Chapter 3

Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity

Tatiana Kosovskaya

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72765

Abstract

The chapter is devoted to the use of predicate calculus for artificial intelligence (AI) problem solving. Here, an investigated object is represented as a set of its elements and is characterized by a fixed number of predicates. Its description is a set of all constant literals (with the chosen predicates), which are valid on the object. The NP-complete problem, "whether an object satisfies a goal formula," is under consideration. The upper bound of number of its solution steps is exponential. The notion of common up to the names of arguments subformula of two predicate formulas and one of their isomorphisms allows to construct a level description of the set of goal formulas and essentially to decrease the upper bounds of the problem solving. The level description permits to define a self-training predicate network, which may change its configuration during the process of training. The extraction of common up to the names of arguments subformulas permits to construct a multiagent description of an object when every agent does not know the true number of the object elements and uses her own notifications for the names of elements. A model example illustrating all algorithms is presented.

Keywords: predicate calculus, NP-completeness, level description, predicate network, multi-agent description

1. Introduction

The choice of initial attributes for description of an object in an artificial intelligence (AI) problem is the first stage of any simulation of an informational process (representation of information for its further use).

At the 60–70th of the twentieth century, many authors (see, for example, [1]) offered to use predicate calculus for AI problem solving. The resolution method seemed to be a very easy and clear tool to solve problems dealing with compound objects, which can be described by properties of its elements and relations between these elements.

Until the notion of NP-complete problem (in particular, described in [2, 3]) was not widely adopted, such an approach seemed to be very convenient, but many such-a-way formalized problems occurred to be NP-complete or even algorithmic unsolvable.

While developing the effective algorithms deciding discrete problems, determination of estimations for number of steps of their run becomes one of the important problems. The absence of the proved estimations for number of an algorithm run steps is considered as an insufficient research of this algorithm. It is especially relevant for problems with big input. It concerns, in particular, to the algorithms deciding various AI problems. At practical use of an algorithm, it is important that it has polynomial upper bound of number of its run steps. The NP-completeness or NP-hardness of a problem means now that the polynomial algorithm of its decision is not known.

In 2007, the author proved NP-completeness of a series of AI problems formalized with the help of predicate calculus formulas [4], proved upper bounds for number of steps of algorithms solving these problems [5], and offered a level description of goal formulas for decreasing the number of proof steps [6]. Such a level description is based on the extraction of a common up to the names of its arguments sub-formula of the set of elementary conjunctions of atomic predicate formulas. These sub-formulas define generalized characteristics of an object.

Extraction of such sub-formulas allows to construct logic-predicate networks [7], which may change its configuration (the number of layers and the number of cells in the layer) during the process of training.

Extraction of these sub-formulas may serve as an instrument for constructing a multi-agent description of an object, when every agent can describe only a part of the object (these parts are intersected), but every agent gives its own names to the elements of the whole object [8].

Here, some AI problems formalized in such a way are under consideration. For these problems, the solving algorithms and upper bounds of their run are obtained. These upper bounds permit to point out the parameters of the problem, which mostly influence on the complexity of the algorithm, and to offer approaches permitting to decrease the complexity.

A model example illustrating the described approach and algorithms is given.

2. Logic-predicate approach to some AI problems and number of steps of these problems solution

Let an investigated object be presented as a set of its elements $\omega = \{\omega_1,\ldots, \omega_t\}$. The set of predicates $p_1,\ldots, p_n$ (every of which is defined on the elements of ω) characterizes properties of these elements or relations between them. Logical description $S(\omega)$ of an object ω is a collection of all true formulas in the form $p_i(\overline{\tau})$ or $\neg p_i(\overline{\tau})$ (where $\overline{\tau}$ is an ordered subset of ω) describing the properties of ω elements or relations between them.

Let the set Ω of all investigated objects be a union of classes Ω_k, ($k = 1,\ldots, K$), i.e., $\Omega = \bigcup_{k=1}^{K}\Omega_k$. Logical description of the class Ω_k is such a formula $A_k(\overline{x})$ 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.

Here and below, the notation $\overline{x}$ is used for an ordered list of the set x. To denote that there exists such a list $\overline{x}$ that all values for variables from the list $\overline{x}$ are distinct the notation $\exists\overline{x}_{\neq}A_k(\overline{x})$ is used.

The introduced descriptions allow to solve many artificial intelligence problems [9]. Main of these problems may be formulated as follows.

Identification problem: to pick out all parts of the object ω that belongs to the class Ω_k.

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

Analysis problem: to find and classify all parts τ of the object ω.

The solution of these problems may be reduced to the proof of logic sequents

$$S(\omega) \Rightarrow \exists\overline{x}_{\neq}A_k(\overline{x}), \tag{1}$$

$$S(\omega) \Rightarrow \bigvee_{k=1}^{K}A_k(\overline{x}), \tag{2}$$

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

respectively, and determination of the values for $\overline{x}$ and k. The number of $A_k(\overline{x})$ variables in the sequent (2) must be equal to the number of constants in ω.

Note that the proof of any of the sequent (1), (2), or (3) answers only the question "whether it is true?" Strictly speaking, in the sequents (1)–(3), instead of the symbols $\exists\overline{x}$ and $\bigvee_{k=1}^{K}$ there must be words "what are the distinct values of $\overline{x}$" denoted as $(?\overline{x})$ and "what are the values of k?" denoted as $?_{k=1}^{K}$, respectively. In such a case, the sequents (1)–(3) would take the form

$$S(\omega) \Rightarrow (?\overline{x}) \neq A_k(\overline{x}), \tag{4}$$

$$S(\omega) \Rightarrow ?_{k=1}^{K}A_k(\overline{x}), \tag{5}$$

$$S(\omega) \Rightarrow ?_{k=1}^{K}(?\overline{x}) \neq A_k(\overline{x}). \tag{6}$$

If one uses an exhaustive or a logical algorithm (derivation in a sequent calculus or proof by resolution method), the algorithm gives the values for $\overline{x}$ and k.

The proof of sequents (1) and (3) is based on the proof of the sequent

$$\mathrm{S}(\omega) \Rightarrow \exists\overline{x}_{\neq} A(\overline{x}), \tag{7}$$

where $A(\overline{x})$ is an elementary conjunction. It follows from the fact that $A_k(\overline{x})$ is a disjunction of the form $C_1\vee\ldots\vee C_r$ of elementary conjunctions of atomic formulas $C_1,\ldots,C_r$, and $\exists\overline{x}_{\neq}$ $(C_1\vee\ldots\vee C_r)$ $\Leftrightarrow$ $(\exists\overline{x}_{\neq}$ $C_1\vee\ldots\vee$ $\exists\overline{x}_{\neq}$ $C_r)$. That is why we can consecutively check the sequents of the form

$S(\omega) \Rightarrow \exists \overline{x}_{\neq}\ C_j$. Here, the maximal value for j is r for the sequent (1), and the sum of the number of elementary conjunctions in all class descriptions for the sequent (3).

An **exhaustive algorithm** is widespread to prove (4). The total estimate for the number of steps (i.e., the number of comparisons) for the exhaustive algorithm solving (4) is

$$t(t\text{–}1)\ldots(t\text{–}m+1)\sum_{i=1}^{n} a_i s_i \tag{8}$$

or, more roughly,

$$O(t^m as). \tag{9}$$

While using a **logical algorithm** (derivation in a predicate sequent calculus or proof by resolution method for predicate calculus), one must find unifier of the formula $A(\overline{x})$ and some subset of $S(\omega)$.

The number of steps (i.e., the number of comparisons) required for the solution of the system and, hence, for the logical algorithm solving (4) is

$$O\left(s_1^{a_1}\ldots s_n^{a_n}\right), \tag{10}$$

a_i and s_i be the numbers of literals with the predicate p_i in $A(\overline{x})$ and in $S(\omega)$, respectively. More roughly

$$O\left(s'^{a}\right), \tag{11}$$

where $s' = \max\{s_1,\ldots, s_n\}$.

The above-received estimations are exponential over the length of $A(\overline{x})$. The ones for an exhaustive algorithm are exponential over the number of variables, and for a logical algorithm they are exponential over the maximal number of literals with the same predicate. It allows to choose the algorithm depending on characteristics of the concrete problem under consideration. Note that the reverse Maslov's method [10, 11] has the same estimations for the solution of the sequent (4), but makes essentially smaller number of steps on the average.

The received estimations cannot be essentially decreased up to polynomial ones if **P** ≠ **NP** (classes **P** and **NP** are the classes of predicates checked in polynomial time by a deterministic or nondeterministic Turing machine respectively). More precisely, the problem (4) is NP-complete and, hence, the problems (1) and (3) are NP-complete, and the problems (4) and (5) are NP-hard [4, 5].

