УДК 004.93.51 Вестник СПбГУ. Прикладная математика... 2017. Т. 13. Вып. 3

Т. М. Косовская, Д. А. Петров

ВЫДЕЛЕНИЕ НАИБОЛЬШЕЙ ОБЩЕЙ ПОДФОРМУЛЫ ПРЕДИКАТНЫХ ФОРМУЛ ДЛЯ РЕШЕНИЯ РЯДА ЗАДАЧ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Санкт-Петербургский государственный университет, Российская Федерация,
199034, Санкт-Петербург, Университетская наб., 7–9

В задачах искусственного ителлекта, в которых объект представлен как множество составляющих его элементов и характеризуется свойствами этих элементов и отношениями между ними, язык исчисления предикатов является адекватным для описания объектов и классов объектов. Так. формализованные задачи оказываются NP-полными или NP-трудными. Примером такой NP-трудной задачи является задача анализа сложного объекта, представленного как множество его элементов и описываемого набором атомарных формул с предикатами, задающими свойства этих элементов и отношения между ними. Для решения данных задач рассматривается задача выделения наибольшей общей с точностью до имен переменных подформулы двух элементарных конъюнкций атомарных предикатных формул. Выделение таких подформул позволяет свести задачу распознавания объекта к серии аналогичных задач с меньшей длиной исходных данных за счет построения многоуровневого описания классов, существенно уменьшающего показатель степени в экспоненциальной оценке числа шагов решения NP-трудной задачи проверки принадлежности объекта этому классу. Приводится алгоритм выделения наибольшей общей с точностью до имен переменных подформулы двух элементарных конъюнкций. Доказываются оценки числа шагов его работы. Анализируется время применения его реализации в зависимости от структуры исходных данных. Анализ числа шагов работы алгоритма показывает, что предложенный алгоритм может быть успешно применен для уменьшения вычислительной сложности многих задач искусственного интеллекта, формализуемых средствами исчисления предикатов. Приводятся результаты численных экспериментов, показывающих влияние структуры исходных данных на время работы алгоритма. Так, например, при увеличении количества аргументов у предикатов значительно снижается время работы за счет уменьшения глубины рекурсии. Также существенное влияние оказывает количество различных предикатных символов, участвующих в записи формулы. При больших исходных данных особенно заметна эффективность предложенного алгоритма, отсекающего «ветви», заведомо не приводящие к успеху. Сильное влияние на время работы алгоритма оказывает распределение переменных в качестве аргументов предикатов. Результаты показывают значительное снижение числа шагов работы алгоритма при неравномерном распределении переменных. В процессе работы описанного алгоритма возникает задача проверки совпадения с точностью до имен переменных (изоморфизма) двух элементарных конъюнкций. Решение этой задачи имеет наибольшую вычислительную сложность среди прочих шагов алгоритма. Предложен полиномиальный алгоритм проверки изоморфизма предикатных формул, правильность работы которого превышает 99,5%. Алгоритм не является абсолютно точным. Доказано, что если он дает отрицательный ответ, то формулы не изоморфны. Если алгоритм дает положительный ответ, то результаты численных экспериментов в 99,5% случаев показали изоморфность формул. Библиогр. 14 назв. Табл. 2.

Ключевые слова: искусственный интеллект, исчисление предикатов, алгоритмическая сложность, изоморфизм предикатных формул.

Косовская Татьяна Матвеевна — доктор физико-математических наук, профессор; kosovtm@gmail.com

Петров Дмитрий Андреевич — студент; imsohay@gmail.com

Kosovskaya Tatiana Matveevna — doctor of physical and mathematical sciences, professor; kosovtm@gmail.com

Petrov Dmitrii Andreevich — student; imsohay@gmail.com

 DOI: 10.21638/11701/spbu10.2017.303

T. M. Kosovskaya, D. A. Petrov

EXTRACTION OF A MAXIMAL COMMON SUB-FORMULA OF PREDICATE FORMULAS FOR THE SOLVING OF SOME ARTIFICIAL INTELLIGENCE PROBLEMS

St. Petersburg State University, 7–9, Universitetskaya nab., St. Petersburg, 199034, Russian Federation

If an investigated object in an Artificial Intelligence problem is regarded as a set of its elements and is characterized by properties of these elements and relations between them, then a predicate calculus language is an adequate one for the description of an object and classes of objects. Such a way formalized problems are NP-complete or NP-hard. Example of such an NP-hard problem is the problem of the analysis of a complex object presented as a set of its elements and described by a set of atomic formulas with the predicates setting properties of these elements and relation between them. To solve such a problem, a problem of extraction of a maximal common up to the names of variables sub-formula of two elementary conjunctions of atomic predicate formulas is under consideration in the paper. Extraction of these sub-formulas allows us to reduce a recognition problem to a series of analogous problems with the less input data length by means of construction a level description of classes, which essentially decreases an exponent in the exponential upper bound for a number of solution algorithm steps for NP-hard problem of checking whether an object belongs to this class. An algorithm of extraction of a maximal common up to the names of variables sub-formula of two elementary conjunctions of atomic predicate formulas is presented in the paper. Bounds on number of the algorithm run number of steps are proved. The time of its implementation in dependence on the input data structure is analyzed. The analysis of the number of algorithm steps shows that the proposed algorithm may be successfully implemented in order to decrease computational complexity of many Artificial Intelligence problems which are formalized by means of predicate calculus. The results of numerical experiments show the influence of input data structure on the algorithm run time. For example, the increasing of amount of predicate arguments involves essential decreasing of the algorithm run time, which is a consequence of the recursion depth decreasing. The number of different predicate symbols occurred in the formula notation also essentially influences on the algorithm run time. At larger input data the effectiveness of the offered algorithm, cutting obviously unsuccessful "brunches", is especially noticeable. The strong impact on the algorithm run time is exerted by distribution of variables as arguments of predicates. Results show its considerable decreasing at a nonuniform distribution of variables. During the described algorithm run a problem of checking the coincidence up to the names of variables (isomorphism) of two elementary conjunctions appears. This problem has the greatest computational complexity among the other problems arising on other steps of the algorithm. A polynomial algorithm of checking whether two elementary conjunctions of atomic predicate formulas coincide up to the names of variables (are isomorphic) is offered. The algorithm is not a precise one. The correctness of this algorithm implementation exceeds 99,5%. It is proved that if the algorithm gives a negative answer then the formulas are not isomorphic. If the algorithm gives a positive answer then the results of numerical experiments show that 99,5% pairs of formulas are isomorphic. Refs 14. Tables 2.

