Algorithm for Extraction Common Properties of Objects Described in the Predicate Calculus Language with Several Predicate Symbols

Kosovskaya Tatiana
St Petersburg University
St. Petersburg, Russia
e-mail: kosovtm@gmail.com

Zhou Juan
St Petersburg University
St. Petersburg, Russia
e-mail: st103098@student.spbu.ru

https://doi.org/10.62343/csit.2024.9

***Abstract* — When solving artificial intelligence problems connected with 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 the extraction of common properties of objects described in the predicate calculus language with predicate symbols. The first of the algorithms extracts maximal common subformulas for elementary conjunctions containing 2 predicate symbols. The second algorithm extracts maximal common subformulas for elementary conjunctions with several predicate symbols. Estimates of their time complexity are given for both algorithms. Both algorithms are implemented in Python.**

***Keywords* — Predicate formulas, isomorphism of predicate formulas, complex structured object, maximal common subformula.**

I. INTRODUCTION

In artificial intelligence problems, connected with the study of complex structured objects, described by the properties of their elements and the relationships between these elements, it is convenient to use predicate calculus formulas. To describe the classes of such objects, it is necessary to highlight their common properties.

An algorithm for extraction maximal common property of two objects described by means of a single predicate is proposed in [1]. An estimate of the computational complexity is proved for this algorithm.

Presented here two algorithms aim to extract maximal common subformulas for elementary conjunctions, which contain two and several different predicate symbols, respectively.

II. WHY NECESSARY DEFINITIONS

Definition 1 [2]. *Two elementary conjunctions of atomic formulas of predicate calculus $F1(a_1,\dots,a_n)$ and $F2(b_1,\dots,b_n)$ are called isomorphic*

$$F1(a_1,\dots,a_n) \sim F2(b_1,\dots,b_n),$$

if there is such an elementary conjunction $R(x_1,\dots,x_n)$ and substitutions of arguments $a_{i_1},\dots,a_{i_n}$ and $b_{j_1},\dots,b_{j_n}$ of formulas $F1(a_1,\dots,a_n)$ and $F2(b_1,\dots,b_n)$ accordingly, instead of all occurrences of variables $x_1,\dots,x_n$ of the formula $R(x_1,\dots,x_n)$, that the results of these substitutions $R(a_{i_1},\dots,a_{i_n})$ and $R(b_{j_1},\dots,b_{j_n})$ coincide up to the order of literals with the formulas $F1(a_1,\dots,a_n)$ and $F2(b_1,\dots,b_n)$, respectively.

The resulting substitutions $\lambda 1 = \{x_1: a_{i_1},\dots, x_n: a_{i_n}\}$ and $\lambda 2 = \{x_1: b_{i_1},\dots, x_n: b_{i_n}\}$ are called unifiers of formulas $F1(a_1,\dots,a_n)$ and $F2(b_1,\dots,b_n)$ with the formula $R(x_1,\dots,x_n)$ respectively.

The formula $R(x_1,\dots,x_n)$ is below referred to as the MCF (Maximum Common sub-Formula).

Definition 2 [1]. *Two substitutions are called contradictory if two different constants a_1 and a_2 are found for the same variable x, i.e., $\{x: a_1, x: a_2\}$, or for different variables x_1 and x_2, the same constant a is found, i.e., $\{x_1: a, x_2: a\}$.*

Definition 3 [1]. *Let $R(x_1,\dots,x_n)$ and $F(b_1,\dots,b_n)$ be two elementary conjunctions of predicate formulas, with $R(x_1,\dots,x_n)$ containing only variables as arguments.*

Substitution $\{\overline{x}: \overline{b}\}$, where $\overline{x}$ is a list of some variables from $R(x_1,\dots,x_n)$, $\overline{b}$ is a list of some different constants from $F(b_1,\dots,b_n)$, is called a partial unifier of the formulas $R(x_1,\dots,x_n)$ and $F(b_1,\dots,b_n)$ if the result of applying this substitution to the formula $R(x_1,\dots,x_n)$ contains a subformula that coincides up to the order of literals with some subformula $F(b_1,\dots,b_n)$. Below all unifiers will be partial.

