ldots, d_8; v^1, v^2) \Rightarrow_P \exists v_6, v_7, v_{8\neq} A_{1,2}^2(d_1, d_6, d_2, d_7, d_5, v_6, v_7, v_8, d_3, d_4; v^1, v^2)$ with 3 initial variables v_6, v_7, v_8 is checked.

There is only a partial sequence. The same string of values for variables in v^2, as it was in the second layer, partially satisfies $A_{1,2}^2(v_1 \ldots v_{10}; v^1, v^2)$.

The number of literals in $A_1(x_1, \ldots, x_{10})$ is $a_1 = 34$. The number of literals in $\tilde{A}_{1,2}^2(v_1 \ldots v_{10}; v^1, v^2)$ is $\tilde{a}_{1,2}^2 = 8$. That's why $q_1^3 = \frac{8}{34} \approx 0.235$ and $cert_1^3 := \min\{1, \frac{8}{34}\} \approx 0.235$.

While going from the cell with P_2^2 to the $cell_2^3$ with A_3^2 partial sequence $S^2(d_1, d_2, \ldots, d_8; v^1, v^2) \Rightarrow_P \exists z_1 \ldots z_{10\neq} A_3^2(z_1 \ldots z_{10}; v^1, v^2)$ is checked. In fact $S^2(d_1, d_2, \ldots, d_8; v^1, v^2) \Rightarrow_P \exists z_7, z_{8\neq} A_3^2(d_1, d_6, d_2, d_7, d_8, d_5, z_7, z_8, d_3, d_4; v^1, v^2)$ with 2 initial variables z_7, z_8 is checked.

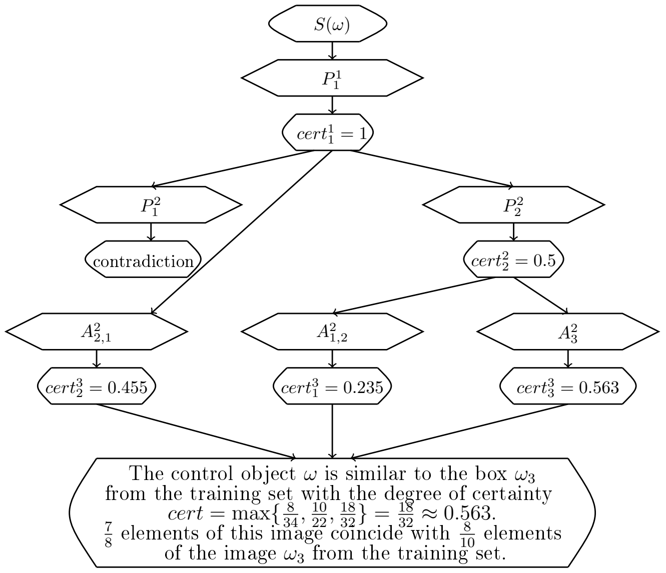
There is only a partial sequence. The variables z_7, z_8 have not taken values. All literals with these variables are absent in $\tilde{A}_3^2(d_1, d_6, d_2, d_7, d_8, d_5, z_7, z_8, d_3, d_4; v^1, v^2)$. The number of such literals is 14.

The number of literals in $A_3^2(z_1 \ldots z_{10}$ is $a_3 = 32$. The number of literals in $\tilde{A}_3^2(z_1 \ldots z_{10}; v^1, v^2)$ is $\tilde{a}_3^2 = 32 - 14 = 18$. That's why $q_2^3 = \frac{18}{32} \approx 0.563$ and $cert_2^3 := \min\{1, \frac{18}{32}\} \approx 0.563$.

The full degree of certainty that it is a representative of the class of "boxes" that is given as a control image is $cert = \max\{\frac{8}{34}, \frac{10}{22}, \frac{18}{32}\} = \frac{18}{32} \approx 0.563$.

Moreover, $\frac{7}{8}$ elements of this image coincide with $\frac{8}{10}$ elements of the image ω_3 from the training set.

The result of the fuzzy network run is presented in Figure 16.



$S(\omega)$
P_1^1
$cert_1^1 = 1$
P_1^2
P_2^2
contradiction
$cert_2^2 = 0.5$
$A_{2,1}^2$
$A_{1,2}^2$
A_3^2
$cert_2^3 = 0.455$
$cert_1^3 = 0.235$
$cert_3^3 = 0.563$
The control object ω is similar to the box ω_3
from the training set with the degree of certainty
$cert = \max\{\frac{8}{34}, \frac{10}{22}, \frac{18}{32}\} = \frac{18}{32} \approx 0.563$.
$\frac{7}{8}$ elements of this image coincide with $\frac{8}{10}$ elements
of the image ω_3 from the training set.

