Algorithms of Isomorphism of Elementary Conjunctions Checking

T. Kosovskaya[a],* and Juan Zhou[a],**

[a] *St. Petersburg State University, St. Petersburg, 199034 Russia*

** e-mail: kosovtm@gmail.com*

*** e-mail: zhoujuanna@163.com*

Abstract—When solving artificial intelligence problems related to the study of complex structured objects, a convenient tool for describing such objects is the language of predicate calculus. The paper presents two algorithms for checking the isomorphism of pairs of elementary conjunctions of predicate formulas (they coincide up to variable names and the order of conjunctive terms). The first of the algorithms checks elementary conjunctions containing a single predicate symbol for isomorphism. Furthermore, if the formulas are isomorphic, it finds a one-to-one correspondence between the arguments of these formulas. If all predicates are binary, the proposed algorithm is an algorithm for checking two directed graphs for isomorphism. The second algorithm checks elementary conjunctions containing multiple predicate symbols for isomorphism. Estimates of their time complexity are given for both algorithms.

Keywords: predicate formulas, isomorphism of predicate formulas, complex structured object

DOI: 10.1134/S1054661824010103

1. INTRODUCTION

When solving problems related to the study of complex structured objects (CSO), a convenient tool for describing such objects is the predicate calculus language. This way of describing the CSO was proposed in the middle of the XX century in [1, 10] and many other authors, and also continues to be considered currently, for example, in [11].

The main problem of using the predicate calculus language is the NP-hardness [2] or GI-hardness [7] of problems arising in such descriptions of objects and classes of objects.

To decrease the computational complexity of such problems, the creation of a multilevel description of classes was proposed in [3] and clarified in [5]. In these works, generalized predicates were used to create a multilevel description of polyhedra, which were obtained based on the specifics of the descriptions of polyhedra. However, there remains the question of obtaining generalized predicates for arbitrary classes of objects.

To solve this problem, the concept of the maximum common property (MCP) of objects was introduced, in the initial descriptions of which there are arbitrary properties of the elements of the CSO and relationships between these elements. The basic concept for the extraction of MCP is the concept of isomorphism of elementary conjunctions of predicate formulas [6].

Received October 6, 2023; revised October 30, 2023; accepted October 30, 2023

Except constructing a multilevel description of objects, MCP extraction can be used, for example, to construct an ontology [8, 9].

The first step to develop an MCP allocation algorithm is to develop an algorithm for checking two elementary conjunctions of predicate formulas for isomorphism (matches up to the names of variables and the order of conjunctive terms).

This paper presents two algorithms for checking two elementary conjunctions for isomorphism, based on the algorithm proposed in [4].

The first of the algorithms checks for isomorphism elementary conjunctions containing a single predicate symbol. In addition, if the formulas are isomorphic, then it finds a one-to-one correspondence between the arguments of these formulas.

Provided that all predicates are binary, the proposed algorithm is an algorithm for checking two directed graphs for isomorphism.

The second algorithm checks for isomorphism elementary conjunctions containing several predicate symbols.

Estimates of their time complexity are given for both algorithms.

2. NECESSARY DEFINITIONS

Definition 1. A *complex structured object* (*CSO*) is an object $\omega = \{\omega_1, \dots, \omega_t\}$, the elements ω_i of those have specified properties (satisfy unary predicates) and are in specified relationships (satisfy multiplace predicates). Let these predicates are $p_1, \dots, p_n$.

Definition 2. *Description $S(\omega)$ of a CSO* ω is an elementary conjunction of atomic formulas with predicates $p_1,\ldots,p_n$, which is the maximum in the number of literals, and is true for ω.

Definition 3. Two elementary conjunctions of atomic formulas of predicate calculus, $P(a_1,\ldots,a_m)$ 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 is such an elementary conjunction $R(x_1,\ldots,x_m)$ and permutations $a_{i_1},\ldots,a_{i_m}$ and $b_{j_1},\ldots,b_{j_m}$ of arguments of formulas $P(a_1,\ldots,a_m)$ and $Q(b_1,\ldots,b_m)$ correspondingly, so that substituting all occurrences of variables $x_1,\ldots,x_m$ of the formula $R(x_1,\ldots,x_m)$ with $a_{i_1},\ldots,a_{i_m}$ and $b_{j_1},\ldots,b_{j_m}$ we obtain formulas $R(a_{i_1},\ldots,a_{i_m})$ and $R(b_{j_1},\ldots,b_{j_m})$ that coincide with the formulas $P(a_1,\ldots,a_m)$ and $Q(b_1,\ldots,b_m)$, respectively, up to the order of literals.

The resulting substitutions $\left|\begin{matrix}x_1,\ldots,x_m\\a_{i_1},\ldots,a_{i_m}\end{matrix}\right.$ [1] and $\left|\begin{matrix}x_1,\ldots,x_m\\b_{j_1},\ldots,b_{j_m}\end{matrix}\right.$ are called *unifiers of formulas $P(a_1,\ldots,a_m)$ and $Q(b_1,\ldots,b_m)$* with the formula $R(x_1,\ldots,x_m)$, respectively.