Keywords: Artificial Intelligence, predicate calculus, algorithmic complexity, isomorphism of predicate formulas.

1. Введение. Формализация задач искусственного интеллекта на языке исчисления предикатов была предложена еще в 70-е годы XX в. [1]. Пока теория сложности алгоритмов и, в частности, понятие NP-полной задачи, подробно описанной в [2], не получила широкого распространения в научной среде, полагали, что алгоритмы автоматического доказательства теорем в исчислении предикатов позволят легко решать так называемые формализованные задачи. Но многие из них оказались NP-трудными [3] или даже в общем случае алгоритмически неразрешимыми (например, задача нахождения таких воздействий на объект, в результате которых объект удовлетворяет заданному целевому условию [4]).

При решении задач искусственного интеллекта часто используется язык исчисления высказываний, позволяющий описывать и распознавать объекты, характеризующиеся конечным набором бинарных (или даже конечнозначных) признаков. Данный подход к решению задач искусственного интеллекта получил название логико-алгебраического [5]. При нем вычислительная сложность решения задач распознавания полиномиальна относительно длины записи описания распознаваемого объекта и описаний классов [6] и имеет слабое отношение к NP-полной задаче ВЫПОЛНИМОСТЬ [2], кроме того, что они обе применяют язык пропозициональных формул.

Однако при распознавании сложного объекта (или выделении его на сложной сцене), который характеризуется не своими глобальными признаками, но сам состоит из элементов, описывающихся набором свойств и отношениями между этими элементами, полезным оказывается язык исчисления предикатов. Такой подход удобно называть логико-предметным. В частности, эти задачи имеют много общего с NP-полной задачей, относящейся к базам данных, КОНЪЮНКТИВНЫЙ БУЛЕВСКИЙ ЗАПРОС [2]. При этом NP-трудность возникающих задач [3] вполне согласуется с тем, что при моделировании их исходных данных с помощью пропозициональных (булевых) переменных длина записи новых описаний экспоненциально зависит от длины записи исходных данных при моделировании задачи с помощью формул исчисления предикатов [7].

Анализ оценок сложности различных алгоритмов решения задач, использующих формализацию на языке исчисления предикатов, позволил предложить многоуровневые описания классов, при определенных условиях существенно уменьшающие число шагов решения таких задач [8]. Этот подход основан на широко распространенном способе выделения из исходной задачи подзадач с меньшей длиной записи исходных данных.

Проблема заключалась в том, что для построения предлагаемого многоуровневого описания классов необходимо выделить достаточно «короткие подформулы» исходных целевых формул. Однако они могут не являться в точности общими подформулами исходных формул, но станут таковыми после некоторой подстановки. Тогда мы будем говорить об общей с точностью до имен переменных подформуле двух предикатных формул, точнее, об изоморфных подформулах.

Определение 1. *Две элементарные конъюнкции A и B атомарных предикатных формул называются изоморфными, если существуют такая элементарная конъюнкция C и такие подстановки $\lambda_{A,C}$ и $\lambda_{B,C}$ различных переменных из формулы C вместо разных аргументов формул A и B соответственно, что результаты этих подстановок $A\lambda_{A,C}$ и $B\lambda_{B,C}$ совпадают с формулой C с точностью до порядка литералов.*

При этом подстановки $\lambda_{A,C}$ и $\lambda_{B,C}$ называются унификаторами формул A и B с формулой C.

Нетрудно видеть, что отношение изоморфизма формул есть отношение эквивалентности.

Две формулы $A = p(a,b)\ \&\ p(b,a)\ \&\ q(b,a,c)$ и $B = p(b,a)\ \&\ p(b,d)\ \&\ q(a,b,d)$ имеют подформулы, изоморфные $C = p(u,v)\ \&\ q(v,u,w)$, которая с точностью до имен переменных есть общая подформула формул A и B, так как при замене переменных u, v, w на a, b, c соответственно она превратится в $p(a,b)\ \&\ q(b,a,c)$ — подформулу формулы A, а при замене переменных u, v, w на b, a, d — в $p(b,a)\ \&\ q(a,b,d)$ — подформулу формулы B.

Именно описанию алгоритма выделения наибольшей общей с точностью до имен переменных подформулы двух элементарных конъюнкций атомарных предикатных формул и построению общих унификаторов (подстановок, в результате которых эта «подформула» действительно станет подформулами заданных формул) и алгоритма проверки изоморфности двух элементарных конъюнкций (дающего верный результат более чем в 99.9% случаев) посвящена настоящая статья.

2. Постановка рассматриваемых задач. Пусть исследуемый объект представлен как множество своих элементов $\omega = \{\omega_1, \ldots, \omega_t\}$. На ω задан набор предикатов $p_1, \ldots, p_n$, характеризующих свойства элементов ω и отношения между ними. Логическим описанием $S(\omega)$ объекта ω называется множество всех атомарных формул или их отрицаний, истинных на ω. Множество всех объектов разбито на классы $\Omega = \cup_{k=1}^{K} \Omega_k$. Логическим описанием класса Ω_k называется формула $A_k(\overline{x})$, заданная в виде дизъюнкции элементарных конъюнкций, такая что если $A_k(\overline{\omega})$ истинна, то $\omega \in \Omega_k$.