Figure 16: The result of the fuzzy network run.

8 Results and Discussion

The main result of this article is the proposal of fuzzy recognition of complex structured objects by logic-predicate network. For a detailed presentation of this result, it was necessary to describe in detail a logic-predicate approach to the recognition of complex structured objects.

A significant disadvantage of this approach is that the problems arising are NP-hard, which to a large extent "scares off" some researchers. On the other hand, the use of predicate logic does not always seem legitimate, since the propositional logic is much simpler for understanding by non-mathematicians.

The use of level descriptions is not something essentially original to decrease the computational complexity of the main problem. It is a well-known technique in which sufficiently complex problems are divided into a sequence of problems of the same type of lower dimension. The main problem was HOW to build such a description. To this end, the mathematical apparatus for extracting the general sub-formulas of two elementary conjunctions of predicate formulas was developed and briefly presented in this paper.

Based on the level description of classes, it was possible to formulate the concept of a logic-predicate network. Unlike existing networks, for example, neural or Bayesian, when retraining such a network, a logic-predicate network can (and, as a rule, this happens) change its configuration. This corresponds to the training of a person in whose brain connections between one neural cells break and connections between others arise.

This is one of the essential features of the proposed approach, which distinguishes it from the use of various other networks. Its other essential feature is to calculate the degree of coincidence of the object being recognized with those ones for which the network was trained.

A trained logic-predicate network quickly recognizes objects with descriptions that are isomorphic to the ones on which it was trained. But its training and retraining processes remain to be NP-hard. This corresponds to the process of teaching a person. But in some cases, it is not necessary to accurately recognize an object or process, but to find out what it looks like and to what extent.

To answer such a question, the concept of a fuzzy logic-predicate network was proposed. The structure of its cell, shown in Figure 13, is quite complex and contains a partial sequence check. Such verification is NP-hard, but after repeated retraining of the network, the formulas $P^l_i(\overline{x}^l_i)$, setting intermediate goal formulas, are quite short and have a small number of essential arguments, as it was shown by the example in section 7. This means that an exponent of the computational complexity of the partial sequence check is a fairly small value.

Special importance has the choice of "good" initial features. As already mentioned in Subsection 5.4, the predicates W, Y and T, which were used to describe contour images in [3], do not give a good decreasing of computational complexity for the logic-predicate network, constructed from them. Apparently, this is due to the fact that they reflect the perception of a person and are essentially composite.

9 Conclusion

In development of the proposed approach, many-valued predicates may be considered. Especially this relates to the properties of an object elements. Such multi-valued predicates $p(x) = a$ with $a \in D$ can be interpreted as binary ones with a numerical argument: $q(x; a) \Leftrightarrow p(x) = a$. The inclusion of such predicates in the formulation of the problem may serve as the subject of further research.

It is also planned to consider the possibility of introducing cell weights in a logic-predicate network. Such weights can be determined depending on the frequency with which parts of an object description satisfy the formula checked in the corresponding cell. Other weighting strategies are possible, similar to strategies for determining the cell weights of a classical artificial neural network.

In addition, if in the problem setting global features of the object that characterize the entire object are considered, then such an object is usually described by a binary or multi-valued string. Class descriptions can be presented as a propositional formula in disjunctive normal form. Extraction the "frequently encountered" sub-formulas of such propositional formulas [24] allows to form a level description of classes. Using this description in a manner similar to that described in this article, it is possible to build a logic-algebraic network, which changes its configuration in the process of learning. This corresponds to a change of connections in biological neural networks.

Acknowledgments The author is grateful to Vakhtang Kvaratskhelia from Muskhelishvili Institute of Computational Mathematics of the Georgian Technical University, whose question at the conference in 2017 allowed this paper to be born. The author thanks the anonymous referees for their useful suggestions.

References

[1] T. Kosovskaya . "Implementation of Formula Partial Sequence for Rough Solution of AI Problems in the Framework of the Logic-Predicate Approach" in 2019 Computer Science and Information Technologies (CSIT),Yerevan, Armenia, 2019. https://doi.org/10.1109/CSITechnol.2019.8895153

[2] N. J. Nilson, Problem-solving methods in Artificial Intelligence, McGraw-Hill Book Company, 1971.

[3] R. Duda, P. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, 1973.

[4] E.B. Hunt, Artificial Intelligence, Academic Press, 1975.

[5] M.R. Garey, D.S Jonson, Computers and Intractability: A Gide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.

[6] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Third edition, Prentice Hall Press Upper Saddle River, 2009.