Definition 4 [3]. *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 [3]. *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.*

For further description of the algorithms some notations will be required.

Notation 1. *A number of substitutions in the unifier λ is called an unifier length and is denoted by $||\lambda||$.*

Notation 2. R_i^* is a list in ascending order of unifier lengths, containing pairs $(R_i(\overline{x}_i), \lambda 2_i)$ for all MCF $R_i(\overline{x}_i)$ of subformulas $\left(F1_i(\overline{a}_i), F2_i(\overline{b}_i)\right)$ and their unifier $\lambda 2_i$ with the corresponding subformulas of formula $F2_i(\overline{b}_i)$.

Note that when defining the formula $R_i(\overline{x}_i)$, it is always possible to organize the numbering of these variables in the list of variables $\overline{x}_i$ so that all substitutions in the unifier $\lambda 1_i$ have the form $\{x_\alpha: a_\alpha\}$ for all variables x_α included in the MCF $R_i(\overline{x}_i)$. Therefore, the unifier MCF $R_i(\overline{x}_i)$ with $F1_i(\overline{a}_i)$ will not be written out below.

Notation 3. $R_{i,j}^*$ – is a list in ascending order of unifier lengths, containing pairs $\left(R_{i,j}(\overline{x}_{i,j}), \lambda 2_{i,j}\right)$ for all MCF

$R_{i,j}(\overline{x}_{i,j})$ of subformulas $\left(F1_{i,j}(\overline{a}_{i,j}), F2_{i,j}(\overline{b}_{i,j})\right)$ and their unifier $\lambda 2_{i,j}$ with the corresponding subformulas of formula $F2_{i,j}(\overline{b}_{i,j})$.

Notation 4. R^* – is a resulting list in ascending order of unifier lengths, containing pairs $(R(\overline{x}), \lambda 2)$ for all MCF $R(\overline{x})$ of subformulas $\left(F1(\overline{a}), F2(\overline{b})\right)$ and their unifier $\lambda 2$ with the corresponding subformulas of formula $F2(\overline{b})$.

III. ALGORITHM MCF2 FOR EXTRACTING COMMON PROPERTIES OF OBJECTS DESCRIBED IN THE PREDICATE CALCULUS LANGUAGE WITH TWO PREDICATE SYMBOLS

Let a pair of elementary conjunctions of atomic predicate formulas $F1(a_1, \ldots, a_m)$ and $F2(b_1, \ldots, b_n)$ with predicate symbols P_i, P_j and constants $a_1, \ldots, a_m$ and $b_1, \ldots, b_n$ as arguments be given[1]. The names of all arguments in each literal are different.

The following algorithm **MCF2** is proposed to extract the maximal elementary conjunction $R(x_1, \ldots, x_s)$ $(s \leq \min(m, n))$ for which $F1(a_1, \ldots, a_m)$ and $F2(b_1, \ldots, b_n)$ have subformulas isomorphic to $R(x_1, \ldots, x_s)$.

1. Create 2 pairs of maximal subformulas from $F1(a_1, \ldots, a_m)$ and $F2(b_1, \ldots, b_n)$, containing only a single predicate symbol: $\left(F1_i(\overline{a}_i),\ F2_i(\overline{b}_i)\right)$ - with predicate P_i and $\left(F1_j(\overline{a}_j),\ F2_j(\overline{b}_j)\right)$ - with predicate P_j[2]. That is, $F1(a_1, \ldots, a_m) = F1_i(\overline{a}_i)\ \&\ F1_j(\overline{a}_j)$, $F2(b_1, \ldots, b_n) = F2_i(\overline{b}_i)\ \&\ F2_j(\overline{b}_j)$.
2. Extract R_i^* using the algorithm **MCF1** for a pair of formulas $\left(F1_i(\overline{a}_i), F2_i(\overline{b}_i)\right)$. Do the same with $\left(F1_j(\overline{a}_j), F2_j(\overline{b}_j)\right)$ and obtain the list R_j^*.
3. In a nested loop over the lists R_i^* and R_j^*, check the unifiers $\lambda 2_i$ and $\lambda 2_j$ for inconsistency.
   If the unifiers are consistent, unify them and connect the current common subformulas with the sign &. The obtained formula $R_{i,j}(\overline{x}_{i,j})$ with two predicate symbols P_i and P_j, which defines the MCF of $F1(a_1, \ldots, a_m)$ and $F2(b_1, \ldots, b_n)$ and their unifiers, is added to $R_{(i,j)}^*$.
   If the unifiers are inconsistent, then go to the next pair of pairs in the lists R_i^* and R_j^*, i.e., go to the next step of the loop.