Problem (2) is strictly connected with the so-called "open" problem ISOMORPHISM OF GRAPHS [3], for which it is not proved neither its polynomiality nor its NP-completeness.

http://dx.doi.org/10.5772/intechopen.72765

2.1. Model example of description and the estimations for the number of an algorithm steps

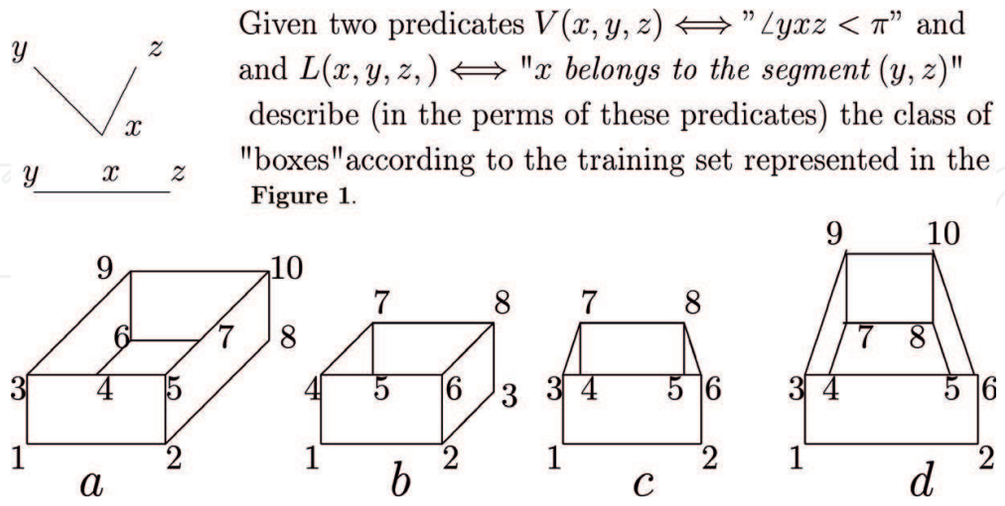
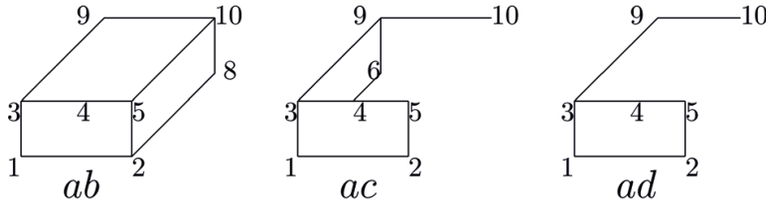


Figure 1. Standard different contour images of a "box".



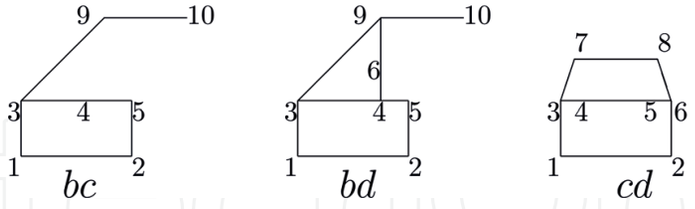


Figure 2. Images corresponding to extraction of common sub-formulas.

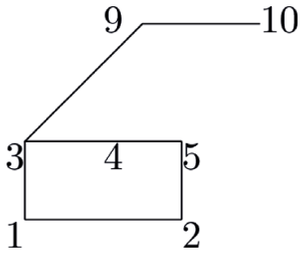


Figure 3. Image corresponding to the second extraction of common sub-formulas.

These standard images allow to form a description (up to mirror image) of almost all boxes. Such a description is a disjunction of four elementary conjunctions containing, respectively, 10, 8, 10, 8 variables and 30 + 2, 23 + 1, 28 + 4, 33 + 4 atomic formulas with predicates V and L, respectively. The elementary conjunctions corresponding to the images are

$$A_{\mathrm{a}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10}) = V(x_1,x_3,x_2)\&V(x_2,x_1,x_5)\&V(x_2,x_5,x_8)\&V(x_3,x_4,x_1)\&$$
$$V(x_3,x_5,x_1)\&V(x_3,x_9,x_4)\&V(x_3,x_9,x_5)\&V(x_3,x_9,x_1)\&V(x_4,x_3,x_6)\&V(x_4,x_6,x_5)\&$$
$$V(x_5,x_2,x_3)\&V(x_5,x_2,x_4)\&V(x_5,x_3,x_7)\&V(x_5,x_3,x_{10})\&V(x_5,x_4,x_7)\&V(x_5,x_4,x_{10})\&$$
$$V(x_5,x_7,x_2)\&V(x_5,x_{10},x_2)\&V(x_6,x_4,x_9)\&V(x_6,x_7,x_4)\&V(x_6,x_9,x_7)\&V(x_7,x_5,x_6)\&$$
$$V(x_7,x_6,x_{10})\&V(x_8,x_2,x_{10})\&V(x_9,x_6,x_3)\&V(x_9,x_{10},x_6)\&V(x_9,x_{10},x_3)\&V(x_{10},x_7,x_9)\&$$
$$V(x_{10},x_8,x_7)\&V(x_{10},x_8,x_9)\&L(x_4,x_3,x_5)\&L(x_7,x_5,x_{10}),$$

$$A_{\mathrm{b}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = V(x_1,x_4,x_2)\&V(x_2,x_1,x_6)\&V(x_2,x_6,x_3)\&V(x_2,x_1,x_3)\&$$
$$V(x_3,x_2,x_8)\&V(x_4,x_5,x_1)\&V(x_4,x_6,x_1)\&V(x_4,x_7,x_5)\&V(x_4,x_7,x_6)\&V(x_4,x_7,x_1)\&$$
$$V(x_5,x_4,x_7)\&V(x_5,x_7,x_6)\&V(x_6,x_2,x_5)\&V(x_6,x_2,x_4)\&V(x_6,x_5,x_8)\&V(x_6,x_4,x_8)\&$$
$$V(x_6,x_8,x_2)\&V(x_7,x_5,x_4)\&V(x_7,x_8,x_5)\&V(x_7,x_8,x_4\&V(x_8,x_3,x_6)\&V(x_8,x_6,x_7)\&$$
$$V(x_8,x_3,x_7)\&L(x_5,x_4,x_6),$$

$$A_{\mathrm{c}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = V(x_1,x_3,x_2)\&V(x_2,x_1,x_6)\&V(x_3,x_4,x_1)\&V(x_3,x_5,x_1)\&$$
$$V(x_3,x_6,x_1)\&V(x_3,x_7,x_4)\&V(x_3,x_7,x_5)\&V(x_3,x_7,x_6)\&V(x_3,x_7,x_1)\&V(x_4,x_3,x_7)\&$$
$$V(x_4,x_7,x_5)\&V(x_4,x_7,x_6)\&V(x_5,x_3,x_8)\&V(x_5,x_4,x_8)\&V(x_5,x_8,x_6)\&V(x_6,x_2,x_5)\&$$
$$V(x_6,x_2,x_4)\&V(x_6,x_2,x_3)\&V(x_6,x_5,x_8)\&V(x_6,x_4,x_8)\&V(x_6,x_3,x_8)\&V(x_6,x_2,x_8)\&$$
$$V(x_7,x_8,x_4)\&V(x_7,x_8,x_3)\&V(x_7,x_4,x_3)\&V(x_8,x_6,x_5)\&V(x_8,x_5,x_7)\&V(x_8,x_6,x_7)\&$$
$$L(x_4,x_3,x_5)\&L(x_4,x_3,x_6)\&L(x_5,x_4,x_6)\&L(x_5,x_3,x_6),$$

$$A_{\mathrm{d}}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10}) = V(x_1,x_3,x_2)\&V(x_2,x_1,x_6)\&V(x_3,x_4,x_1)\&$$
$$V(x_3,x_5,x_1)\&V(x_3,x_6,x_1)\&V(x_3,x_9,x_4)\&V(x_3,x_9,x_5)\&V(x_3,x_9,x_6)\&V(x_3,x_9,x_1)\&$$
$$V(x_4,x_3,x_7)\&V(x_4,x_7,x_5)\&\&V(x_4,x_7,x_6)\&V(x_5,x_4,x_8)\&V(x_5,x_3,x_8)\&V(x_5,x_8,x_6)\&$$
$$V(x_6,x_2,x_5)\&V(x_6,x_2,x_4)\&V(x_6,x_2,x_3)\&V(x_6,x_5,x_{10})\&V(x_6,x_4,x_{10})\&V(x_6,x_3,x_{10})\&$$
$$V(x_7,x_4,x_9)\&V(x_7,x_8,x_4)\&V(x_7,x_9,x_8)\&V(x_8,x_5,x_7)\&V(x_8,x_7,x_{10})\&V(x_8,x_{10},x_5)\&\&$$
$$V(x_9,x_7,x_3)\&V(x_9,x_{10},x_3)\&V(x_9,x_{10},x_7)\&V(x_{10},x_6,x_8)\&V(x_{10},x_8,x_9)\&V(x_{10},x_6,x_9)\&$$
$$L(x_4,x_3,x_5)\&L(x_4,x_3,x_6)\&L(x_5,x_4,x_6)\&L(x_5,x_3,x_6).$$

Given a "box" inside a complex contour image containing t nodes and s be the maximal number of occurrences of the predicate V in the description $S(\omega)$, it would be recognized (according to the estimations (5′) and (6′)) in $O(t^{\mathbf{10}})$ steps by an exhaustive algorithm and in $O(s^{\mathbf{37}})$ steps by a logical algorithm.