[7] T.M. Kosovskaya. "Estimating the number of steps that it takes to solve some problems of pattern recognition which admit logical description" in VESTNIK ST PETERSBURG UNIVERSITY-MATHEMATICS, **40**(4), 82–90, 2007. (In Russian) https://doi.org/10.3103/S1063454107040061

[8] T.M. Kosovskaya, "Multilevel descriptions of classes for decreasing of step number of solving of a pattern recognition problem described by predicate calculus formulas" VESTNIK SANKT-PETERBURGSKOGO UNIVERSITETA SERIYA 10 PRIKLADNAYA MATEMATIKA INFORMATIKA PROTSESSY, **1**, 64–72 (2008). (In Russian)

[9] T.M. Kosovskaya, "Partial deduction of predicate formula as an instrument for recognition of an object with incomplete description" VESTNIK SANKT-PETERBURGSKOGO UNIVERSITETA SERIYA 10 PRIKLADNAYA MATEMATIKA INFORMATIKA PROTSESSY, **1**, 74–84, 2009. (In Russian) https://doi.org/10.24412/FgeTsEutG6w

[10] T.M. Kosovskaya, "Partial Deduction in Predicate Calculus as a Tool for Artificial Intelligence Problem Complexity Decreasing" in 2015 IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS 2015, 73–76, 2016. https://doi.org/10.1109/IntelCIS.2015.7397199

[11] J. Schmidhuber, "Deep learning in neural networks: An overview" Neural Networks, **61**, 85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003

[12] T. Chunwei, X. Yong, Z. Wangmeng, "Image denoising using deep CNN with batch renormalization" Neural Networks, **121**, 461–473, 2020. https://doi.org/10.1016/j.neunet.2019.08.022

[13] E.H. Dawn, C. J. Lakhmi (Eds), Innovations in Bayesian Networks. Theory and Applications, Springer-Verlag, 2008.https://doi.org/10.1007/978-3-540-85066-3

[14] R. Agrahari, A. Foroushani, T.R. Docking et al. "Applications of Bayesian network models in predicting types of hematological malignancies", Sci Rep 8, 6951 (2018). https://doi.org/10.1038/s41598-018-24758-5

[15] A. Tulupiev, C. Nikolenko, A. Sirotkin, Bayesian Networks: logic-probabilistic approach, Nauka, 2006. (in Russian)

[16] A. Shahzad, R. Asif, A. Zeeshan, N. Hamad, A. Tauqeer, "Underwater Resurrection Routing Synergy using Astucious Energy Pods" Journal of Robotics and Control (JRC), **1**(5), 173–184, 2020. https://doi.org/10.18196/jrc.1535

[17] T. Kosovskaya, "Self-modificated predicate networks" International Journal on Information Theory and Applications, **22**(3), 245–257, 2015.

[18] T. Kosovskaya, "Fuzzy logic-predicate network" in Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019), Part of series ASUM (Atlanties Studies in Uncertainty Modeling), **1**, 9–13, 2019. https://doi.org/10.2991/eusflat-19.2019.2

[19] T. Kosovskaya, "Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity", 1–20. 2018. Chapter 3 in "Intelligent System", open access peer-reviewed Edited volume. Edited by Chatchawal Wongchoosuk Kasetsart University https://doi.org/10.5772/intechopen.72765

[20] T. M. Kosovskaya, N. N. Kosovskii, "Polynomial Equivalence of the Problems Predicate Formulas Isomorphism and Graph Isomorphism" VESTNIK ST PETERSBURG UNIVERSITY-MATHEMATICS, **52**(3), 286–292, 2019. https://doi.org/10.1134/S1063454119030105

[21] T.M. Kosovskaya, "Some artificial intelligence problems permitting formalization by means of predicate calculus language and upper bounds of their solution steps" SPIIRAS Proceedings, **14**, 58–75, 2010. (In Russian) https://doi.org/10.15622/sp.14.4

[22] T. M. Kosovskaya, D. A. Petrov, "Extraction of a maximal common sub-formula of predicate formulas for the solving of some Artificial Intelligence problems" VESTNIK SANKT-PETERBURGSKOGO UNIVERSITETA SERIYA 10 PRIKLADNAYA MATEMATIKA INFORMATIKA PROTSESSY, **3**, 250–263, 2017. (In Russian) https://doi.org/10.21638/11701/spbu10.2017.303

[23] T. Kosovskaya, "Construction of Class Level Description for Efficient Recognition of a Complex Object" International Journal "Information Content and Processing", **1**(1), 92–99, 2014.

[24] D.A. Koba, "Algorithm for constructing a multi-level description in pattern recognition problems and estimating the number of its run steps" in Proceedings of the XVII International School-Seminar "Synthesis and Complexity of Control Systems" named after Acad. O.B. Lupanov (Novosibirsk, October 27 - November 1, 2008), Publishing House of the Institute of Mathematics, Novosibirsk, 55–59, 2008. (In Russian)