In the above definition, one could do without introducing the formula $R(x_1,\ldots,x_m)$ into it, as this is done in the definition of isomorphic graphs. But, firstly, it is to take into account that no substitution in the formula instead of constants is allowed. Secondly, in the future, this formula will act precisely as a formula with variables that sets the common property of two CSO.

Note that the arguments of elementary conjunctions P and Q can be both object variables and object constants. In addition, the concept of isomorphism of elementary conjunctions of atomic formulas of predicate calculus differs from the concept of equivalence of these formulas, because they can have significantly different arguments. In fact, for isomorphic formulas, there are such permutations of their arguments that they define the same relation between these arguments.

Definition 4. An elementary conjunction that does not contain constants is called a *common property* of two objects if it is isomorphic to some subformulas of each of the descriptions of these objects.

Definition 5. An elementary conjunction that does not contain constants is called a *maximum common property* (*MCP*) of two objects if it is their common property with the largest number of literals.

Definition 6. A string of the form $\chi(a_i)=(n_1,\ldots,n_k)$, where n_j is the number of occurrences of the variable a_i as the jth argument of the literal is called a *characteristic of an argument a_i* in the elementary conjunction of literals with the only one k-ary predicate symbol.

Definition 7. An ordered

• by the minimal nonzero number of occurrences of the variable in the 1st, 2nd, ... , kth place;

• by the minimal nonzero number of occurrences of other variables among a group with the same minimal nonzero number of occurrences of a variable string of the form $\chi(C)=(\chi(a_{i_1}),\ldots,\chi(a_{i_n}))$, where n is the number of variables in C, is called a *characteristic of an elementary conjunction C* of literals with the same k-ary predicate symbol.

Example 1. For elementary conjunction

$$P(d,b,e)\,\&\,P(e,f,a)\,\&\,P(a,b,c)$$
$$\&\,P(e,d,c)\,\&\,P(a,f,d)$$

the characteristics of the arguments have the form

$$\chi(a)=(2,0,1),$$
$$\chi(b)=(0,2,0),$$
$$\chi(c)=(0,0,2),$$
$$\chi(d)=(1,1,1),$$
$$\chi(e)=(2,0,1),$$
$$\chi(f)=(0,2,0),$$

and the characteristic of this formula is

$$\chi(C)=(\chi(d),\chi(a),\chi(e),\chi(b),\chi(f),\chi(c))$$
$$=((1,1,1)(2,0,1)(2,0,1)(0,2,0)(0,2,0)(0,0,2)).$$

Theorem 1 [4]. *In order for two elementary conjunctions of literals with the same predicate symbol to be isomorphic, it is necessary that their characteristics are equal.*

Definition 8. The number of arguments in the formula that have this characteristic value is called the *length of the characteristic value*.

When describing the algorithm for checking for isomorphism of two elementary conjunctions *R* and *F* containing a single predicate symbol (moreover, the elementary conjunction *R* contains variables, and the elementary conjunction *F* contains only constants), the structure *mapping* will be used. Let $[x_{i_1},\ldots,x_{i_k}]$ be a list of all variables with the same characteristic in the formula *R*, $[c_{j_1},\ldots,c_{j_k}]$ be a list of constants with the same characteristic in the formula *F*. Then the structure *mapping* has the form $\{[x_{i_1},\ldots,x_{i_k}]:[c_{j_1},\ldots,c_{j_k}]\}$.

[1] The notation $P\left|\begin{matrix}x_1,\ldots,x_m\\a_1,\ldots,a_m\end{matrix}\right.$ is used to replace all free occurrences in the formula P of variables $x_1,\ldots,x_m$ with constants $a_1,\ldots,a_m$ respectively.

That is, each of these constants is possible for substitution in R instead of any of these variables, and no other constant from F is suitable for substitution in R.

More precisely, this structure is initially constructed as follows: in a cycle by the values of the characteristics of variables:

– write out all the variables of the formula R having the same characteristic value,

– write out all the constants of the formula F that have the same characteristic value.

Example 2. In Example 1 for elementary conjunction

$$C = P(d,b,e) \ \& \ P(e,f,a) \ \& \ P(a,b,c) \ \& \ P(e,d,c) \ \& \ P(a,f,d),$$

its characteristic was calculated

$$\chi(C) = (\chi(d),\chi(a),\chi(e),\chi(b),\chi(f),\chi(c)) = ((1,1,1),(2,0,1),(2,0,1),(0,2,0),(0,2,0),(0,0,2)).$$

For elementary conjunction

$$C' = P(y,x,z) \ \& \ P(z,u,v) \ \& \ P(v,x,w) \ \& \ P(z,y,w) \ \& \ P(v,u,y)\ ,$$

its characteristic has the form

$$\chi(C') = (\chi(y),\chi(z),\chi(v),\chi(u),\chi(x),\chi(w)) = ((1,1,1),(2,0,1),(2,0,1),(0,2,0),(0,2,0),(0,0,2)).$$

As you can see, the characteristics match.