Здесь и далее посредством $\overline{x}$ обозначается список элементов конечного множества x, соответствующий некоторой перестановке номеров его элементов. Тот факт, что элементами списка $\overline{x}$ являются элементы множества y, будем записывать в виде $x \subseteq y$.

С помощью построенных описаний объектов и классов предлагается решать следующие задачи.

З а д а ч а и д е н т и ф и к а ц и и. *Проверить, удовлетворяет ли объект ω или его часть описанию класса $A_k(\overline{x})$, и предъявить эту часть объекта.*

З а д а ч а к л а с с и ф и к а ц и и. *Найти все такие номера k, что верна формула $A_k(\overline{\omega})$.*

З а д а ч а а н а л и з а. *Найти и классифицировать все части τ объекта ω, для которых $A_k(\overline{\tau})$.*

Для того чтобы записать, что значения для переменных списка $\overline{x}$, удовлетворяющие формуле $A(\overline{x})$, различны, вместо формулы

$$\exists x_1 ... \exists x_m (\&_{i=1}^{m} \&_{j=i+1}^{m} (x_i \neq x_j) \ \& \ A(x_1, ..., x_m))$$

будет использоваться обозначение

$$\exists \overline{x}_{\neq} A(\overline{x}).$$

Решение задач идентификации, классификации и анализа для распознавания сложного объекта сведено в [9] к доказательству соответственно логических следований

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

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

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

Строго говоря, в формулах (1)–(3) вместо квантора существования и дизъюнкции следовало бы писать «при каких наборах различных значений $\overline{x}$» ($?\overline{x}_{\neq}$) и «при каких значениях k» ($?k_{k=1}^{M}$) соответственно

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

$$S(\omega) \Rightarrow (?k_{k=1}^{M})A_k(\overline{\omega}), \tag{2'}$$

$$S(\omega) \Rightarrow (?k_{k=1}^{M})(?\overline{x}_{\neq})\ A_k(\overline{x}), \tag{3'}$$

но как переборный алгоритм, так и алгоритмы построения вывода в исчислении предикатов (например, в секвенциальном исчислении предикатов или доказательстве методом резолюций для исчисления предикатов) при доказательстве логических следований (1)–(3) не только отвечают на вопрос «*существует ли ... ?*», но и предъявляют значения для переменных [10].

Заметим, что для того чтобы уметь доказывать (1)–(3), достаточно доказывать логическое следование

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

где $A(\overline{x})$ — элементарная конъюнкция атомарных формул и их отрицаний. В [6, 11] доказаны оценки числа шагов алгоритмов, решающих задачу (4), а также задачи (1)–(3). Эти оценки имеют экспоненциальный от длины записи формулы $A(\overline{x})$ вид. Для алгоритма полного перебора в показателе оценки находится количество переменных формулы $A(\overline{x})$, а для алгоритмов, основанных на построении вывода в исчислении предикатов, — количество атомарных формул, входящих в $A(\overline{x})$. Там же доказана NP-полнота задач (1)–(3) и, следовательно, NP-трудность задач (1′)–(3′).

3. Многоуровневое описание классов. Рассматриваются объекты, структура которых позволяет выделить достаточно простые их части и дать описание объекта в терминах свойств этих частей и отношений между ними, что, в частности, можно сделать, выделяя «часто» встречающиеся подформулы $P_i^1(\overline{y}_i^1)$ формул $A_k(\overline{x})$ «небольшой сложности». При этом записывается система равносильностей вида $p_i{}^1(y_i^1) \Leftrightarrow P_i^1(\overline{y}_i^1)$, где $p_i{}^1$ — новые предикаты, которые будем называть предикатами первого уровня, а переменные y_i^1 — новые переменные для списков исходных переменных, которые будем называть переменными первого уровня.

Обозначим формулы, полученные из $A_k(\overline{x}_k)$ путем замены всех вхождений формул вида $P_i^1(\overline{y}_i^1)$ на атомарные формулы $p_i^1(x_i^1)$ (при $y_i^1 \subseteq x$) посредством $A_k^1(\overline{x}_k^1)$. Здесь $\overline{x}_k^1$ — список всех переменных формулы $A_k^1(\overline{x}^1)$, состоящий как из некоторых (быть может всех) исходных переменных формулы $A_k(\overline{x}_k)$, так и из переменных первого уровня, появившихся в формуле $A_k^1(\overline{x}_k^1)$. Такие формулы $A_k^1(\overline{x}_k^1)$ можно рассматривать как описания классов в терминах предикатов исходного (нулевого) и первого уровней.

Описанием объекта $S^1(\omega)$ первого уровня назовем множество всех атомарных формул вида $p_i^1(\omega_{ij}^1)$, для которых истинна определяющая подформула $P_i^1(\overline{\tau}_{ij}^1)$ при $\tau_{ij}^1 \subset \omega$, а объект первого уровня ω_{ij}^1 представляет из себя список исходных объектов $\overline{\tau}_{ij}^1$.

Процедуру выделения «часто» встречающихся подформул «небольшой сложности» можно повторить с формулами $A_k^1(\overline{x}^1)$.