A block diagram of the algorithm **MCF2** is shown in Fig. 1.

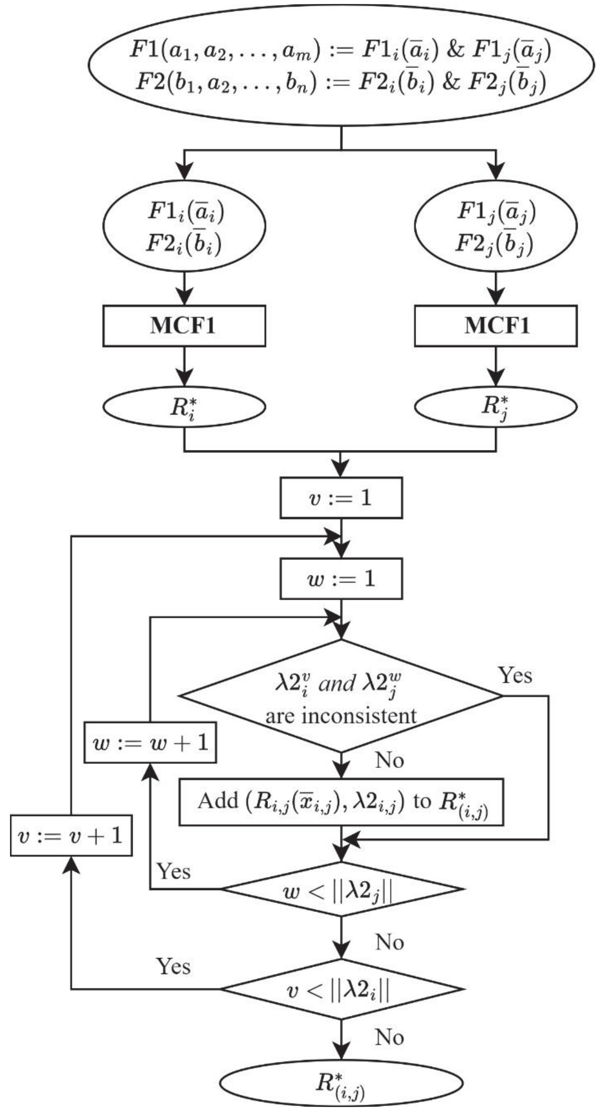

Fig. 1. Algorithm **MCF2** block diagram.

IV. ABOUT THE ALGORITHM MCF2 COMPLEXITY

The number of steps in items 1-2, the complexity of the algorithm **MCF1** implemented to $F1_i(\overline{a}_i)$ and $F2_i(\overline{b}_i)$, is $O(n_i^{2n_i})$, where n_i is maximal number of arguments in these formulas [1].

The complexity of checking for consistency of $\lambda 2_i$ with $\lambda 2_j$ (item 3) is $O(\sum_{v=1}^{||\lambda 2_i||} \sum_{w=1}^{||\lambda 2_j||}(\,||\lambda 2_i||\,||\lambda 2_j||)) = O(||\lambda 2_i||^2 ||\lambda 2_j||^2) \leq O(n_i^2 n_j^2) \leq O(n^4)$, where n_i, n_j are the numbers of arguments in the formulas F_i, F_j, respectively.