The structure *mapping* is a list of all pairs ordered in ascending order of the lengths of the pairs

$$\{\,[y]:[d],\ [w]:[c],\ [z,v]:[a,e],\ [x,u]:[b,f]\}.$$

Definition 9. A pair of subformulas R' and F' of elementary conjunctions R and F, respectively, is called *contradictory* if there is a pair {[...x_t, ...] : [...c_r, ...]} in the structure mapping'for R' and F', but in the structure mapping for R and F a pair {[..,x_t, ...] : [......]} does not contain c_r or the pair {[......] : [...c_r, ...]} does not contain x_t.

3. ALGORITHM ISOM-1 FOR CHECKING ISOMORPHISM OF TWO ELEMENTARY CONJUNCTIONS CONTAINING A SINGLE PREDICATE SYMBOL

Let two elementary conjunctions of literals $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ with the same number of literals with a single predicate symbol be given. To check them for isomorphism, the following algorithm is proposed.

(1) Find the characteristics of elementary conjunctions $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$.

(2) If the characteristics do not coincide, then the formulas are not isomorphic. The algorithm stops its run.

(3) As a formula $R(x_1, ..., x_n)$ take the formula $F_1(a_1, ..., a_n)$. The unifier of formulas $R(x_1, ..., x_n)$ and $F_1(a_1, ..., a_n)$ is an identical substitution $\lambda_{RF_1} = \left|{}^{x_1,\dots,x_n}_{a_1,\dots,a_n}\right.$.

(4) For each value of the characteristics of the arguments of the formulas $R(x_1, ..., x_n)$ and $F_2(b_1, ..., b_n)$ calculate its length.

(5) Write out the argument lists for $R(x_1, ..., x_n)$ and $F_2(b_1, ..., b_n)$ having the same characteristic.

(6) Fill in the *mapping* structure for $R(x_1, ..., x_n)$ and $F_2(b_1, ..., b_n)$.[2]

(7) If in the *mapping* structure every pair has a length greater than 1, then go to Item 14.

If there are both pairs with a length equal to 1 and pairs with a length greater than 1, then go to Item 8.

Otherwise, *mapping* contains only entries of the form $[x_i] : [b_j]$ with a length equal to 1. This means that the value for x_i can only be b_j. A unifier was found for $R(x_1, ..., x_n)$ and $F_2(b_1, ..., b_n)$ containing all variables. The formulas are isomorphic. The algorithm stops its run.

(8) For each pair of *mapping* structure with the length 1 of the form $[x_i] : [b_j]$ replace the variable x_i in the formula R with the constant b_j.

(9) If the formulas R and $F_2(b_1, ..., b_n)$ coincide up to the permutation of literals, then the formulas are isomorphic. The algorithm stops its run.

(10) Otherwise, divide each of the formulas R and $F_2(b_1, ..., b_n)$ into sub-formulas R^+ and F_2^+ containing only literals with the constant b_j, and R^-, F_2^-, in which the constant b_j is missing.

(11) If the numbers of literals in the formulas R^+ and F_2^+ are not equal, the formulas are not isomorphic. The algorithm stops its run.

Otherwise, check R^+ and F_2^+ for inconsistency.

(12) If R^+ and F_2^+ are contradictory, then the formulas are not isomorphic. The algorithm stops its run.

(13) If there is no inconsistency, then we take the value b_j as the value for the variable x_i in the formula R^+ and in *mapping*.

If values are found for all variables in R^+, then take R^-, F_2^- as R and F_2. Proceed to the execution of Item 7.

If there are variables left in R^+, then delete literals without variables in R^+ and F_2^+ and take them as R and F_2. Go to Item 15.

[2] Because in the process of the algorithm run, some (and eventually all) variables in the formula $R(x_1, ..., x_n)$ will be replaced with constants, then in the further description of the algorithm, the arguments of this formula will not be written out and we will simply write R.