Can seem that many atomic formulas such as $V(x_{\mathbf{5}},x_{\mathbf{2}},x_4)$, $V(x_{\mathbf{5}},x_3,x_{\mathbf{7}})$, $V(x_{\mathbf{5}},x_4,x_{\mathbf{7}})$, and $V(x_{\mathbf{5}},x_{\mathbf{7}},x_2)$ in $V(x_{\mathbf{5}},x_{\mathbf{2}},x_3)$ & $V(x_{\mathbf{5}},x_{\mathbf{2}},x_4)$ & $V(x_{\mathbf{5}},x_3,x_{\mathbf{7}})$ & $V(x_{\mathbf{5}},x_3,x_{\mathbf{10}})$ & $V(x_{\mathbf{5}},x_4,x_{\mathbf{7}})$ & $V(x_{\mathbf{5}},x_4,x_{\mathbf{10}})$ & $V(x_{\mathbf{5}},x_{\mathbf{7}},x_2)$ & $V(x_{\mathbf{5}},x_{\mathbf{10}},x_2)$ are unnecessary. But if we delete such "unnecessary" formulas, it would be needed to add to a premise of a sequent, a condition that every point that belongs to a segment (y,z) may be substituted instead of y or z (be the second or the third argument) in every atomic

formula with the predicate V. It would be another setting of a problem. Moreover, such "unnecessary" formulas can help to decrease the number of algorithm run steps if we use branch and bound algorithm inside the exhaustive algorithm or the reverse Maslov's method for a logical one [11].

3. Level description of classes

Below, the designation $A_k(\overline{x}_k)$ will be used for elementary conjunctions, which are disjunctive terms of a class description.

The notion of level description of classes was introduced in [6]. Such a description essentially allows to decrease the number of steps for an algorithm solving every of the above-formulated problems. This notion is based on the extraction of "frequently" appeared "sub-formulas" $P_i^1(\overline{y}_i^1)$ ($i = 1,\ldots, n_1$) of $A_1(\overline{x}_1),\ldots, A_K(\overline{x}_K)$ with "small complexity" and changing them in these formulas by atomic formulas $p_i^1(y_i^1)$ defined by an equivalence of the form $p_i^1(y_i^1) \Leftrightarrow P_i^1(\overline{y}_i^1)$. New predicates p_i^1 having new first-level arguments y_i^1 for lists $\overline{y}_i^1$ of initial variables are called first-level predicates. The formula $A_k^1(\overline{x}_k^1)$ is received from $A_k(\overline{x}_k)$ by means of a substitution of $p_i^1(y_i^1)$ instead of $P_i^1(\overline{y}_i^1)$

Repeat the above-described procedure with all formulas $A_k^1(\overline{x}_k^1)$. After L repetitions, an L-level description in the following form is received:

$$\begin{cases} A_k^L(\overline{x}^L) \\ p_1^1(y_1^1) \Leftrightarrow P_1^1(\overline{y}_1^1) \\ \vdots \\ p_{n_1}^1\left(y_{n_1}^1\right) \Leftrightarrow P_{n_1}^1\left(\overline{y}_{n_1}^1\right) \\ \vdots \\ p_i^l(y_i^l) \Leftrightarrow P_i^l(\overline{y}_i^l) \\ \vdots \\ p_{n_L}^L\left(y_{n_L}^L\right) \Leftrightarrow P_{n_L}^L\left(\overline{y}_{n_L}^L\right) \end{cases} \tag{12}$$

The solution of the problem of the form (4) with the use of the level description of classes is decomposed on the sequential ($l = 1, \ldots, L$) implementation of the actions 1–4:

1. For every i ($i = 1,\ldots, n_l$) check $S^{l-1}(\omega) \Rightarrow \exists\overline{y}_i^1 \neq P_i^1(\overline{y}_i^1)$ and find all lists $\overline{\tau}_i^1$ of previous levels constants for the values of the variable list y_i^1 such that $S^{l-\mathbf{1}}(\omega) \Rightarrow P_i^1(\overline{\tau}_i^1)$.

2. Introduce new l-level atomic formulas $p_i^1(y_i^1)$ defined by the equalities $p_i^1(y_i^1) \Leftrightarrow P_i^1(\overline{y}_i^1)$ with new l-level variables.

3. Substitute $p_i^1(y_i^1)$ instead of $P_i^1(\overline{y}_i^1)$ into $A_k^{l-1}(\overline{y}_k^{l-1})$ and obtain $A_k^l(\overline{y}_k^l)$.

4. Add all constant atomic *l*-level formulas in the form $p_i^1(\tau_i^1)$ (τ_i^1 were received at the first step) to $S^{l-1}(\omega)$ and obtain $S^l(\omega)$. Here $\tau_i^1\ \tau_i^l$ are new *l*-level constants for the lists of $(l-1)$-level constants

5. At last check $S^L(\omega) \Rightarrow \exists \overline{y}_k^L \neq A_k^L(\overline{y}_k^L)$.

The decreasing of the number of steps for an algorithm solving every of the above formulated problems (1)–(3) with the use of a level description follows from the fact that in items 1, 2, and 5, we solve the same problem as it was formulated in Section 1 and has the number (4). The estimations of number of steps exponentially depend on the parameters of the formula, i.e., on the right part of implication. That is why the term "small complexity" for $P_i^l(\overline{y}_i^l)$ must be interpreted as "small number of variables in $P_i^l(\overline{y}_i^l)$" for an exhaustive algorithm, and "small number of literals in $P_i^l(\overline{y}_i^l)$" for a logical algorithm

Why did we use quotation marks for the term "sub-formulas?" Such formulas (elementary conjunctions) $P_j^l\left(\overline{y}_j^l\right)$ are not obliged to be precisely sub-formulas of $A_1(\overline{x}_1), \ldots, A_K(\overline{x}_K)$ but may differ from these sub-formulas in names of variables and order of conjunctive terms

Definition 1. *Elementary conjunctions P and Q are called isomorphic if there is an elementary conjunction R and substitutions $\lambda_{R,P}$ and $\lambda_{R,Q}$ of the arguments of P and Q, respectively, instead of the variables in R such that the results of these substitutions coincide up to the order of literals*

The substitutions $\lambda_{R,P}$ and $\lambda_{R,Q}$ are called unifiers of R with P and Q, respectively.

Definition 2. *Elementary conjunction C is called a common up to the names of arguments sub-formula of two elementary conjunctions A and B if it is isomorphic to some sub-formulas A′ and B′ of A and B, respectively*

For example, let $A(x,y,z) = p_1(x)\ \&\ p_1(y)\ \&\ p_1(z)\ \&\ p_2(x, y)\ \&\ p_3(x, z)$, $B(x,y,z) = p_1(x)\ \&\ p_1(y)\ \&\ p_1(z)\ \&\ p_2(x, z)\ \&\ p_3(x, z)$

Is the formula $P(u,v) = p_1(u)\ \&\ p_1(v)\ \&\ p_2(u, v)$ their common sub-formula?

The formula $P(u,v)$ is their common up to the names of variables sub-formula with the unifiers $\lambda_{P,A}$—substitution of x and y instead of u and v, respectively, and $\lambda_{P,B}$—substitution of x and z instead of u and v, respectively. It is so 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 $B(x,y,z)$.

An algorithm of extraction of a maximal (having a maximal number of literals) common up to the names of arguments sub-formula *C* of two elementary conjunctions *A* and *B* and determining the unifiers $\lambda_{c,A'}$ and $\lambda_{c,B'}$ is described in [12]. The number of steps of this algorithm is $O(a^a b^b)$, where *a* and *b* are the numbers of literals in *A* and *B*, respectively. The minimal number of steps of this algorithm is $O((ab)^2)$, the middle estimate is $O((ab)^{1/2\ \log(ab)})$.

This algorithm allows to construct a level description for a set of goal elementary conjunctions. Essential difference between maximal common up to the names of arguments sub-formulas and sub-formulas in the level description consists in the fact that in the level description it is

needed to extract sub-formulas with "small complexity" but not a maximal one. An algorithm of level description construction is in [6]. It consists in sequential pairwise extraction of common up to the names of variables sub-formulas of $A_i(\overline{x}_i)$ and $A_j(\overline{x}_j)$ with finding their unifiers and then the analogous procedure with the obtained sub-formulas.

Let N be the maximal number of literals $A_k(\overline{x}_k)$ in (k = 1,…, K). The upper bound of this algorithm number of steps is $O(K^2N^{2N})$.

3.1. Example of sub-formula extraction and a level description construction

Return to the example in the previous section. There, we have seen a description of a class of "boxes" represented in **Figure 1**. According to these descriptions, we have received that given a "box" inside a complex contour image containing t nodes it would be recognized in $O(t^{10})$ steps by an exhaustive algorithm and in $O(s^{37})$ steps by a logical algorithm (here, s is the maximal number of occurrences of the same predicate in the object description $S(\omega)$).

Pairwise extraction of common up to the names of variables of elementary conjunctions, corresponding to these images, allows to extract common up to the names of variables sub-formulas corresponding to the images represented in **Figure 2**

These sub-formulas contain, respectively, 8, 8, 7, 7, 7, 8 variables and 18, 15, 11, 11, 15, 16 atomic formulas.

The following extraction by means of pairwise partial deduction between common sub-formulas corresponding to images *ab*, *ac*, *ad*, *bc*, *bd*, *cd* gives a sub-formula corresponding to the image represented in **Figure 3**.

Elementary conjunction $P^1(x_1,x_2,x_3,x_4,x_5,x_9,x_{10}) = V(x_1,x_3,x_2)$ & $V(x_2,x_1,x_5)$ & $V(x_3,x_4,x_1)$ & $V(x_3,x_5,x_1)$ & $V(x_3,x_9,x_4)$ & $V(x_3,x_9,x_5)$ & $V(x_3,x_9,x_1)$ & $V(x_5,x_2,x_4)$ & $V(x_5,x_2,x_3)$ & $V(x_9,x_{10},x_3)$ & $T(x_4,x_3,x_5)$, corresponding to this image, defines a first-level predicate $p^1(x^1)$. The first-level variable x^1 is a variable for a list of seven initial variables $x^1 = (x_1,x_2,x_3,x_4,x_5,x_9,x_{10})$. The unifier of $P^1(x_1,x_2,x_3,x_4,x_5,x_9,x_{10})$ with the description of ab, ac, ad, ac, and bd is an identical substitution, but its unifier with the description of cd is a substitution of x_6 instead of x_5.