The main contribution to the **MCF2** algorithm's computational complexity estimation comes from the implementation of item 2, namely, **MCF1** implemented consistently to pairs $\left(F1_i(\overline{a}_i), F2_i(\overline{b}_i)\right)$. Computational complexity of the algorithm **MCF2** is $O(n^{2n})$, where n is the maximal number of arguments in subformulas with a single predicate symbol and is not greater than the number of arguments in $F1_i(\overline{a}_i)$ and $F2_i(\overline{b}_i)$.

[1] The names of constants in different formulas may coincide, and constants with the same names in different formulas may be used as names of different object elements and stand in different places.

[2] Here $\overline{a}_i$ and $\overline{b}_i$ are lists of all arguments that are included in maximal subformulas with predicate P_i. Similarly for the predicate P_j.

V. ALGORITHM MCFN FOR EXTRACTING COMMON PROPERTIES OF OBJECTS DESCRIBED IN THE PREDICATE CALCULUS LANGUAGE WITH SEVERAL PREDICATE SYMBOLS

Let formulas $F1(a_1, \dots, a_m)$ and $F2(b_1, \dots, b_n)$ be elementary conjunctions of predicate formulas with l predicate symbols $P_1, \dots, P_l$, and literals with the same predicate symbol are consecutive.

Consider that the numbering of literals is ordered so that if $i < j$, then the minimum number of arguments for all occurrences of the predicate P_i in $F1(a_1, \dots, a_m)$ and $F2(b_1, \dots, b_n)$ does not exceed the minimum number of arguments for all occurrences of the predicate P_j in $F1(a_1, \dots, a_m)$ and $F2(b_1, \dots, b_n)$.

For example, if

$$F1(a_1, \dots, a_m) = \underbrace{P_i(\cdot)\ \&\ \dots P_i(\cdot)}_{n1\ arguments}\ \&\ \underbrace{P_j(\cdot)\ \&\ \dots P_j(\cdot)}_{m1\ arguments}$$

$$F2(b_1, \dots, b_n) = \underbrace{P_i(\cdot)\ \&\ \dots P_i(\cdot)}_{n2\ arguments}\ \&\ \underbrace{P_j(\cdot)\ \&\ \dots P_j(\cdot)}_{m2\ arguments}$$

and $i < j$, then $\min\{n1, n2\} \le \min\{m1, m2\}$[3].

Algorithm **MCFn** is as follows:

1) $R^* := \emptyset$.
2) Organize the loop by $i = 1, \dots, l/2$.[4]
   a) For $F1(a_1, \dots, a_m)$ and $F2(b_1, \dots, b_n)$, generate two pairs of subformulas $\left(F1_{2i-1}(\overline{a}_{2i-1}),\ F2_{2i-1}(\overline{b}_{2i-1})\right)$ and $\left(F1_{2i}(\overline{a}_{2i}),\ F2_{2i}(\overline{b}_{2i})\right)$, containing the single predicate symbol P_{2i-1} and P_{2i}, respectively.
   b) For pairs of subformulas $\left(F1_{2i-1}(\overline{a}_{2i-1}),\ F2_{2i-1}(\overline{b}_{2i-1})\right)$ and $\left(F1_{2i}(\overline{a}_{2i}),\ F2_{2i}(\overline{b}_{2i})\right)$ using the algorithm **MCF1** extract the lists R^*_{2i-1} and R^*_{2i}.
   c) For each of the obtained pairs from R^*, check $\lambda 2_{2i-1}$ with $\lambda 2$, then $\lambda 2_{2i}$ with $\lambda 2$ for consistency.
   If the unifiers are consistent, then
      I. call the algorithm **MCF2** for $(F1_{2i-1}, F2_{2i-1})$ and $(F1_{2i}, F2_{2i})$, get a list $R^*_{(2i-1,\,2i)}$ of pairs, such as $\left(R_{2i-1,\,2i}(\overline{x}_{2i-1,2i}),\ \lambda 2_{2i-1,2i}\right)$[5];
      II. merge $\lambda 2$ and $\lambda 2_{2i-1,2i}$, attach $R(\overline{x})$ and $R_{2i-1,2i}(\overline{x}_{2i-1,2i})$. Write the resulting formula $R(\overline{x}) := R(\overline{x}) \cup R_{2i-1,2i}(\overline{x}_{2i-1,2i})$, specifying the MCF of formulas $F1(a_1, \dots, a_m)$ and $F2(b_1, \dots, b_n)$, and their unifiers in R^*.
   Otherwise, go to the next step in the cycle.