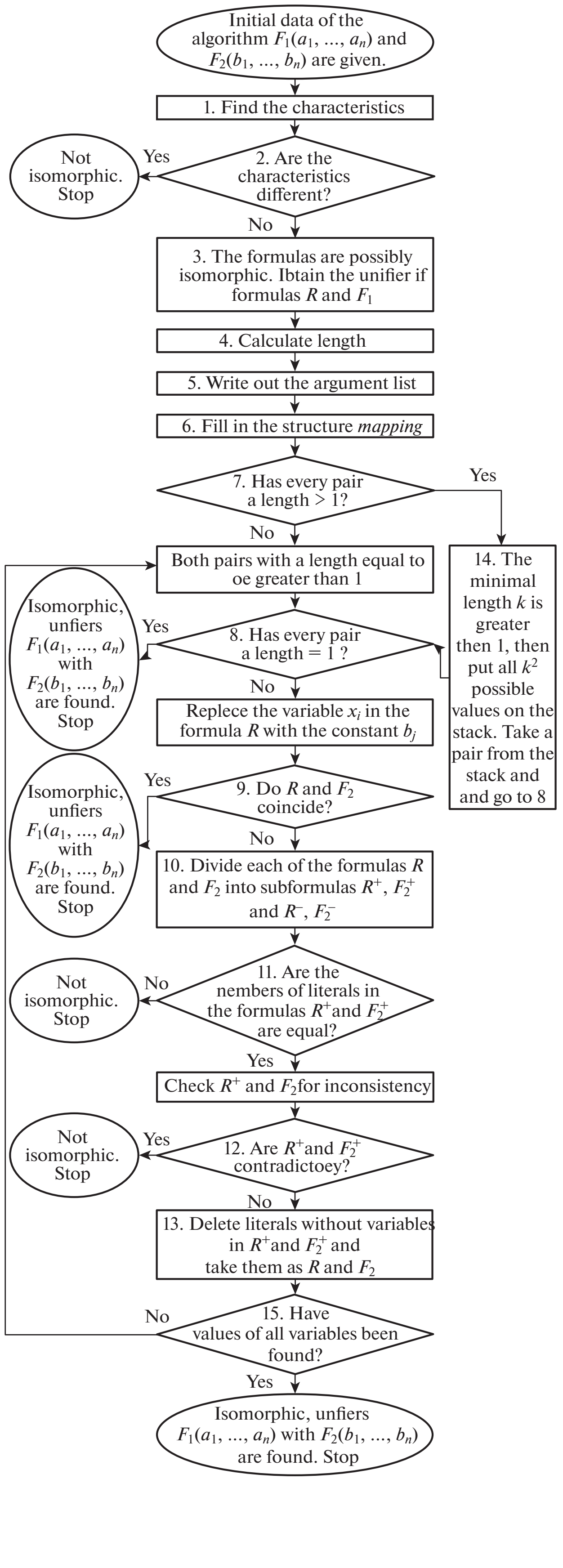

Initial data of the algorithm $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are given.
1. Find the characteristics
2. Are the characteristics different?
Yes
Not isomorphic. Stop
No
3. The formulas are possibly isomorphic. Ibtain the unifier if formulas R and F_1
4. Calculate length
5. Write out the argument list
6. Fill in the structure *mapping*
7. Has every pair a length > 1?
Yes
No
Both pairs with a length equal to oe greater than 1
14. The minimal length k is greater then 1, then put all k^2 possible values on the stack. Take a pair from the stack and go to 8
8. Has every pair a length = 1 ?
Yes
Isomorphic, unfiers $F_1(a_1, ..., a_n)$ with $F_2(b_1, ..., b_n)$ are found. Stop
No
Replece the variable x_i in the formula R with the constant b_j
9. Do R and F_2 coincide?
Yes
Isomorphic, unfiers $F_1(a_1, ..., a_n)$ with $F_2(b_1, ..., b_n)$ are found. Stop
No
10. Divide each of the formulas R and F_2 into subformulas R^+, F_2^+ and R^-, F_2^-
11. Are the nembers of literals in the formulas $R^+$and F_2^+ are equal?
No
Not isomorphic. Stop
Yes
Check R^+ and $F_2$for inconsistency
12. Are $R^+$and F_2^+ contradictoey?
Yes
Not isomorphic. Stop
No
13. Delete literals without variables in $R^+$and F_2^+ and take them as R and F_2
15. Have values of all variables been found?
No
Yes
Isomorphic, unfiers $F_1(a_1, ..., a_n)$ with $F_2(b_1, ..., b_n)$ are found. Stop

Fig. 1. Block-scheme of the algorithm ISOM-1.

(14) If the minimal length k of the pair $\{[x_{i_1},\ldots,x_{i_k}]:[b_{j_1},\ldots,b_{j_k}]\}$ in the structure *mapping* is greater than 1, then put all k^2 possible values for one variable $\{[x_{i_r}]:[b_{j_t}]\}$ for $1 \le r \le k$, $1 \le t \le k$ on the stack. Take a pair from the stack and in *mapping* replace $\{[x_{i_1},\ldots,x_{i_k}]:[b_{j_1},\ldots,b_{j_k}]\}$ with $\{[x_{i_r}]:[b_{j_t}]\}$. Repeat Items 8–13 until a value for the variable x_{i_r} is found or one of the following situations occurs:

(a) Formulas R and $F_2(b_1, ..., b_n)$ coincide up to the permutation of literals, i.e., formulas $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are isomorphic.

(b) The numbers of literals in formulas R^+ and F_2^+ are not equal (formulas $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are not isomorphic).

(c) Inconsistency found in R^+ and F_2^+ (formulas $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are not isomorphic).

(15) Check whether the values of all variables have been found. If yes, then the formulas are isomorphic, the algorithm stops its run.

If not, then go to Item 7.

The block-scheme of the algorithm ISOM-1 is presented in Fig. 1.

Comment. If the formulas are isomorphic and you need to find all their unifiers, then after answering that they are isomorphic, you should check that the stack started in Item 14 of this algorithm is not empty.

4. ABOUT THE ALGORITHM ISOM-1 COMPLEXITY

Items 1–13 of the algorithm are executed in a polynomial (no more than a quadratic) number of steps. The main contribution to the evaluation of computational complexity is made by the implementation of Item 14 of the algorithm.