Elementary conjunctions $P_1^2(x^1,x_1,x_2,x_3,x_4,x_5,x_8,x_9,x_{10})$, $P_2^2(x^1,x_4,x_5,x_6,x_9,x_{10})$, $P_3^2(x^1,x_3,x_4,x_5,x_{10})$, $P_4^2(x^1,x_2,x_5,x_6,x_{10})$, corresponding to the images *ab*, *ac*, *bd*, *cd* and written with the use of the predicate $p^1(x^1)$, define second-level predicates $p_1^2(x_1^2)$, $p_2^2(x_2^2)$, $p_3^2(x_3^2)$, $p_4^2(x_4^2)$ with the second-level variables $x_1^2 = (x^1,x_1,x_2,x_3,x_4,x_5,x_8,x_9,x_{10})$, $x_2^2 = (x^1,x_4,x_5,x_6,x_9,x_{10})$, $x_3^2 = (x^1,x_3,x_4,x_5,x_{10})$, $x_4^2 = (x^1,x_2,x_5,x_6,x_{10})$.

For example, a sub-formula corresponding to the image ab is $P_1^2(x^1,x_1,x_2,x_3,x_4,x_5,x_8,x_9,x_{10}) = p^1(x^1)$ & $V(x_2,x_5,x_8)$ & $V(x_2,x_1,x_8)$ & $V(x_5,x_4,x_{10})$ & $V(x_5,x_3,x_{10})$ & $V(x_8,x_2,x_{10})$ & $V(x_{10},x_8,x_5)$ & $V(x_{10},x_5,x_9)$ & $V(x_{10},x_8,x_9)$. The unifier of $P_1^2(x^1,x_1,x_2,x_3,x_4,x_5,x_8,x_9,x_{10})$ with the description of a is an identical substitution, and with the description of b, it is a substitution of x_4,x_5,x_6,x_7,x_8 instead of x_2,x_4,x_5,x_9,x_{10}. Descriptions of images c and d are not unified with it.

The three-level description of the image b takes the form $A_b^2(x_1^2,x_4,x_5,x_6,x_7) = p_1^2(x_1^2)$ & $V(x_5,x_4,x_7)$ & $V(x_5,x_7,x_6)$ or $A_b^2(x_3^2,x_4,x_5,x_6,x_7) = p_3^2(x_3^2)$ & $V(x_3,x_2,x_8)$ & $V(x_5,x_4,x_7)$ & $V(x_5,x_7,x_6)$.

Given a "box" inside a complex contour image containing t nodes, the proof the sequence from S(ω) of elementary conjunction $P^{\mathbf{1}}(x_1,x_2,x_3,x_4,x_5,x_{9,}x_{10})$ defining the first-level predicate $p^1(x^1)$ and the denotation of variables $x_1,x_2,x_3,x_4,x_5,x_{9,}x_{10}$ would be done in $O(t^{\mathbf{7}})$ steps by an exhaustive algorithm and in $O(s^{\mathbf{11}})$ steps by a logical algorithm.

Elementary conjunctions $P_1^2(x_1^1)$, $P_2^2(x_1^1)$, $P_3^2(x_1^1)$, $P_4^2(x_1^1)$ contain respectively only 1, 1, 0, 1 "new" variables (not containing in the first-level variables) and 7, 4, 4, 5 "new" atomic formulas. The proof of the sequence from $S^{\mathbf{1}}(\omega\ )$ of these elementary conjunctions defining the second-level predicates $p_1^2(x_1^2)$, $p_2^2(x_2^2)$, $p_3^2(x_3^2)$, $p_4^2(x_4^2)$, and the denotation of the "new" variables would be done in $O(t)$ steps by an exhaustive algorithm and in $O(s^{\mathbf{7}})$ steps by a logical algorithm.

Elementary conjunctions obtained from the class description by means of second-level predicates instead of the corresponding sub-formulas contain respectively 2, 0, 2, 2 "new" variables and 7, 4, 11, 16 "new" atomic formulas. The proof of the sequence from $S^{\mathbf{2}}(\omega\ )$ of these elementary conjunctions and the denotation of the "new" variables would be done in $O(t^{\mathbf{2}})$ steps by an exhaustive algorithm and in $O(s^{\mathbf{16}})$ steps by a logical algorithm.

As $O(t^{\mathbf{7}}) + O(t) + O(t^{\mathbf{2}}) = O(t^{\mathbf{7}}) < O(t^{\mathbf{10}})$ and $O(s^{\mathbf{11}}) + O(s^{\mathbf{7}}) + O(s^{\mathbf{16}}) = O(s^{\mathbf{16}}) < O(s^{\mathbf{37}})$ then both an exhaustive algorithm and a logical algorithm using the built level description of the class of "boxes" make the less number of steps then the same ones using the initial description. At the same time, the decreasing of number of steps of a logical algorithm is more noticeably.

4. Logic-predicate network

Traditional neuron network deals with binary or many-valued characteristics of an object and is an adder of weighted inputs followed by a function mapping the result into the segment [0, 1]. The neuron network configuration is fixed and only the weights may be changed.

A logic-predicate network is described later. The inputs for this network are atomic formulas setting properties of the elements composing an investigated object and relations between them [7]. The proposed model of logic-predicate network has two blocks: a training block and a recognition block. The input of every block is an elementary conjunction of atomic predicate formulas or their negations. Configuration of the recognition block is formed after an implementation of the training block and may be changed with its help.

The training block is a "slowly running" block. At the same time, the recognition block is a "quickly running" one. The base of the proposed predicate network is a logic-objective approach to AI problems and level description of classes.

The scheme of the logic-predicate network is presented in **Figure 4**

At a training stage of logic-predicate network construction, we have a training set of objects. Let a training set of objects $\omega_1,\ldots, \omega_K$ be given to form an initial variant of the network training block. Replace every constant ω_k^j in S(ω_k) by a variable x_k^j ($k = 1,\ldots, K$, $j = 1,\ldots, t_k$) and substitute the sign & between the atomic formulas. Initial goal formulas $A_1(\overline{x}_1), \ldots, A_K(\overline{x}_K)$ are obtained.

http://dx.doi.org/10.5772/intechopen.72765

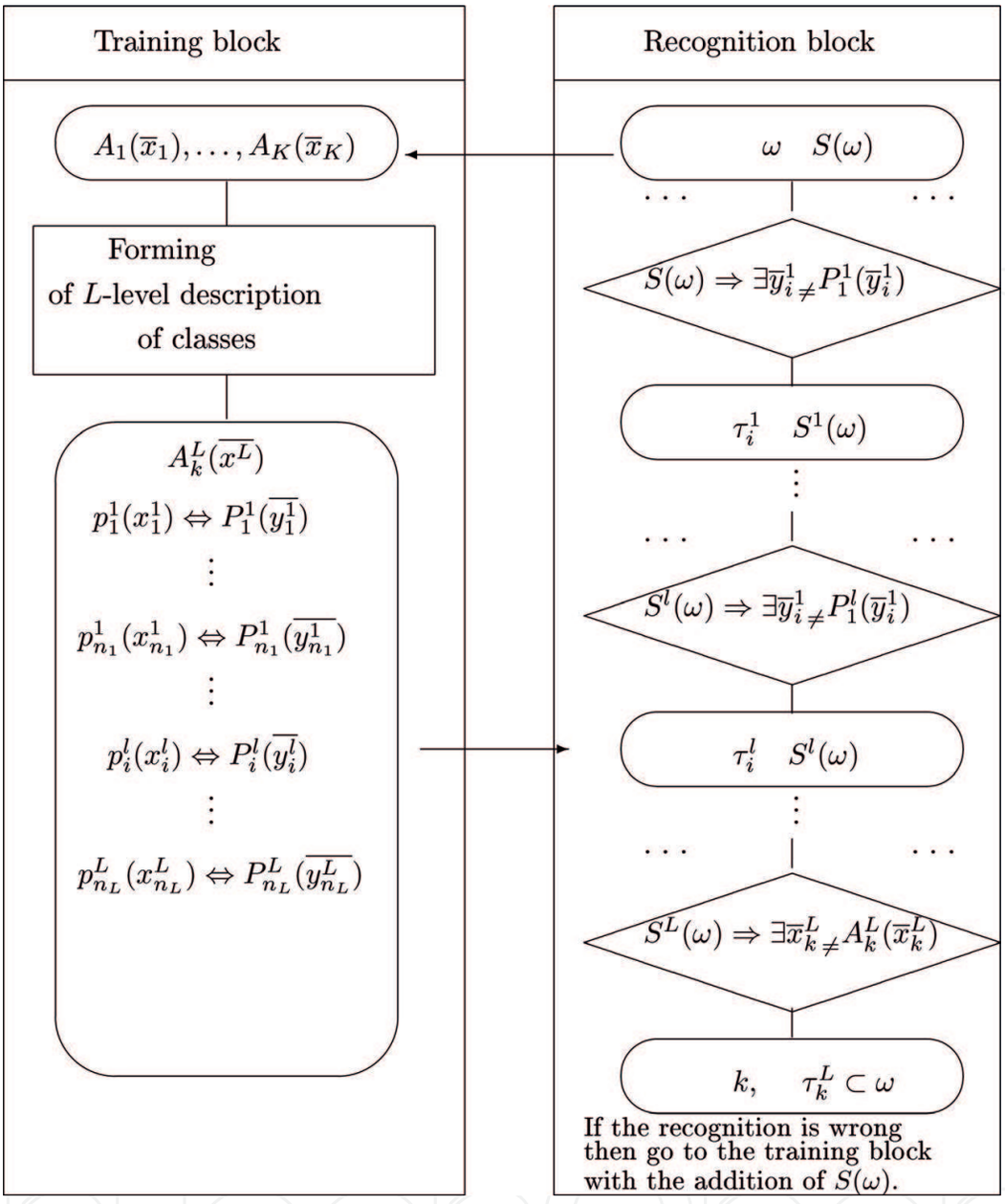


Figure 4. Scheme of the logic-predicate network.