В результате построения составных предикатов (т. е. предикатов различных уровней) и многоуровневого описания классов исходное множество описаний классов $\{A_k(\overline{x})\}$ может быть записано с помощью равносильной ей многоуровневой системы описаний классов вида

$$\left\{\begin{array}{rcl} & A_k^L(\overline{x}^L), & \\ p_1^1(x_1^1) & \Leftrightarrow & P_1^1(\overline{y}_1^1), \\ & \vdots & \\ p_{n_1}^1(x_{n_1}^1) & \Leftrightarrow & P_{n_1}^1(\overline{y}_{n_1}^1), \\ & \vdots & \\ p_i^l(x_i^l) & \Leftrightarrow & P_i^l(\overline{y}_i^l), \\ & \vdots & \\ p_{n_L}^L(x_{n_L}^L) & \Leftrightarrow & P_{n_L}^L(\overline{y}_{n_L}^L). \end{array}\right.$$

В [8] доказаны оценки изменения числа шагов проверки (4) при использовании двухуровневого описания классов и приведены модельные примеры, иллюстрирующие существенное уменьшение показателя экспоненты при его использовании. Однако в этой работе применяется эвристическое выделение общих подформул для построения двухуровневого описания классов. В [9] предложен алгоритм построения многоуровневого описания класса по обучающей выборке, в основе которого лежит понятие неполной выводимости.

4. Понятие неполной выводимости формулы. Для распознавания объектов с неполной информацией в [12] было введено понятие неполной выводимости предикатной формулы.

Рассматривается задача проверки того, что из истинности всех формул множества $S(\omega)$ следует истинность $A(\overline{x})$ или некоторой ее максимальной подформулы $\widetilde{A}(\overline{y})$ на наборе различных констант из ω, где список переменных $\overline{y}$ является подсписком списка переменных $\overline{x}$.

Пусть a и $\widetilde{a}$ — количества атомарных формул в элементарных конъюнкциях $A(\overline{x})$ и $\widetilde{A}(\overline{y})$ соответственно, m и $\widetilde{m}$ — количества предметных переменных в $A(\overline{x})$ и $\widetilde{A}(\overline{y})$ соответственно.

Числа q и r вычисляются по формулам $q = \frac{\widetilde{a}}{a}$, $r = \frac{\widetilde{m}}{m}$ и характеризуют степень совпадения формул $A(\overline{x})$ и $\widetilde{A}(\overline{y})$. В этом случае подформула $\widetilde{A}(\overline{y})$ называется (q,r)-фрагментом формулы $A(\overline{x})$. Она является максимальной подформулой элементарной конъюнкции $A(\overline{x})$, если является ее (q,r)-фрагментом с максимальным среди всех (q,r)-фрагментов значением параметра q. То есть для $\widetilde{A}(\overline{y})$ справедливо $S(\omega) \Rightarrow \exists \overline{y}_{\neq} \widetilde{A}(\overline{y})$, и ни для какой подформулы формулы $A(\overline{x})$ с большим значением параметра q это следствие не выполняется.

Параметр q характеризует, насколько информативен фрагмент, содержащий лишь r-ю часть переменных.

Задача нахождения максимального (q,r)-фрагмента формулы $\widetilde{A}(\overline{y})$ при условии справедливости множества постоянных атомарных формул $S(\omega)$ называется задачей проверки неполной выводимости этой формулы. В [12] приведен один из возможных алгоритмов ее решения.

5. Нахождение наибольшей общей подформулы двух формул. Понятие неполной выводимости из множества постоянных атомарных формул или их отрицаний легко обобщается до понятия неполной выводимости двух элементарных конъюнкций.

Пусть $A(\overline{x})$ и $B(\overline{y})$ — две элементарные конъюнкции предикатных формул со списками предметных переменных $\overline{x}$ и $\overline{y}$ соответственно. Проверка неполной выво-

димости $A(\overline{x}) \Rightarrow_P \exists\overline{y}_{\neq}B(\overline{y})$ заключается в получении максимального (q,r)-фрагмента $Q_{AB}(\overline{z})$ формулы $B(\overline{y})$ и такой подстановки $\lambda_{AQ} = |\frac{\overline{z}}{\overline{y}'}$ списка переменных $\overline{y}'$ из $\overline{y}$ вместо переменных списка $\overline{z}$, что $Q_{AB}(\overline{y'})$ является максимальной подформулой формулы $B(\overline{y})$, при этом $A(\overline{x}) \Rightarrow \exists\overline{y'}_{\neq}Q_{AB}(\overline{y'})$.

Аналогично при проверке неполной выводимости $B(\overline{y}) \Rightarrow_P \exists\overline{x}_{\neq}A(\overline{x})$ получаем максимальный (q,r)-фрагмент $Q_{BA}(\overline{z})$ формулы $A(\overline{x})$ и подстановку $\lambda_{BQ} = |\frac{\overline{z}}{\overline{x}'}$ списка переменных $\overline{x}'$ из $\overline{x}$ вместо переменных списка $\overline{z}$, что $Q_{BA}(\overline{y'})$ есть максимальная подформула формулы $A(\overline{x})$, такая что $B(\overline{y}) \Rightarrow \exists\overline{x'}_{\neq}Q_{BA}(\overline{x'})$.

Можно доказать, что формулы $Q_{AB}(\overline{y'})$ и $Q_{BA}(\overline{x'})$ изоморфны. В качестве максимальной (с точностью до имен переменных) общей подформулы двух элементарных конъюнкций $A(\overline{x})$ и $B(\overline{y})$ можно взять любую из них. Обозначим эту подформулу посредством $Q(\overline{z})$.

Найденные в процессе проверки неполной выводимости подстановки λ_{AQ} и λ_{BQ} являются унификаторами формул A и B соответственно с формулой Q.

6. Описание алгоритма проверки неполной выводимости двух формул. Вначале введем некоторые обозначения.