In the worst case, after the first execution of Item 7, the algorithm proceeds to the Item 14. Here there is an exhaustive search – a tree with height k and degrees of branching k^2, $(k-1)^2$, ... Thus, the upper bound of the computational complexity of the algorithm is $2^{n\log n}$, where n is the number of arguments in each of the formulas.

5. GRAPH ISOMORPHISM

In [7] it is proved that the problem of checking two elementary conjunctions for isomorphism is polynomially equivalent to the problem of checking for isomorphism of two graphs (GI). Moreover, the GI problem is a narrowing of the considered problem if there is only one predicate in the elementary conjunction and it is two-place.

In particular, if the graph is oriented and without loops, then the characteristic of the vertex is a pair: the degree of exodus (the number of edges leaving the vertex) and the degree of entry (the number of edges entering the vertex). In this case, the algorithm proposed above can be applied without changes.

For a nonoriented graph, its degree acts as a characteristic of a vertex.

6. ALGORITHM ISOM FOR CHECKING ISOMORPHISM OF TWO ELEMENTARY CONJUNCTIONS CONTAINING SEVERAL PREDICATE SYMBOLS

Let two elementary conjunctions of literals $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ with the same number of literals with multiple predicate characters $p_1, ..., p_m$ be given. It is required to check them for isomorphism and, in the case of isomorphism, find the elementary conjunction $R(x_1, ..., x_n)$ and its unifiers with $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$.

To solve this problem, it is necessary to expand the definition of the characteristic of an argument and the characteristic of an elementary conjunction.

By means of C_{p_i} we will denote a subformula of elementary conjunction C containing all literals with k-ary predicate symbol p_i and only them.

Definition 10. A string of the form $\chi_{p_i}(a_i) = p_i(n_1,\ldots,n_k)$ is called a *characteristic of the argument* a_i in the subformula C_{p_i} of the elementary conjunction C. Here $(n_1, ..., n_k)$ *is* a characteristic of an elementary conjunction C_{p_i} with one predicate symbol p_i.

Definition 11. The list of characteristics of the arguments of an elementary conjunction C, ordered by increasing the number of variables in the subformulas C_{p_i} $(i = 1, ..., m)$ is called a *characteristic of an elementary conjunction C*.

It is necessary to make small changes to the structure *mapping*. This structure for the elementary conjunction C will consist of a list of structures *mapping* (marked by the predicate symbol) for the subformulas C_{p_i}.

Example 3. For the elementary conjunction

$$C(x,y,a,b,c,d)$$
$$= p_1(x,y)\ \&\ p_1(a,b)\ \&\ p_1(c,d)\ \&\ p_1(d,y)$$
$$\&\ p_2(x,z,a,b)\ \&\ p_2(x,y,b,a)$$

its characteristic has the form

$$\chi(C(x,y,z,a,b,c,d)) = (\chi(C_{p_2}),\ \chi(C_{p_1}))$$
$$= p_2((2,0,0,0),(0,1,0,0),(0,1,0,0),(0,0,1,1),(0,0,1,1)),$$
$$p_1((1,0),(0,2),(0,0),(1,0),(0,1),(1,0),(1,1)).$$

To check elementary conjunctions with several predicate symbols for isomorphism and, if they are isomorphic, to find the elementary conjunction $R(x_1, ..., x_n)$ and its unifiers with $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$, the following algorithm ISOM is proposed.

(1) Find the characteristics of elementary conjunctions $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$.

(2) If the characteristics do not coincide, then the formulas are not isomorphic. The algorithm stops its run.

(3) As a formula $R(x_1, ..., x_n)$, take the formula $F_1(a_1, ..., a_n)$. The unifier of formulas $R(x_1, ..., x_n)$ and $F_1(a_1, ..., a_n)$ is an identical substitution $\lambda_{RF_1} = \left|{}^{x_1,\ldots,x_n}_{a_1,\ldots,a_n}\right.$.

(4) Pick the predicate p_i, for which the formula $R_{(p_i)}$ contains the minimal number of variables.[3]

(5) Check the subformulas R_{p_i} and F_{2p_i} for isomorphism using the algorithm ISOM-1 described above.

(6) If R_{p_i} and F_{2p_i} are not isomorphic, then the original formulas $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are not isomorphic. The algorithm stops its run.

Otherwise, using the algorithm ISOM-1 described earlier, find all the unifiers for the subformulas R_{p_i} and F_{2p_i}. During the running of this algorithm, we enter all the values found for variables in the structure *mapping*.

(7) Sort the unifiers in increasing order of the number of variables in them. Organize the cycle 7a–7d by the number of unifiers found.

(a) Apply the unifier to the formula R. Check the consistency of the obtained current values of the formulas R and F_2.

(b) Check the current values of the formulas R and F_2 for contradiction.

(c) If yes, then go to Item 7a (or end the cycle if all the unifiers are checked).

Otherwise, apply the unifier to the formula R. From the current values of the formulas R and F_2, remove the literals with the predicate p_i.