Construct a level description for these goal formulas with the use of algorithm of level description. The first approximation to the recognition block is formed. Formulas $P_i^l(\overline{y}_i^l)$ ($i = 1,\ldots, n_l$, $l = 1,\ldots, L$) obtained in the training block (together with the unifiers) are the contents of the cells forming the recognition block. This block runs as it was described in the section level description of classes.

The recognition block tries to identify a new object according to the level description of classes, obtained in the training block.

If after the "recognition block" run an object is not recognized or has wrong classification, then it is possible to train anew the network. The description of the "wrong" object must be

added to the input set of the training block. The training block extracts common sub-formulas of this description and previously received formulas forming the recognition block. Some sub-formulas in the level description would be changed. Then, the recognition block is reconstructed.

4.1. Model example of a logic-predicate network construction

Given a training set for the class of contour images of "boxes" presented in **Figure 1** (Section 2). Pairwise extraction of common up to the names of variables of elementary conjunctions, corresponding to these images, allows to extract common sub-formulas corresponding to the images presented in **Figures 2** and **3** (Section 3). Fragments of the images corresponding to a three-level network are presented in **Figure 5**.

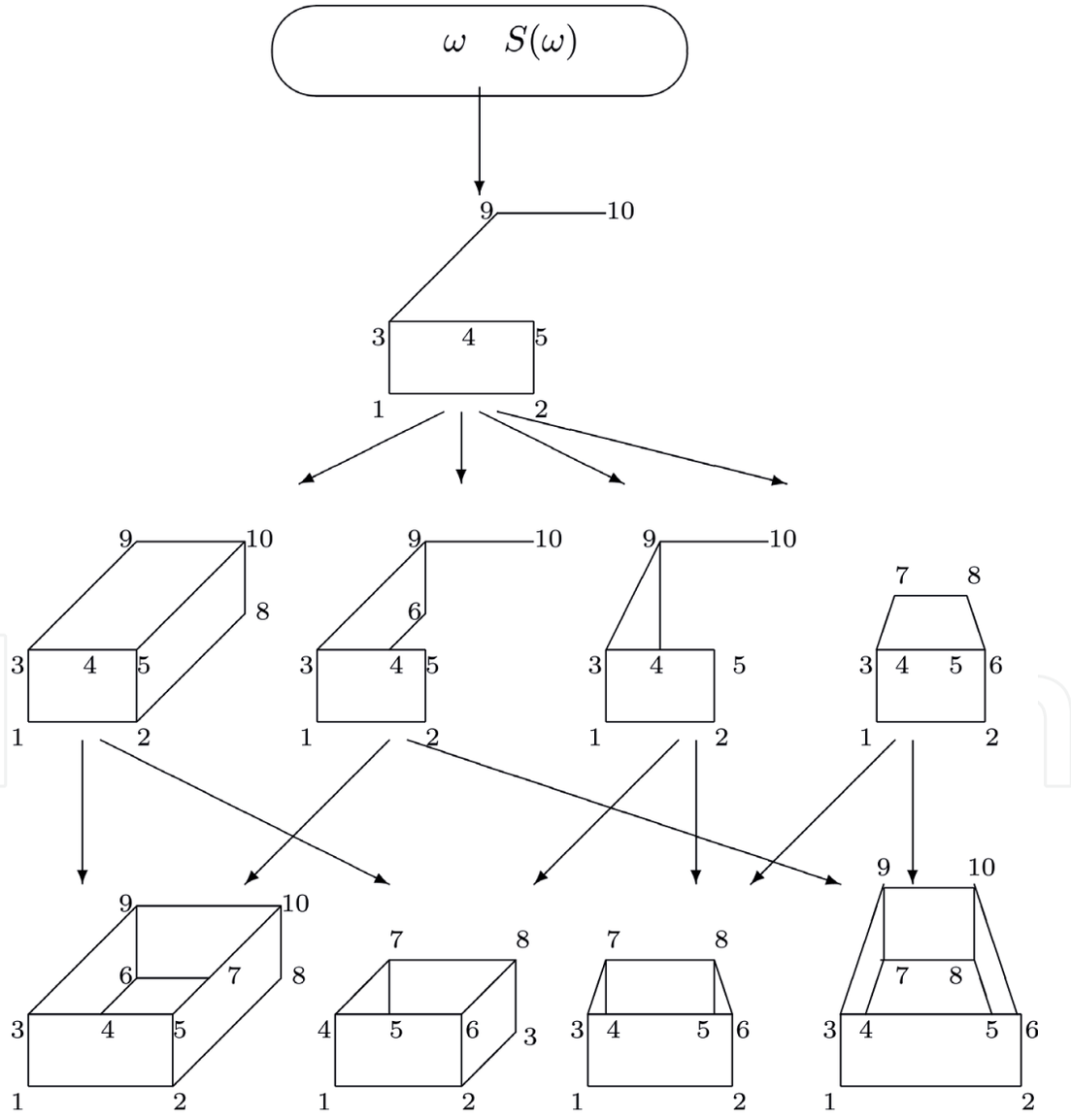


Figure 5. Fragments of the images corresponding to a three-level network.

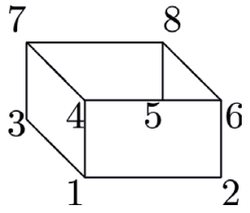


Figure 6. Control image.

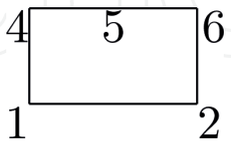


Figure 7. Image corresponding to the new first-level predicate.

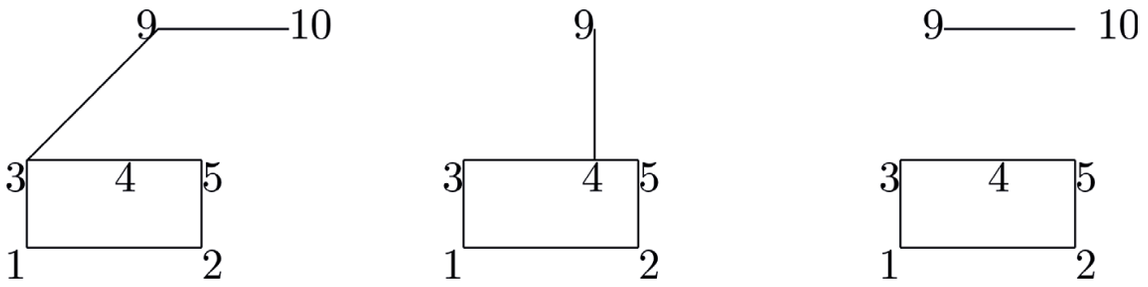


Figure 8. Images corresponding to three new second-level predicates.

Given, a new image represented in **Figure 6** for recognition, the network would not recognize it because the first-level predicate is not valid.

Add the description of this control image to the input data of the training block. The extraction of common sub-formulas for this description and the formula defining the first-level predicate gives a formula corresponding to the image represented in **Figure 7**.

New second-level predicates correspond to three images represented in **Figure 8**.

The set of the third-level predicates coincides with the set of previous second-level predicates. So, the recognition block is constructed anew and represents four-level description of the class. Fragments of the images corresponding to a four-level network are presented in **Figure 9**.

5. Multi-agent description of an object

A problem of multi-agent description of a complex object is under consideration in this section. It is supposed that every agent knows only a part of an investigated object description. Moreover, she does not know the true names of elements and gives them names arbitrary. It is similar to the parable about tree blind men who feel an elephant. To overcome such a