A block diagram of the algorithm **MCFn** is shown in Fig. 2.

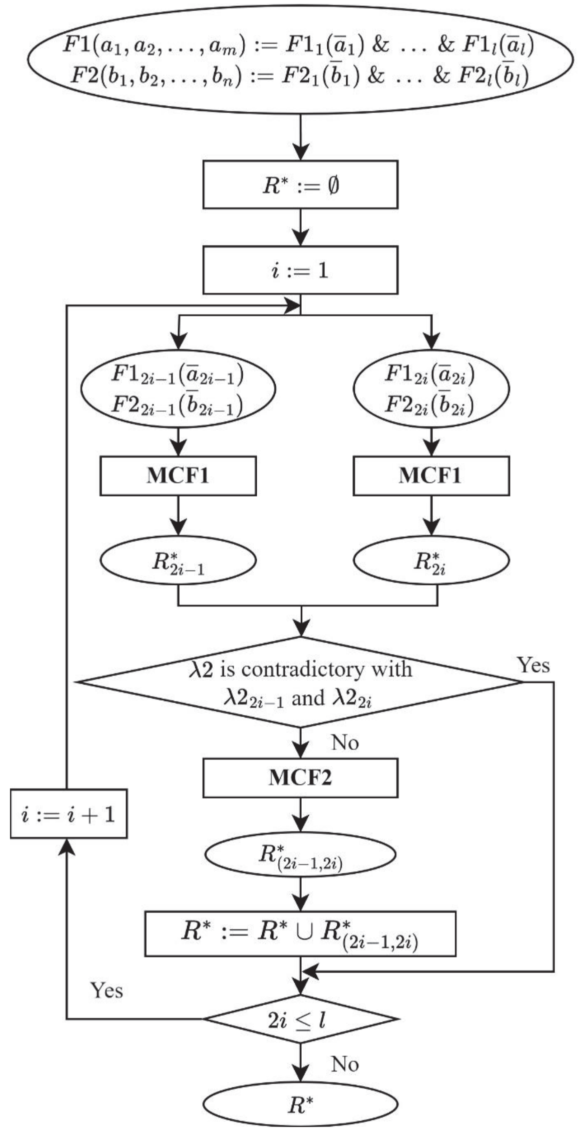

Fig. 2. Algorithm **MCFn** block diagram.

VI. ABOUT THE ALGORITHM MCFN COMPLEXITY

The number of steps in items 1-2b is the complexity of **MCF1** implemented to pairs $\left(F1_{2i-1}(\overline{a}_{2i-1}),\ F2_{2i-1}(\overline{b}_{2i-1})\right)$ and $\left(F1_{2i}(\overline{a}_{2i}),\ F2_{2i}(\overline{b}_{2i})\right)$. It is $O(n_{2i-1}^{2n_{2i-1}} + n_{2i}^{2n_{2i}}) \le O(n^{2n})$, where n_{2i-1}, n_{2i}, n are the maximal numbers of arguments in subformulas with a single predicate symbol P_{2i-1}, P_{2i} and in the initial formulas respectively.

In item 2c, the complexity of checking for consistency of $\lambda 2_i$ with $\lambda 2_{2i-1}$, $\lambda 2_i$ with $\lambda 2_{2i}$, is $O(||\lambda 2_{2i-1}||||\lambda 2_{2i}||) \le O(n_{2i-1} n_{2i}) \le O(n^2)$.