If the stack with unifiers is not empty, then go to Item 7a.

(d) If the current values of the formulas R and F_2 are empty, then the original formulas are isomorphic and all the unifiers are found. The algorithm stops its run.

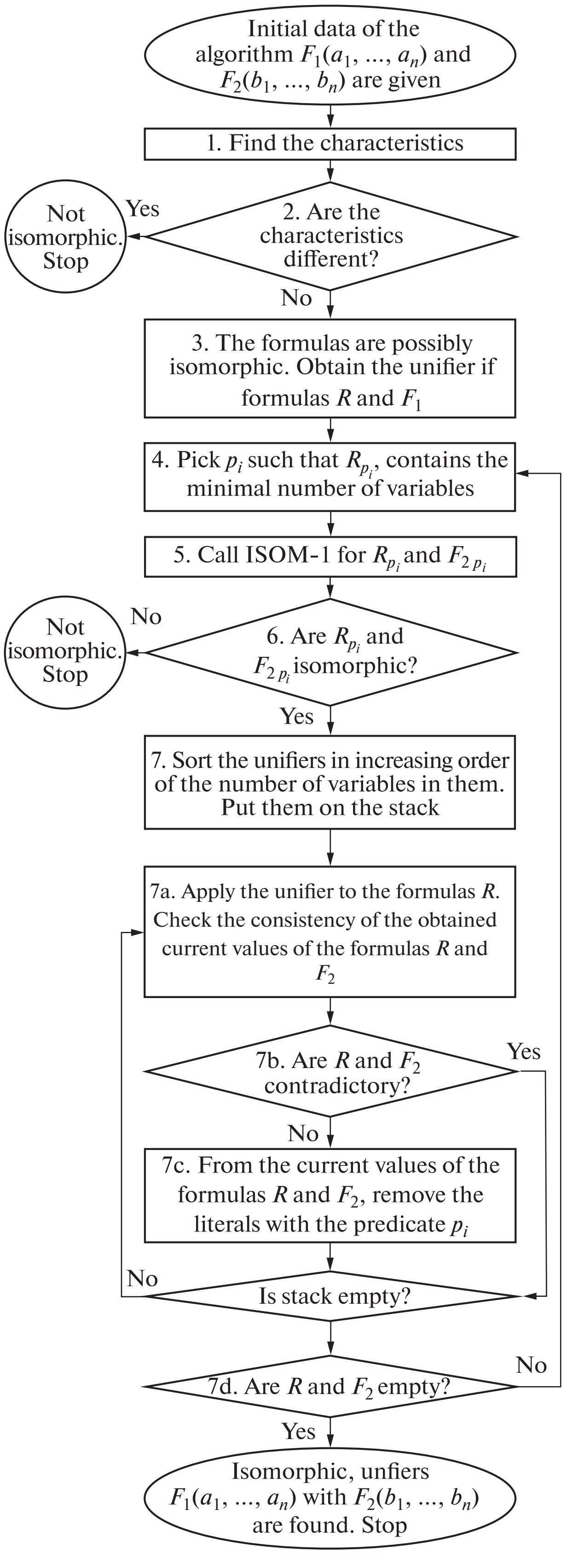
Otherwise, go to Item 4.

The block-scheme of the algorithm ISOM is presented in Fig. 2.

7. ABOUT THE ALGORITHM ISOM COMPLEXITY

All Items of the algorithm, except the Item 5 (call of the algorithm ISOM-1), including cycle 7a–7c by

[3] This choice is due to the fact that the algorithm ISOM-1 used next has an exponential under the number of arguments in formulas checked for isomorphism complexity.



Initial data of the algorithm $F_1(a_1, ..., a_n)$ and $F_2(b_1, ..., b_n)$ are given
1. Find the characteristics
2. Are the characteristics different?
Yes
Not isomorphic. Stop
No
3. The formulas are possibly isomorphic. Obtain the unifier if formulas R and F_1
4. Pick p_i such that R_{p_i}, contains the minimal number of variables
5. Call ISOM-1 for R_{p_i} and F_{2p_i}
6. Are R_{p_i} and F_{2p_i} isomorphic?
No
Not isomorphic. Stop
Yes
7. Sort the unifiers in increasing order of the number of variables in them. Put them on the stack
7a. Apply the unifier to the formulas R. Check the consistency of the obtained current values of the formulas R and F_2
7b. Are R and F_2 contradictory?
Yes
No
7c. From the current values of the formulas R and F_2, remove the literals with the predicate p_i
No
Is stack empty?
7d. Are R and F_2 empty?
No
Yes
Isomorphic, unfiers $F_1(a_1, ..., a_n)$ with $F_2(b_1, ..., b_n)$ are found. Stop

Fig. 2. Block-scheme of the algorithm ISOM.