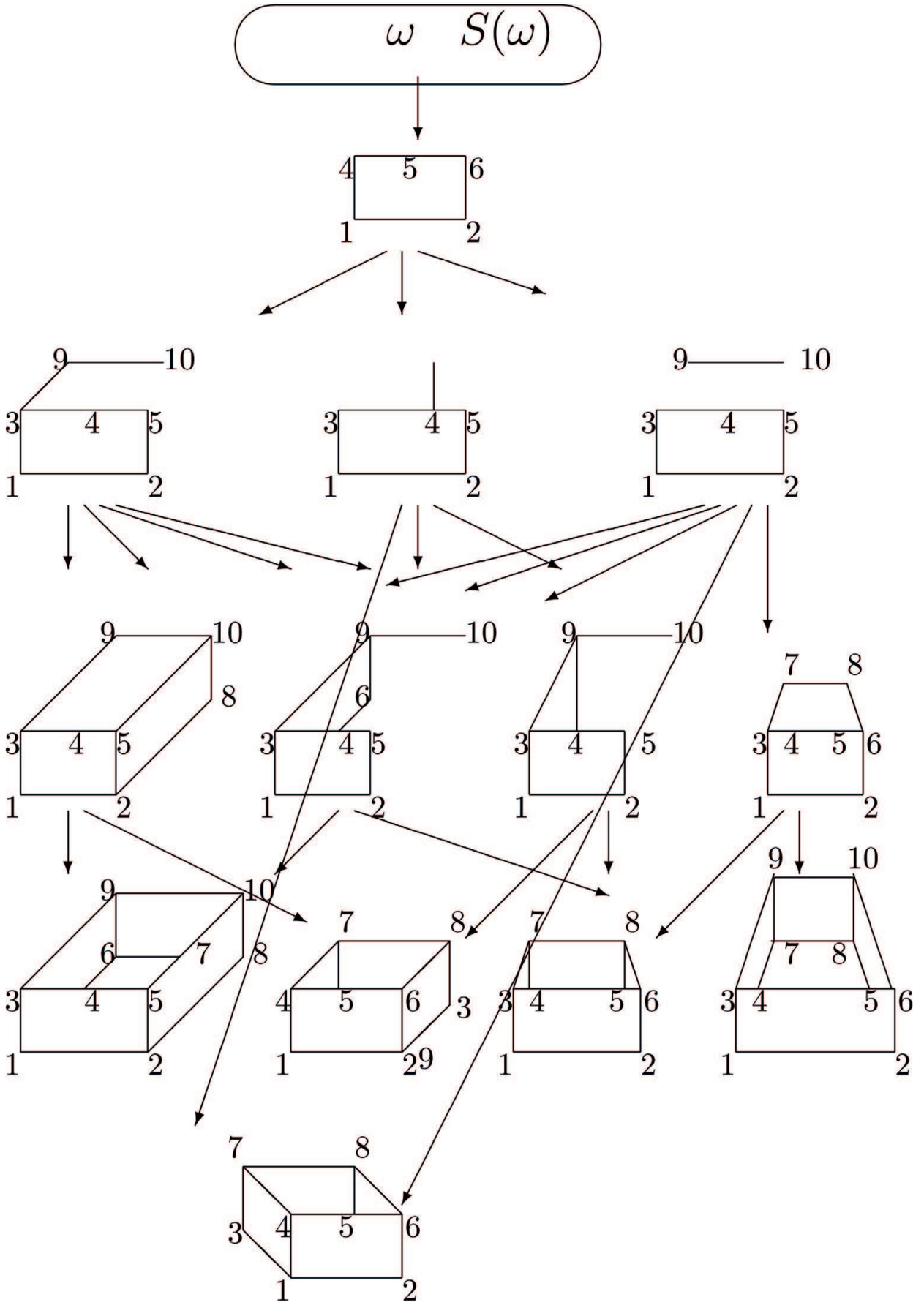


Figure 9. Fragments of the images corresponding to a four-level network.

paradox, it is supposed that every two agents have information concerning some common part of an object. The main difficulty in this problem is to find and identify these parts [8].

5.1. Setting of the problem

Let an investigated object is represented as a set of its elements $\omega = \{\omega_1, \ldots, \omega_t\}$ and is characterized by the set of predicates $p_1,\ldots, p_n$, every of which is defined on the elements of ω and gives properties of these elements or relations between them.

Information (description) of an object is an elementary conjunction of atomic formulas with predicates $p_1,\ldots, p_n$ and some constants as arguments.

There are m agents $a_1,\ldots, a_m$ which can measure some values for some predicates of some elements of ω. The agent a_j does not know the true number of the ω elements and suppose that she deals with the object $\omega_j = \left\{\omega_1^j, \ldots, \omega_{t_j}^j\right\}$. That is, the agent a_j has the information $I_j\left(\omega_1^j, \ldots, \omega_{t_j}^j\right)$ in the form of elementary conjunction of atomic formulas. It is required to reconstruct the full description $I(\omega_1, \ldots, \omega_t)$ of ω (if it is possible).

As every agent uses her own notifications for the names of the object elements, it is needed to find all common up to the names of arguments sub-formulas C_{ij} of the information $I_i\left(\omega_1^i, \ldots, \omega_{t_i}^i\right)$ and $I_j\left(\omega_1^j, \ldots, \omega_{t_j}^j\right)$ $(i \neq j)$ and their unifiers, i.e., such substitutions for the argument names that the extracted pairs of sub-formulas are identical.

5.2. Algorithm of multi-agent description

Below, the arguments of information will be omitted. Let every agent a_j has information I_j about the described object ω ($j = 1,\ldots, m$). To construct a description of ω the following algorithm is offered.

1. Change all constants in $I_1,\ldots, I_m$ by variables in such a way that different constants are changed by different variables and the names of variables in Ii and Ij $(i \neq j)$ does not coincide. Obtain $I'_1,\ldots, I'_m$.
2. For every pair of elementary conjunctions I'_i and I'_j ($i = 1,\ldots, m-1$, $j = i+1, \ldots, m$) find their maximal common up to the names of arguments sub-formula C_{ij} and unifiers $\lambda_{i,ij}$ and $\lambda_{j,ij}$. Every argument of C_{ij} has a unique name.
3. For every pair i and j $(i > j)$ check if I'_i and I'_j contain a contradictory pair of atomic formulas or two sub-formulas which cannot be satisfied simultaneously (for example, "x is green" and "x is red"). If such a contradiction is established, then delete from C_{ij} atomic formulas containing the variables, which are in the contradictory sub-formulas. Change the unifiers by means of elimination of these variables.
4. For every i, identify the variables in C_{ij} $(i \neq j)$ which are substituted in I'_i and I'_j instead of the same variable. The names of the identified variables are changed in unifiers by the same name.
5. With the use of the unifiers obtained in items 2–4 change the names of variables in $I'_1,\ldots, I'_m$. Obtain $I''_1, \ldots, I''_m$.
6. Write down the conjunction I''_1 & … & I''_m and delete the repeated atomic formulas.

5.3. Upper bound of the number of steps

To estimate the number of the algorithm run steps, we estimate every item of the algorithm.

1. Item 1 requires not more than $\sum_{j=l^m} \|I_j\|$ "steps."

2. Item 2 requires $O\left(t_i^{t_j} \cdot 2^{\|I_j\|}\right)$ "steps" for an exhaustive algorithm and $O\left(s_i^{\|I_j\|} \cdot \|I_i\|^3\right)$ "steps" for an algorithm based on the derivation in the predicate calculus.

   It is needed to summarize the above estimates for $i = 1,\ldots, m-1$, $j = i, \ldots, m$. So, we have $O\left(t^t \cdot 2^s m^2\right)$ "steps" for an exhaustive algorithm and $O\left(s^{s+3} \cdot m^2\right)$ "steps" for an algorithm based on the derivation in the predicate calculus. Here, t and $\|I\|$ are the maximal numbers of variables and atomic formulas in I_j ($j = 1,\ldots, m$), respectively.

3. Consistency checking of the formulas I_i and I_j requires $\|I_i\|\|I_j\|$ "steps." This item of the algorithm requires not more than $\sum_{i=1}^{m} (m-i)\|I_i\|$ "steps" that is $O(m^2 s)$ "steps."

4. For every i, identification of the variables in C_{ij} ($i > j$) consists in the comparison of the replaced part of the unifiers $\lambda_{i,ij}$ and $\lambda_{j,ij}$. It requires not more than $(m-i)t_i^2$ "steps." Summarizing it for $i = 1,\ldots, m$ we have not more than $\sum_{i=1}^{m} (m-i)t_i^2 = O\left(m^2 t^2\right)$ "steps".

5. The number of "steps" required for the changing of the names of variables in $I1, \ldots, Im$ is linear under $\sum_{i=1}^{m} \|I_i\| = O(m\|I\|)$ "steps."

6. The number of "steps" required for the deleting of the repeated conjunctive terms is not more than $\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \|I_i\|\|I_j\|$ "steps."

The whole number of the algorithm run steps is $O(t^t\, 2^s\, m^2)$ for an exhaustive algorithm and $O(s^{s+3}\, m^2)$ for an algorithm based on the derivation in the predicate calculus.

The analysis of the received estimation shows that the main contribution is made by the summarized number of partial deduction checking (item 2).

5.4. Example of a multi-agent description

Let the initial predicates be V and L described in Section 2. Each of the three agents has a description of one of the fragment presented in **Figure 10**.

According to the item 1 of the algorithm, all constants in the fragment descriptions are replaced by variables in such a way that different constants are changed by different variables and the names of variables in I_i and I_j ($i \neq j$) does not coincide. The fragment descriptions take the form:

I′1(x1,…,x6) = V(x1,x2,x4) & V(x1,x5,x4) & V(x1,x3,x2) & V(x1,x3,x5) & V(x1,x3,x4) & V(x2,x1, x3) & V(x2,x3,x5) & V(x3,x2,x1) &V(x3,x6,x2) & V(x3,x6,x1) & L(x2,x1,x5),

I′2(y1,…,y6) = V(y3,y1,y4) & V(y1,y2,y3) & V(y1,y5,y3) & V(y1,y6,y2) & V(y1,y6,y5) & V(y1,y6, y3) & L(y2,y1,y5),

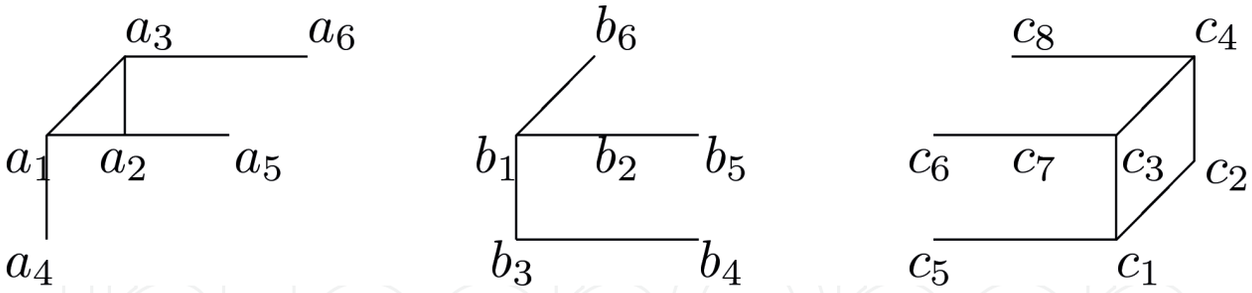


Figure 10. Fragments of the image received by three agents.