Пусть проверяется неполная выводимость $A(\overline{x}) \Rightarrow_P \exists\overline{y}_{\neq}B(\overline{y})$. Для краткости эти формулы будем обозначать A и B соответственно. Литералами будем называть атомарные формулы или их отрицания. Пусть n — число литералов в формуле A; k — число литералов в формуле B; $C(\overline{z})$ — максимальная по числу литералов найденная подформула на текущем этапе работы алгоритма для краткости обозначим ее через C; r — число литералов в C; S — текущий частичный унификатор для формул A и B, т. е. $S = |^{y'}_{x'}$, $x' \subset x$, $y' \subset y$; P — множество литералов, являющихся литералами формулы A, которые не выкинуты из рассмотрения на текущем шаге работы алгоритма; L — множество литералов, являющихся литералами формулы B, каждый из которых графически совпадает с каким-либо литералом формулы A при текущем частичном унификаторе; L' — множество литералов, являющихся литералами формулы B, которые не выкинуты из рассмотрения на текущем шаге работы алгоритма и могут быть включены в L.

Алгоритм выделения максимальной с точностью до имен переменных общей подформулы.

1. Формула C не инициализирована, подстановка S пуста, множество P совпадает с множеством литералов A, множество L' совпадает с множеством литералов B, множество L пусто.

2. При текущей подстановке S проверяем, какие литералы из L' заведомо не могут графически совпасть ни с одним литералом A (например, $p_i(y_1, y_2, y_3)\ |S = p_i(x_1, y_2, y_3)$, но в A нет ни одного литерала с предикатом p_i без отрицания и первым аргументом которого является переменная x_1). Такие литералы отбрасываются из множества L'.

Также проверяем, какие литералы графически совпали с одним из литералов формулы A. Их перемещаем из множества L' в L.

3. При текущей подстановке S проверяем, какие литералы из P не могут быть выполнены (например, $p_i(x_1, x_2) \in P$, $S = |^{y_3}_{x_2}$, но в L' нет литералов с предикатом p_i без отрицания, где вторым аргументом не было еще ничего подставлено или стояла x_2, т. е. заведомо никакой литерал из L' не может графически совпасть с данным) и удаляем такие литералы из P.

4. Если количество литералов в L больше количества литералов в C, то конъюнкцию литералов из L записываем в C.

5. Если подстановка S полная, т. е. $S = |\overline{\frac{y}{x}}$, то переходим к п. 8 (предыдущий шаг рекурсии).

6. Если $|L| + |L'| < r$, то переходим к п. 8 (предыдущий шаг рекурсии).

7. Каждый литерал из L' пытаемся последовательно унифицировать с каждым литералом из P. Если унификация возможна, то делаем рекурсионный шаг — переходим к п. 2, при этом удаляя выбранный литерал из P, добавляя к S переменные, определяемые данной подстановкой, также перемещаем литерал из L' в L, если он графически совпал с одним из литералов из P.

8. Проверка аналогична п. 6. Если все возможные подстановки перебраны, то переходим к п. 8 (предыдущий шаг рекурсии; если его не было, алгоритм заканчивает работу). В противном случае продолжаем перебор (переходим к п. 7).

7. Оценки числа шагов работы алгоритма выделения максимальной с точностью до имен переменных общей подформулы. Алгоритм заканчивает работу за конечное число шагов, так как число литералов в формулах A и B конечно и каждый шаг рекурсии удаляет хотя бы один элемент как из множества L', так и из множества P.

На асимптотику работы алгоритма самое главное влияние оказывает структура входных данных. В худшем случае каждый шаг рекурсии удаляет только по фиксированному количеству a элементов из множеств L' и P. Это возможно, например, в случае, когда каждая переменная встречается ровно в двух литералах (если переменная встречается только в одном литерале, то она не зависит от подстановок остальных переменных, и для нее сразу можно определить единственно верную подстановку, не учитывая ее вхождения при переборе), все предикаты в формулах двухместные и все литералы равны с точностью до имен переменных. В таком случае число шагов работы алгоритма имеет порядок

$$O(\underbrace{(n \cdot k) \cdot ((n-a) \cdot (k-a)) \cdot \ldots \cdot ((n-sa) \cdot (k-sa))}_{s\,=\,\min\{\lfloor \frac{n}{a} \rfloor,\ \lfloor \frac{k}{a} \rfloor\}}) = O(n^s \cdot k^s).$$

В лучшем случае глубина рекурсии равна единице. Это возможно, например, когда $|\overline{x}| = |\overline{y}| = t$ и все предикаты t-местные. Тогда выбор единственной пары графически совпадающих литералов определяет подстановку всех переменных. В данном случае число шагов работы алгоритма имеет порядок

$$O(n \cdot k \cdot (n \cdot k)) = O(n^2 \cdot k^2).$$

В среднем после каждого шага рекурсии количество литералов для следующего рекурсивного шага будет уменьшаться в α раз при некотором $\alpha > 1$. В таком случае число шагов работы алгоритма (при условии, что $d = \log_\alpha(n \cdot k)$) имеет порядок

$$O\left(\underbrace{n \cdot k \cdot \frac{n \cdot k}{\alpha} \cdot \frac{n \cdot k}{\alpha^2} \cdot \ldots \cdot 1}_{d\,=\,\log_\alpha(n \cdot k)}\right) = O\left(\frac{(n \cdot k)^d}{\alpha^{\frac{d \cdot (d+1)}{2}}}\right) =$$

$$= O\left(\frac{(n \cdot k)^{\log_\alpha(n \cdot k)}}{(n \cdot k)^{1/2\ \log_\alpha(n \cdot k)}}\right) = O\left((n \cdot k)^{1/2\ \log_\alpha(n \cdot k)}\right).$$

8. Анализ зависимости работы алгоритма от структуры входных данных. Описанный алгоритм был реализован на языке программирования C++. Все

данные о времени работы алгоритмов относятся к запуску без распараллеливания на компьютере с процессором Intel B950 с частотой 2.1 ГГц. Они получены в результате сбора статистики после 50 запусков алгоритма.