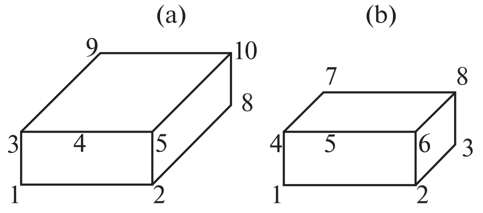


Fig. 3. Images (a) *a* and (b) *b*.

the number of unifiers for R_{p_i} and F_{2p_i}, are performed in no more than a polynomial under the formula notation length number of steps. This is due to the fact that the number of unifiers found by the algorithm ISOM-1 for R_{p_i} and F_{2p_i}, does not exceed n_i^2.

Therefore, the number of executions of the cycle 7a–7c does not exceed n_i^2, and it is polynomial (quadratic) under the length of the notation of subformulas with the predicate p_i. Inside the cycles, the number of operations is also no more than quadratic under the length of R_{p_i} and F_{2p_i} notations.

The number of executions of the cycle 4–7d is equal to the number of predicate symbols in the formula F_2. At the same time, the number of steps in Item 7 is $O(2^{n_i \log n_i})$, where n_i is the number of arguments in the formulas R_{p_i} and F_{2p_i}.

Summing up the obtained estimates of the number of steps, we obtain an estimate of the number of steps of the algorithm ISOM $O\left(\sum_{i=1}^{m} 2^{n_i \log n_i}\right)$, where m is the number of predicate symbols.

8. APPLICATION OF THE ISOM ALGORITHM

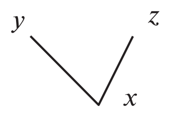
Let there be two contour images (see Fig. 3), composed of segments defined by their ends.

One predicate $P1$ is given, defined as follows (see Fig. 4).

The descriptions of the images *a* and *b* are the following elementary conjunctions

$$\begin{aligned} &F_1(a_1, a_2, a_3, a_4, a_5, a_8, a_9, a_{10}) \\ &= P1(a_1, a_3, a_2) \,\&\, P1(a_2, a_1, a_5) \,\&\, P1(a_2, a_5, a_8) \\ &\&\, P1(a_2, a_1, a_8) \,\&\, P1(a_3, a_4, a_1) \,\&\, P1(a_3, a_5, a_1) \\ &\&\, P1(a_3, a_9, a_4) \,\&\, P1(a_3, a_9, a_5) \,\&\, P1(a_3, a_9, a_1) \\ &\&\, P1(a_5, a_2, a_3) \,\&\, P1(a_5, a_2, a_4) \,\&\, P1(a_5, a_3, a_{10}) \\ &\&\, P1(a_5, a_4, a_{10}) \,\&\, P1(a_5, a_{10}, a_2) \,\&\, P1(a_8, a_2, a_{10}) \\ &\&\, P1(a_9, a_{10}, a_3) \,\&\, P1(a_{10}, a_5, a_9) \,\&\, P1(a_{10}, a_8, a_5) \\ &\&\, P1(a_{10}, a_8, a_9) \end{aligned}$$

and



$$P1(x, y, z) \Leftrightarrow (\angle yxz < \pi)$$

Fig. 4. Predicate $P1$.

$$\begin{aligned}&F_2(b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8)\\&= P1(b_1, b_4, b_2) \,\&\, P1(b_2, b_1, b_6) \,\&\, P1(b_2, b_6, b_3)\\&\& \, P1(b_2, b_1, b_3) \,\&\, P1(b_3, b_2, b_8) \,\&\, P1(b_4, b_5, b_1)\\&\& \, P1(b_4, b_6, b_1) \,\&\, P1(b_4, b_7, b_5) \,\&\, P1(b_4, b_7, b_6)\\&\& \, P1(b_4, b_7, b_1) \,\&\, P1(b_6, b_2, b_5) \,\&\, P1(b_6, b_2, b_4)\\&\& \, P1(b_6, b_5, b_8) \,\&\, P1(b_6, b_4, b_8) \,\&\, P1(b_6, b_8, b_2)\\&\& \, P1(b_7, b_8, b_4) \,\&\, P1(b_8, b_3, b_6) \,\&\, P1(b_8, b_6, b_7)\\&\& \, P1(b_8, b_3, b_7).\end{aligned}$$

Thus, the characteristics of the formulas F_1 and F_2 are

$$\begin{aligned}&\chi(F_1) = (\chi(a_8), \chi(a_1), \chi(a_9), \chi(a_{10}),\\&\chi(a_2), \chi(a_3), \chi(a_5), \chi(a_4))\\&= ((1,2,2),(1,2,3),(1,3,2),(3,2,3),\\&(3,3,2),(5,2,2),(5,3,3),(0,2,2)),\\&\chi(F_2) = (\chi(b_3), \chi(b_1), \chi(b_7), \chi(b_8),\\&\chi(b_2), \chi(b_4), \chi(b_6), \chi(b_5))\\&= ((1,2,2),(1,2,3),(1,3,2),(3,2,3),\\&(3,3,2),(5,2,2),(5,3,3),(0,2,2)).\end{aligned}$$

Moreover, the unifiers of formulas R with F_1

$$\lambda_{RF_1} = \Big|^{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8}_{a_1,a_3,a_2,a_5,a_8,a_4,a_9,a_{10}}$$

and F_2

$$\lambda_{RF_2} = \Big|^{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8}_{b_1,b_4,b_2,b_6,b_3,b_5,b_7,b_8},$$

together with the structure

$$\begin{aligned}&mapping = \{[x_1]:[b_1],[x_2]:[b_4],[x_3]:[b_2],\\&[x_4]:[b_6],[x_5]:[b_3],[x_6]:[b_5],[x_7]:[b_7],[x_8]:[b_8]\}\end{aligned}$$

are also presented.