I′3(z1,…,z8) = V(z1,z5,z3) & V(z1,z3,z2) & V(z1,z5,z2) & V(z3,z1,z7) & V(z3,z1,z6) & V(z3,z7, z4) & V(z3,z6,z4) & V(z3,z4,z1) & V(z4,z2,z3) & V(z4,z3,z8) & V(z4,z2,z8) & L(z7,z6,z3).

According to the item 2 of the algorithm, find maximal common up to the names of arguments sub-formula of formulas I′1(x1,…,x6) and I′2(y1,…,y6). It is C12(u0,…,u4) of the form C12(u0, …,u4) = V(u0,u1,u2) & V(u0,u3,u2) & V(u0,u4,u1) & V(u0,u4,u3) & V(u0,u4,u2) & L(u1,u0,u3).

It has unifiers $\lambda_{I1,C12}$—substitution of u0, u1, u4, u2, u3 instead of x1, x2, x3, x4, x5, respectively, and $\lambda_{I2,C12}$—substitution of u0, u1, u2, u3, u4 instead of y1, y2, y3, y5, y6, respectively. Besides,

I′1(u0,u1,u2,u3,u4,x6) = V(u1,u0,u4) & V(u1,u4,u3) & V(u4,u1,u0) & V(u4,x6,u1) & V(u4,x6,u0) & C12(u0, …, u4),

I′2(u0,u1,u2,y4,u3,u4) = V(u2,u0,y4) & C12(u0,…,u4).

Maximal common up to the names of arguments sub-formula of I′2(y1,…,y6) and I′3(z1,…,z8) is C23(v0,v2,v4,v5,v6,v7) of the form

C23(v0,v2,v4,v5,v6,v7) = V(v6,v2,v7) & V(v2,v4,v6) & V(v2,v5,v6) &V(v2,v0,v4) & V(v2,v0,v5).

It has unifiers $\lambda_{I2,C23}$—substitution of v2, v4, v6, v7, v5, v0 instead of y1, y2, y3, y4, y5, y6, respectively, and $\lambda_{I3,C23}$—substitution of v0, v2, v6, v5, v4, v7 instead of z1, z3, z5, z6, z7, z8, respectively. Besides,

I′2(v2,v4,v6,v7,v5,v0) = V(v2,v0,v6) & L(v4,v2,v5) & C23(v0,v2,v4,v5,v6,v7),

I′3(v0,z2,v2,v6,z5,v5,v4,v7) = V(v2,v6,v0) & V(v0,z5,v2) & V(v0,v2,z2) & V(v0,v5,z2) & V(v6, z2,v2) & V(v6,v2,v7) & L(v4,v5,v2) & C23(v0,v2,v4,v5,v6,v7).

As I′2(v2,v4,v6,v7,v5,v0) contains V(v2,v0,v6) and I′3(v0,z2,v2,v6,z5,v5,v4,v7) contains V(v2, v6,v0) and according to the definition of the predicate V, the formula V(x,y,z) & V(x,z,y) is a contradiction, so substitutions with this unifiers cannot give a consistent description of the object. After deleting from I′2(y1,…,y6) and I′3(z1,…,z8), the variables y1 and z3, respectively, a new maximal common up to the names of arguments their sub-formula C′23(v0,v2,v4,v5,v6, v7) of the form C′23(v0,v1,v2) = L(v1,v0,v2) will be received with the unifiers $\lambda_{I2,C'23}$— substitution of v0, v1, v2 instead of y1, y2, y3, respectively, and $\lambda_{I3,C'23}$—substitution of v2, v0, v1 instead of z3, z6, z7, respectively. Besides,

I′2(v0,v1,v2,y4,y5,y6) = V(v2,v0,y4) & V(v0,v1,v2) & V(v0,y5,v2) & V(v0,y6,v1) &V(v0,y6,y5) & V(v0,y6,v2) & C′23(v0,v1,v2),

I′3(z1,z2,v2,z4,z5,v0,v1,z8) = V(z1,z5,v2) & V(z1,v2,z2) & V(z1,z5,z2) & V(v2,z1,v1) & V (v2,z1, v0) & V(v2,v1,z4) & V(v2,v0,z4) & V(v2,z4,z1) & V(z4,z2,v2) & V(z4,v2,z8) & V(z4,z2,z8) & C′23(v0,v1,v2).

Maximal common up to the names of arguments sub-formula of I1(x1,…,x6) and I3(z1,…,z8) is C13(w0, …,w6) in the form

C13(w0, …,w6) = V(w2,w4,w6) & V(w2,w5,w6) & V(w2,w0,w4) & V(w2,w0,w5) & V(w0,w1,w2).

It has unifiers $\lambda_{I1,C13}$— substitution of w2, w4, w0, w6, w5, w6 instead of x1, x2, x3, x4, x5, x6, respectively, and $\lambda_{I3,C13}$—substitution of w0, w2, w6, w1, w5, w2 instead of z1, z3, z4, z5, z6, z7, respectively. Besides,

I′1(w2,w4,w0,w6,w5,w1) = V(w2,w0,w6) & V(w0,w1,w4) & V(w0,w4,w2) & L(w2,w4,w5) & C13(w0,…,w6),

I′3(w0,z2,w2,w6,w1,w5,w4,z8) = V(w0,w2,w3) & V(w0,w1,w3) & V(w2,w6,w0) & V(w6,w3, w2) & V(w6,w2,w7) & V(w6,w3,w7) & C13(w0,…,w6).

As I′1(w2,w4,w0,w6,w5,w1) contains V(w2,w0,w6), I3(w0,z2,w2,w6,w1,w5,w4,z8) contains V (w2,w6,w0) and according to the definition of the predicate V, the formula V(x,y,z) & V(x,z,y) is a contradiction, so substitutions with this unifiers cannot give a consistent description of the object.

After deleting from I′1(x1,…,x6) and I′3(z1,…,z8) literals with the variables x1 and z3, respectively, a new maximal common up to the names of arguments their sub-formula

C′13(w0,w1,w2) of the form C′13(w0,w1,w2) = L(w1,w0,w2)

will be received with the unifiers $\lambda_{I1,C'13}$—substitution of w0, w1, w2 instead of x1, x2, x5, respectively, and $\lambda_{I3,C'13}$—substitution of w2, w1, w0 instead of z3, z4, z5 respectively. Besides,

I′1(w0,w1,x3,x4,w2,x6) = V(w0,w1,x4) & V(w0,w2,x4) & V(w0,x3,w1) & V(w0,x3,w2) & V(w0,x3,x4) & V(w1,w0,x3) & V(w1,x3,w2) & V(x3,w1,w0) & V(x3,x6,w1) & V(x3,x6,w0) & C′13(w0,w1,w2),

I′3(z1,z1,w2,w1,w0,z6,z7,z8) = V(z1,w0,w2) & V(z1,w2,z2) & V(z1,w0,z2) & V(w2,z1,z7) & V (w2,z1,z6) & V(w2,z7,w1) & V(w2,z6,w1) & V(w2,w1,z1) & V(w1,z2,w2) & V(w1,w2,z8) & V (w1,z2,z8) & C′13(w0,w1,w2).

According to the item 4 of the algorithm, we identify new variables substituted instead of the same initial variable. That is we identify the following variables:
u0 and w0 (are substituted instead of the variable x1),

u1 and w1 (are substituted instead of the variable x2),

u2 and w2 (are substituted instead of the variable x4),

u0 and v0 (are substituted instead of the variable y1),

u1 and v1 (are substituted instead of the variable y2),

u2 and v2 (are substituted instead of the variable y3),

v0 and w0 (are substituted instead of the variable z6),

v1 and w1 (are substituted instead of the variable z3),

v2 and w2 (are substituted instead of the variable z7).

The identified variables denote as α 0, α 1, and α 2. So, we have the equalities u0 = v0 = w0 = α 0, u1 = v1 = w1 = α 1, u2 = v2 = w2 = α 2.

As a result, we have the following descriptions of the fragments:

I″1(α 0, α 1,u4,u2, α 2,x6) = V(α 0, α 1,u2) & V(α 0, α 2,u2) & V(α 0,u4, α 1) & V(α 0,u4, α 2) & V (α 0,u4,u2) & V(α 1, α 0,u4) & V(α 1,u4, α 2) & V(x3, α 1, α 0) & V(u4,x6, α 1) & V(u4,x6, α 0) & L(α 1, α 0, α 2),

I″2(α0, α 1,u2,y4, α 2,u4) = V(u2, α 0,y4) & V(α 0, α 1,u2) & V(α 0, α 2,u2) & V(α 0,u4, α 1) & V (α 0,u4, α 2) & V(α 0,u4,u2) & L(α 1, α 0, α 2),

I″3(z1,z2, α 2,z4,z5, α 0, α 1,z8) = V(z1,z5, α 2) & V(z1, α 2,z2) & V(z1,z5,z2) & V(α 2,z1, α 1) & V(α 2,z1, α 0) & V(α 2, α 1,z4) & V(α 2, α 0,z4) & V(α 2,z4,z1) & V(z4,z2, α 2) & V(z4, α 2,z8) & V(z4,z2,z8) & L(α 1, α 0, α 2).