In items 2ci-2cii, the complexity **MCF2**, is $O(n^{2n})$.

The number of executions of **MCFn** is equal to half the number of predicate symbols $l/2$. At the same time, the number of steps in Items 1-2cii is $O(n^{2n}) + O(n^2) + O(n^{2n})$.

---

[3] The order of predicate symbols may depend on the specifics of the initial data. For example, if in each formula there are one or two occurrences of a predicate symbol containing more than half of all variables, then by assigning the number 1 to this predicate, we can find a partial unifier for more than half of the variable values in one step (in the worst case, in 4 steps).

[4] If the initial number of predicates l is odd, l will be increased by one ($l := l + 1$), and the elementary conjunctions with the fictive predicate will be assumed to be empty ($F1_l(\overline{a}_l) := \emptyset, F2_l(\overline{b}_l) := \emptyset$).

[5] $\lambda 2_{2i-1}$ and $\lambda 2_{2i}$ only contain those partial unifiers that do not contradict the unifiers of $\lambda 2$.

Summing up the obtained estimates of the number of steps, we obtain an estimate of the number of steps of the algorithm **MCFn** $O(\sum_{i=1}^{l/2}(n^{2n}+n^2+n^{2n}))=O(n^{2n})$.

Thus, the main contribution to the **MCFn** algorithm's computational complexity estimation also comes from the implementation of **MCF1**, whose computational complexity is $O(n^{2n})$.

VII. CONCLUSION

The paper presents two algorithms **MCF2** and **MCFn** for extraction maximal common subformulas (up to the precision of argument names) of two elementary conjunctions. The implementation was carried out in the Python [4] programming language.

Extraction of such subformulas is an important actual task of searching for common properties of complex structured objects (CSO) described in the predicate calculus language, when solving such problems as

- level descriptions of classes for significantly decreasing the computational complexity of CSO recognition [5,6];
- fuzzy recognition of CSO [7];
- creation of a logic ontology for CSO [8].

ACKNOWLEDGMENT

Kosovskaya T. M. acknowledges Saint-Petersburg State University for a research project 95438429.

Zhou Juan acknowledges the support of the China Scholarship Council program (Project ID: 202108620001).

REFERENCES

1. J. Zhou and T. M. Kosovskaya. "Algorithm for extraction common properties of objects described in the predicate calculus language with a single predicate symbol", *Vestnik of Saint Petersburg University. Mathematics. Mechanics. Astronomy*, vol. 11 (69), issue 4, 2024. (In print) DOI: 10.21638/spbu01.2024.409
2. 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", *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, vol. 13, no. 3, pp. 250–263, 2017. DOI: 10.21638/11701/spbu10.2017.303
3. T. M. Kosovskaya and J. Zhou, "Algorithms of Isomorphism of Elementary Conjunctions Checking", *Pattern Recognition and Image Analysis*, vol. 34, no. 1, pp. 102–109, 2024. DOI: 10.1134/S1054661824010103
4. E. Matthes, *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*, 3rd Edition. No Starch Press, San Francisco, 2023.
5. T. Kosovskaya, *Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity*. Chapter 3 in Intelligent System. Open access peer-reviewed Edited volume. Edited by Chatchawal Wongchoosuk Kasetsart University, pp. 1 – 20, 2018. [Online]. Available: https://www.intechopen.com/chapters/58698
6. T. M. Kossovskaya, "Level descriptions of classes for decreasing step number of pattern recognition problem solving described by predicate calculus formulas", *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, vol. 10, no. 2, pp. 62–70, 2008.
7. T. Kosovskaya, "Fuzzy Recognition by Logic-Predicate Network", *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 4, pp. 686–699, 2020. https://doi.org/10.25046/aj050482
8. T. M. Kosovskaya and N. N. Kosovskii, "Extraction of common properties of objects for creation of a logic ontology", *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, vol. 18, no. 1, pp. 37–51, 2022. https://doi.org/10.21638/11701/spbu10.2022.103