The algorithm ISOM-1 is implemented in Python.

CONCLUSIONS

The concept of isomorphism is widely used in various fields of mathematics. In particular, in the theory of algorithm complexity, the problem of checking graphs for isomorphism (GI) is widely known. Its NP-completeness has not been proved and a polynomial algorithm for its solution is not known. There is even a term GI-complete problem.

The GI problem, on the one hand, is a subproblem of checking elementary conjunctions of predicate formulas for isomorphism considered in the article, and, on the other hand, these problems are polynomial equivalent [7].

The paper proposes algorithms for checking two elementary conjunctions for isomorphism, that is, for matching up to the names of arguments and the order of literals.

The ISOM-1 algorithm is designed to check for isomorphism of elementary conjunctions with a single predicate symbol. This algorithm can also be used to check the isomorphism of oriented graphs.

The ISOM algorithm is designed to check for isomorphism of elementary conjunctions with several predicate symbols. It essentially uses the ISOM-1 algorithm.

Both of these algorithms are the basis for the currently being developed algorithm for extraction the maximal common (up to the names of the arguments) subformula of two given elementary conjunctions of predicate formulas. The extraction of such subformulas is an important urgent task of finding common properties of complex structured objects (CSO) described by means of the predicate calculus language when solving such problems as

• construction of level description of classes, which allows to significantly reduce the computational complexity of CSO recognition [3, 5];

• creating a logic-predict network that can change its configuration during training [5];

• multiagent description of an object [5], in the case when the agents collecting information about the object do not have data on the true names of parts of objects;

• fuzzy recognition of CSO [8];

• creating an ontology for a set of CSO [8].

ACKNOWLEDGMENTS

Zhou Juan expresses gratitude to China Scholarship Council for the scholarship.

FUNDING

This work was supported by St. Petersburg State University, project no. 94062114.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. R. O. Duda, O. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. (Wiley, New York, 2000).

2. T. M. Kosovskaya, "Estimating the number of steps that it takes to solve some problems of pattern recognition which admit logical description," Vestn. St. Petersburg Univ.: Math. **40**, 287–293 (2007). https://doi.org/10.3103/s1063454107040061
3. T. M. Kosovskaya, "Multilevel descriptions of classes decreasing the number of steps in solving pattern recognition problems described by propositional formulas," Vestn. St. Petersburg Univ.: Math. **41**, 21–27 (2008). https://doi.org/10.3103/s1063454108010056
4. T. M. Kosovskaya and D. A. Petrov, "Extraction of a maximal common sub-formula of predicate formulas for the solving of some artificial intelligence problems," Vestn. S.-Peterb. Univ., Ser. 10: Prikl. Mat. Inf. Protsessy Upr. **13**, 250–263 (2017). https://doi.org/10.21638/11701/spbu10.2017.303
5. T. Kosovskaya, "Predicate calculus as a tool for AI problems solution: Algorithms and their complexity," in *Intelligent System*, Ed. by Ch. Wongchoosuk (InTech, 2018). https://doi.org/10.5772/intechopen.72765
6. T. M. Kosovskaya, "Isomorphism of predicate formulas in artificial intelligence problems," Int. J. Inf. Theories Appl. **26**, 221–230 (2019).
7. T. M. Kosovskaya and N. N. Kosovskii, "Polynomial equivalence of the problems predicate formulas isomorphism and graph isomorphism," Vestn. St. Petersburg Univ., Math. **52**, 286–292 (2019). https://doi.org/10.1134/S1063454119030105
8. T. Kosovskaya, "Fuzzy recognition by logic-predicate network," Adv. Sci., Technol. Eng. Syst. **5** (4), 686–699 (2020). https://doi.org/10.25046/AJ050482
9. T. M. Kosovskaya and N. N. Kosovskii, "Extraction of common properties of objects for creation of a logic ontology," Vestn. S.-Peterb. Univ., Ser. 10: Prikl. Mat. Inf. Protsessy Upr. **18**, 37–51 (2022). https://doi.org/10.21638/11701/spbu10.2022.103
10. N. J. Nilson, *Problem-Solving Methods in Artificial Intelligence* (McGraw-Hill, New York, 1971).
11. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, Upper Saddle River, N.J., 2009).

Tatiana Kosovskaya. Doctor of Physical and Mathematical Sciences, Professor of St. Petersburg State University. Area of interest: application of methods of mathematical logic to problems of artificial intelligence, complexity theory of algorithms.

Juan Zhou. PhD student at the Faculty of Mathematics and Mechanics, St Petersburg State University. Scientific interests: mathematical logic, artificial intelligence.