Для получения сравнительных результатов скорости работы алгоритма выделения наибольшей общей (с точностью до имен аргументов) подформулы двух формул было проведено несколько запусков на наборах пар формул. Распределение переменных в формулах было случайным и близко к равномерному, т. е. все переменные встречались в записи формулы примерно одинаковое число раз, во всех формулах был использован один и тот же предикатный символ (без отрицания), если не оговорено противное.

На число шагов работы данного алгоритма оказывает серьезное влияние структура входных данных. Существует зависимость числа шагов работы алгоритма от количества аргументов предикатов, входящих в исходные формулы.

Параметры формул: 25 литералов, 12 переменных, 1 предикатный символ.

Количество аргументов у предиката...	4	5
Время, с		
минимальное	0.146	0.0478
среднее	0.217	0.0518
максимальное	0.269	0.0655

Результаты показывают значительное снижение числа шагов работы алгоритма при увеличении количества аргументов предиката. Это является следствием уменьшения глубины рекурсии в процессе перебора множеств литералов во время работы алгоритма. Также существенное влияние оказывает количество различных предикатных символов, участвующих в записи формулы.

Параметры формул: 25 литералов, 12 переменных, количество аргументов у предикатов 3.

Количество различных предикатных символов ...	2	3
Время, с		
минимальное	0.452	0.083
среднее	0.723	0.164
максимальное	1.131	0.352

При больших исходных данных особенно заметна эффективность предложенного алгоритма, отсекающего «ветви», заведомо не приводящие к успеху (пп. 2 и 3 алгоритма) по сравнению с алгоритмом полного перебора всех возможных вариантов. Так, например, для алгоритма полного перебора соответствующие величины были следующими:

Количество различных предикатных символов ...	3
Время, с	
минимальное	7.914
среднее	12.078
максимальное	18.321

Выбор предикатных символов во время запуска случаен, все предикатные символы встречались в записи формулы примерно равное количество раз. Результаты показывают существенное снижение числа шагов работы алгоритма при увеличении количества различных предикатных символов в формулах. Данный факт является следствием увеличения эффективности отброса «ветвей» во время процесса перебора множеств литералов, так как наличие нескольких предикатных символов, в определенном смысле, разбивает процесс поиска на подзадачи меньшей размерности. Также

немаловажный фактор для изменения времени работы алгоритма — это распределение переменных в записи формулы. Все вышеприведенные результаты получены с учетом равномерного распределения переменных, т. е. с равным в среднем числом вхождений переменных в формулу (табл. 1).

Таблица 1. **Зависимость времени от распределения переменных**

Распределение переменных	Время, с		
	минимальное	среднее	максимальное
Равномерное	0.156	0.213	0.261
Нормальное с параметрами (6, 2)	0.047	0.056	0.065

Параметры формул: 25 литералов, 12 переменных, 1 предикатный символ, количество аргументов у предикатов 3.

Нормальное распределение переменных следует воспринимать с учетом нумерации переменных вида x_i, в данном случае i меняется от 0 до 11, т. е. наибольшее число вхождений в среднем у переменной x_6 меньше, чем у x_5, x_7 и т. д. Результаты показывают значительное снижение числа шагов работы алгоритма при неравномерном распределении переменных. Данный факт также является следствием более эффективного избавления от «ветвей» в процессе перебора множеств литералов.

Суммируя все замечания касательно структуры входных данных, можно добиться существенно более эффективного использования данного алгоритма (табл. 2).

Таблица 2. **Зависимость времени от структуры входных данных**

Тип формул	Время, с		
	минимальное	среднее	максимальное
1	7.914	12.078	18.321
2	11.04	16.85	34.53

Формулы типа 1: 25 литералов, 12 переменных, 1 предикатный символ, количество аргументов у предикатов 3; равномерное распределение переменных.

Формулы типа 2: 100 литералов, 40 переменных, 6 предикатных символов, количество аргументов у предикатов 5; нормальное распределение с параметрами (20, 6).

Таким образом, несмотря на увеличение длины записи входных данных в 4 раза (алгоритм имеет экспоненциальные оценки числа шагов работы от длины записи входных данных), за счет использования информации о зависимости алгоритма от структуры входных данных удалось сохранить время работы на прежнем уровне.

9. Алгоритм проверки изоморфности элементарных конъюнкций атомарных предикатных формул. При построении многоуровневого описания классов большое значение имеет проверка изоморфности двух элементарных конъюнкций. Эта задача близка к так называемой «открытой» задаче проверки изоморфизма графов, для которой неизвестен полиномиальный алгоритм и не доказано, что она NP-полна.

Предложим полиномиальный алгоритм, который по результатам численных экспериментов дает правильный ответ более чем в 99.5% случаев. Полученную оценку числа шагов этого алгоритма можно назвать генерической сложностью задачи [13].

Определение 2. *Характеристикой переменной v в формуле A для литерала L_i, $\chi(v)^A_{L_i}$, называется упорядоченный набор $(e_1, e_2, ..., e_q)$, где q — количество переменных литерала L_i, e_j — количество атомарных формул с тем же предикатом, что и в литерале L_i, в которых переменная v является j-м аргументом.*

Определение 3. *Полной характеристикой переменной v в формуле A, $\chi(v)^A$, называется упорядоченный набор $(\chi(v)^A_{L_1}, ..., \chi(v)^A_{L_i}, ..., \chi(v)^A_{L_n})$, где L_i пробегает по всем литералам формулы A.*

З а м е ч а н и е. Стоит заметить, что равенство полных характеристик переменных $\chi_1(v)^A = (\chi_1(v)^A_{L_1}, ..., \chi_1(v)^A_{L_i}, ..., \chi_1(v)^A_{L_n})$ и $\chi_2(u)^A = (\chi_2(u)^A_{L'_1}, ..., \chi_2(u)^A_{L'_i}, ..., \chi_2(u)^A_{L''_n})$ влечет за собой равенство множеств литеральных символов $\{L_1, ..., L_n\}$ и $\{L'_1, ..., L'_{n'}\}$.