Their conjunction

V(α0, α 1,u2) & V(α 0, α 2,u2) & V(α 0,u4, α 1) & V(α 0,u4, α 2) & V(α 0,u4,u2) &

V(α 1, α 0,u4) & V(α 1,u4, α 2) & V(x3, α 1, α 0) & V(u4,x6, α 1) & V(u4,x6, α 0) &

V(u2, α 0,y4) & V(z1,z5, α 2) & V(z1, α 2,z2) & V(z1,z5,z2) & V(α 2,z1, α 1) &

V(α 2,z1, α 0) & V(α 2, α 1,z4) & V(α 2, α 0,z4) &V(α 2,z4,z1) & V(z4,z2, α 2) &

V(z4, α 2,z8) & V(z4,z2,z8) & L(α 1, α 0, α 2)

allows to "stick together" the images of fragments according to the same variable. The image corresponding to the result of "sticking" is presented in **Figure 11**.

If a description of the investigated object is presented in the database, it may be found according the principle "the nearest neighbor" with the use of metric for predicate formulas presented in [13].

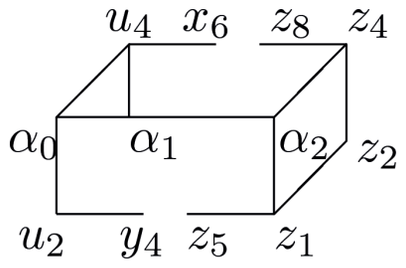


Figure 11. Image corresponding to the result of "sticking".

6. Conclusion

Logic-predicate approach to an AI problem has a rather powerful capability, essentially when an investigated object is a compound one and is characterized by properties of its elements and relations between them.

Setting of pattern recognition problems considered in Section 2 (except the problem (2)) differs from the classical one. The setting of the problems (1) and (3), in which it is needed to find parts of an investigated object, turns out to be a rather difficult one in the frameworks of a standard approach in the frameworks of which an object is regarded as a whole indivisible one.

In particular, an exponential estimation for number of propositional variables in a formula simulating a predicate formula in a finite domain for planning problems $T \cdot |Act| \cdot O^P$ is mentioned in [14]. Here, T is the number of time stages, $|Act|$ is the number of schemes of actions, O is the number of objects in the domain, P is the maximal number of parameters in schemes of actions. In Section 2, the analogous estimate (5′) was received for an exhaustive algorithm solving the problem (4).

The problem (2) is polynomial equivalent to an "open" problem ISOMORPHISM OF GRAPHS [3] and the problems (1) and (3) are NP-complete.

A notion of level description of classes has been introduced in Section 3 in order to decrease the number of steps of algorithms solving these problems. Such a description reduces the solution of the main problem to a series of solutions of the same form problems with the inputs with the essentially less notation lengths. At the same time, the constructing of a level description still deals with big input data. So, a problem with big input data is solving only once, and then the problem with the essentially less input data is solving repeatedly.

The idea of decomposition of a problem to a series of the "less dimension" problems is not a new one and is frequently used. The difficulty consists in a precise definition of the term "common sub-formula of small complexity."

The development of a precise definition and of an algorithm for the extraction of a common up to the names of arguments sub-formula of two elementary conjunctions (and their unifiers) allows not only to work out an algorithm of level description construction but also to find an approach to the solution of some else AI problems.

Note that the extracted sub-formulas define generalized characteristics of an object. This has an analogy in medical diagnostics: initial characteristics are symptoms and the generalized ones are syndromes.

Level description of classes allowed to introduce the notion of logic-predicate network described in Section 4. Such a network may be regarded as a self-training network which changes its configuration after an additional training. It corresponds to the fact that in the process of a man training, new notions and relations between them are formed in a human brain.

The presence of an algorithm for the extraction of a common up to the names of arguments sub-formula of two elementary conjunctions (and their unifiers) allows to find an approach to a problem of multi-agent description of an object described in Section 5. Just an extraction of

such sub-formulas and determining of their unifiers with the input formulas makes possible to "stick together" such parts of descriptions in which different agents gives different names to one element of the whole object.

Note that the formulation of the problem (1) from Section 2 coincides with the one for a well-known problem CONJUNCTIVE BOOLEAN QUERY from [3]. The difference is in the implementation of these problems. While repeated implementation of the problem (1) the premise S(ω) of the sequent $S(\omega) \Rightarrow \exists \overline{x}_{\neq} A_k(\overline{x})$ is different, while every implementation and the conclusion $A_k(\overline{x})$ is a constant part. That is why just a collection of class descriptions $A_1(\overline{x}_1)$, …, $A_K(\overline{x}_K)$ permits to construct a level description.

While repeated implementation of the problem CONJUNCTIVE BOOLEAN QUERY, the premise S(ω) of the sequent $S(\omega) \Rightarrow \exists \overline{x}_{\neq} A_k(\overline{x})$ is constant while every implementation and queries $A_k(\overline{x})$ are different while every implementation. An approach to the construction of a level data base is presented in [15].

The possibility of reduction of an object description length by means of adding a formula setting some properties of initial predicates to the premise of a sequent was mentioned in the model example in Section 2. Properties of initial predicates also were used in the item 3 of the algorithm of multi-agent description. In fact, in the both cases instead the sequent of the form (4) $S(\omega) \Rightarrow \exists \overline{x}_{\neq} A(\overline{x})$ it is needed to check another sequent of the form $C(\overline{y})\ S(\omega) \Rightarrow \exists \overline{x}_{\neq} A(\overline{x})$, where $C(\overline{y})$ is a set of formulas setting properties of initial predicates. Investigation of computational complexity of such a form sequent may be an interesting problem in the further research.

To solve the problem (2) and to extract a maximal common up to the names of arguments subformula of two elementary conjunctions it is needed to check whether two elementary conjunctions are isomorphic. A polynomial in time rough algorithm for such a checking was offered in [12] by Petrov. Numerical experiments with this algorithm give over 99.95% of valid results.

Author details

Tatiana Kosovskaya

Address all correspondence to: kosovtm@gmail.com

St. Petersburg State University, St. Petersburg, Russia

References

[1] Nilson N. Problem-Solving Methods in Artificial Intelligence. New York: McGraw-Hill Book Company Press. Publ; 1971. 280p

[2] Du DZ, Ko KI. Theory of Computational Complexity. New York: John Wiley & Sons, Inc.; 2000:512 p. A Wiley-Interscience Publication

[3] Garey M, Johnson D. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: Freeman Press; 1979. 340p

[4] Kossovskaya T. Proofs of the number of steps bounds for solving of some pattern recognition problems with logical description. Series 1. Mathematics, Mechanics, Astronomy. St. Petersburg: Vestnik of St. Petersburg State University; 2007. issue 4. pp. 82-90 (in Russian)

[5] Kosovskaya T. Some artificial intelligence problems permitting formalization by means of predicate calculus language and upper bounds of their solution steps. In: SPIIRAS Proceedings; No. 14; 2010. pp. 58-75 (in Russian)

[6] Kosovskaya T. Construction of class level description for efficient recognition of a complex object. International Journal Information Content and Processing. 2014;**1**(1):92-99

[7] Kosovskaya T. Self-modificated predicate networks. International Journal on Information Theory and Applications. 2015;**22**(3):245-257

[8] Kosovskaya T. Multi-agent description of an object by means of a predicate calculus language. International Journal on Information Theories and Applications. 2016;**23**(4): 338-346

[9] Kosovskaya T. Discrete artificial intelligence problems and number of steps of their solution. International Journal on Information Theories and Applications. 2011;**18**(1):93-99

[10] Orevkov VP. The inverse method of logical derivation. In: Adamenko A, Kuchukov A, editors. Programming Logic and Visual Prolog. St. Petersburg: BHV-Petersburg; 2003. pp. 952-965. (in Russian)

[11] Kosovskaya T, Petukhova N. The inverse method for solving artificial intelligence problems in the frameworks of logic-objective approach and bounds of its number of steps. International Journal Information Models and Analyses. 2012;**1**:84-93

[12] Kosovskaya TM, Petrov DA. Extraction of a maximal common sub-formula of predicate formulas for the solving of some Artificial Intelligence problems. Series 10. Applied Mathematics. Computer Science. Control Processes. St. Petersburg: Vestnik of St. Petersburg State University; 2017. issue 3. pp. 250-263 (in Russian)

[13] Kosovskaya T. Distance between objects described by predicate formulas. In: Deza M, Petitjean M, Markov K. editors. International Book Series. Information Science and Computing. Book 25. Mathematics of Distances and Applications. Sofia, Bulgaria: ITHEA; 2012. pp. 153-159

[14] Russel SJ, Norvig P. Artificial Intelligence. A Modern Approach. New York: Pearson Education, Inc.; 2003:1132 p

[15] Kosovskaya T. Conjunctive Boolean Query as a logic-objective recognition problem. International Journal on Information Theories and Applications. 2017;**24**(3):79-82