Определение 4. *Множеством характеристик формулы A, $\chi(A)$, называется мультимножество $\{\ \chi(v_k)^A\ \}$, где v_k пробегает множество всех переменных формулы A.*

Обозначения:

• исходные формулы $A(\overline{x})$ и $B(\overline{y})$ соответственно;

• k_A и k_B — количество различных предметных переменных в формулах соответственно;

• n_A и n_B — количество литералов в формулах соответственно;

• l_A и l_B — количество разных предикатных символов в формулах соответственно.

Алгоритм проверки равенства характеристик.

1. Если $k_A \neq k_B$, или $n_A \neq n_B$, или $l_A \neq l_B$, то алгоритм возвращает отрицательный ответ, иначе п. 2.

2. Если $\chi(A) \neq \chi(B)$, то алгоритм возвращает отрицательный ответ, в противном случае алгоритм возвращает положительный ответ.

Важно отметить, что алгоритм не является абсолютно точным. В большинстве случаев положительный результат работы алгоритма означает равенство формул. В п. 10 будут приведены статистические оценки этого факта. В случае возврата отрицательного ответа результат работы алгоритма абсолютно верен.

Теорема. *Если алгоритм проверки равенства предикатных формул возвращает отрицательный ответ, то формулы A и B не равны.*

Д о к а з а т е л ь с т в о. Очевидно, что формулы не равны в случае возврата отрицательного ответа при выполнении п. 1 алгоритма. Рассмотрим возврат из п. 2.

Предположим противное — формулы A и B равны. С одной стороны, неравенство множеств $\chi(A)$ и $\chi(B)$ означает, что в множестве $\chi(A)$ есть по крайней мере одна полная характеристика переменной (не уменьшая общности, обозначим ее через $\chi(x)^A$), которой нет в $\chi(B)$; с другой — равенство формул A и B означает, что существует такая подстановка переменных $S = |\frac{\overline{x}}{\overline{y}}$ и такая перестановка литералов в записи конъюнкции формулы A, что в результате формула A графически совпадет с формулой B. Следовательно, с помощью описанных операций была изменена полная характеристика $\chi(x)^A$. Однако данные операции не могут изменить число вхождений переменной в предикаты или символы предикатов, одним из аргументов которых является переменная. ■

Однако из равенства множеств характеристик двух предикатных формул не следует равенство самих формул.

Пример.

$$A(\overline{x}) = p_1(x_3, x_0, x_2)\ \&\ p_1(x_1, x_3, x_0)\ \&\ p_1(x_2, x_1, x_3),$$

$$B(\overline{y}) = p_1(y_3, y_0, y_2)\ \&\ p_1(y_1, y_2, y_0)\ \&\ p_1(y_2, y_1, y_3).$$

Нетрудно видеть, что формулы не изоморфны, но их множества характеристик

$$\chi(A) = \{\chi(x_0)^A, \chi(x_1)^A, \chi(x_2)^A, \chi(x_3)^A\} =$$

$$= \{\ (\chi(x_0)^A_{p_1}), (\chi(x_1)^A_{p_1}), (\chi(x_2)^A_{p_1}), (\chi(x_3)^A_{p_1})\ \} =$$
$$= \{\ ((0,1,1)), ((1,1,0)), ((1,0,1)), ((1,1,1))\ \},$$
$$\chi(B) = \{\chi(y_0)^B, \chi(y_1)^B, \chi(y_2)^B, \chi(y_3)^B\} =$$
$$= \{\ (\chi(y_0)^B_{p_1}), (\chi(y_1)^B_{p_1}), (\chi(y_2)^B_{p_1}), (\chi(y_3)^B_{p_1})\ \} =$$
$$= \{\ ((0,1,1)), ((1,1,0)), ((1,1,1)), ((1,0,1))\ \}$$

совпадают.

10. Оценки числа шагов алгоритма проверки изоморфности формул. Основной этап алгоритма заключается в построении множества характеристик формул и проверки их на равенство. Для построения данных множеств нужно просмотреть все литералы соответствующих формул и посчитать число вхождений переменных в них. Этот этап занимает порядка $O(n \cdot d)$ шагов, где n — число литералов в формуле, $d = \max_{i=1,\ldots,n}\{|L_i|\}$, L_i — литерал с предикатным символом p_i. Далее следует проверка равенства полученных множеств характеристик. Она требует порядка $O(k^2)$ шагов, где k — число различных переменных в формулах. Таким образом, число шагов работы всего алгоритма имеет порядок $O(n \cdot d + k^2)$.

Для получения статистических оценок равенства самих формул при равенстве множеств характеристик двух формул были проведены численные расчеты. Производились запуск данного алгоритма и проверка корректности его работы с помощью алгоритма выделения наибольшей общей подформулы с точностью до имен аргументов. Для этого использовались пары формул с числом переменных от 3 до 7, с числом литералов от 3 до 10 и с размерностью предикатов от 3 до 5 (в обеих формулах эти числа совпадали). Число переменных всегда было больше размерности предикатов. Указанные числа выбирались случайным образом, также распределение переменных в формулах было случайным и было близко к равномерному. После проверки 10^6 данных пар формул было получено, что только в 0.038% случаях имеет место быть ложноположительный результат работы алгоритма. Таким образом, точность алгоритма составляет примерно 99.95%.

11. Заключение. Исходя из оценок числа шагов работы алгоритма, можно сделать вывод, что для решения задачи выделения наибольшей общей с точностью до имен переменных подформулы двух элементарных конъюнкций может быть успешно применен данный алгоритм. Требует дальнейшего исследования зависимость числа шагов работы алгоритма от вида входных данных. По-видимому, оказывает сильное влияние распределение переменных в качестве аргументов предикатов. В соответствии с этим отметим, что возможно усовершенствование алгоритма на этапе выбора литералов из множеств P и L'. К тому же такой алгоритм не вводит порядок перебора возможных подстановок: это не оказывает сильного влияния на время его работы с более «равномерными» данными, но, скорее всего, будет играть значительную роль в случае, когда некоторые переменные будут на порядок чаще встречаться в предикатах, чем другие. Также возможен эвристический вариант алгоритма, ограничивающий глубину рекурсии или множество выбираемых литералов для следующего шага рекурсии.

Также требует дальнейшего изучения алгоритм проверки изоморфизма формул. Возможны классификация случаев ложноположительного результата работы и доработка алгоритма, исходя из полученных данных.

Алгоритм может быть применен при исследовании сложных составных объектов, описание которых строится на основе свойств их элементов и отношений между ними. В частности, он важен при решении таких задач:

— введение метрики в множестве описаний объектов на языке исчисления предикатов [15];

— построение многоуровневого описания классов объектов [8], существенно уменьшающее время распознавания сложного составного объекта и позволяющее производить параллельные вычисления;

— построение самообучающейся логико-предикатной сети, которая во время обучения может изменять свою конфигурацию [14], в отличие от классической нейронной сети, конфигурация которой задается изначально.

Литература

1. *Нильсон Н.* Искусственный интеллект. Методы поиска решений / пер. с англ. В. Л. Стефанюка; под ред. С. В. Фомина. М.: Мир, 1973. 270 с. (*Nilson Nils J.* Problem-solving methods in artificial intelligence.)

2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи / пер. с англ. Е. В. Левнера, М. А. Фрумкина; под ред. А. А. Фридмана. М.: Мир, 1982. 416 с. (*Garey M. R., Johnson D. S.* Computers and Intractability: A Guide to the Theory of NP-Completeness.)

3. *Косовская Т. М.* Некоторые задачи искусственного интеллекта, допускающие формализацию на языке исчисления предикатов, и оценки числа шагов их решения // Труды СПИИРАН. 2010. Вып. 14. С. 58–75.

4. *Косовская Т. М.* Распознавание объектов из классов, замкнутых относительно группы преобразований // Вестн. С.-Петерб. ун-та. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2009. Вып. 3. С. 45–55.

5. *Журавлев Ю. И.* Об алгоритмах распознавания с представительными наборами (о логических алгоритмах) // Журн. вычисл. математики и матем. физики. 2002. Т. 42, № 9. С. 1425–1435.

6. *Косовская Т. М.* Доказательства оценок числа шагов решения некоторых задач распознавания образов, имеющих логические описания // Вестн. С.-Петерб. ун-та. Сер. 1. Математика. Механика. Астрономия. 2007. Вып. 4. С. 82–90.

7. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход. 2-е изд. / пер. с англ. К. Птицына. М.: Издат. дом «Вильямс», 2006. 1408 с. (*Russel S. J., Norvig P.* Artificial Intelligence. A Modern Approach.)

8. *Косовская Т. М.* Многоуровневые описания классов для уменьшения числа шагов решения задач распознавания образов, описываемых формулами исчисления предикатов // Вестн. С.-Петерб. ун-та. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2008. Вып. 1. С. 64–72.

9. *Косовская Т. М.* Подход к решению задачи построения многоуровневого описания классов на языке исчисления предикатов // Труды СПИИРАН. 2014. № 3 (34). С. 204–217.

10. *Клини С.* Математическая логика / пер. с англ. Ю. А. Гастева; под ред. Г. Е. Минца. М.: Мир, 1973. 480 с. (*Kleene S. C.* Mathematical logic.)

11. *Косовская Т. М.* Частичная выводимость предикатных формул как средство распознавания объектов с неполной информацией // Вестн. С.-Петерб. ун-та. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2009. Вып. 1. С. 74–84.

12. *Рыбалов А. Н.* Об одном генерическом отношении рекурсивно перечислимых множеств // Алгебра и логика. 2016. Т. 55, № 5. С. 587–596.

13. *Kosovskaya T.* Distance between objects described by predicate formulas // Intern. Book Series. Information Science and Computing. Book 25. Mathematics of Distances and Applications / eds: M. Deza, M. Petitjean, K. Markov. Sofia, Bulgaria, ITHEA Publ., 2012. P. 153–159.

14. *Косовская Т. М.* Самообучающаяся сеть с ячейками, реализующими предикатные формулы // Труды СПИИРАН. 2015. № 6 (43). С. 94–113.

Для цитирования: *Косовская Т. М., Петров Д. А.* Выделение наибольшей общей подформулы предикатных формул для решения ряда задач искусственного интеллекта // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2017. Т. 13. Вып. 3. С. 250–263. DOI: 10.21638/11701/spbu10.2017.303

References

1. Nilson Nils J. *Problem-solving methods in artificial intelligence.* New York, McGRAW-HILL BOOK COMPANY Press, 1971, 280 p. (Russ. ed.: Nilson N. *Iskusstvennyi intellekt. Metody poisra reshenii.* Moscow, Mir Publ., 1973, 270 p.)

2. Garey M. R., Johnson D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, Freeman Press, 1979, 340 p. (Russ. ed.: Garey M. R., Johnson D. S. *Vychislitelnye mashiny i trudnoreshaemye zadachi*. Moscow, Mir Publ., 1982, 416 p.)

3. Kosovskaya T. M. Nekotorye zadachi iskusstvennogo intellekta, dopuskaiushchie formalizatsiiu na iazyke ischisleniia predikatov, i otsenki chisla shagov ikh resheniia [Some artificial intelligence problems permitting formalization by means of predicate calculus language and upper bounds of their solution steps]. *SPIIRAS Proceedings*, 2010, no. 14, pp. 58–75. (In Russian)

4. Kossovskaya T. M. Raspoznavanie ob"ektov iz klassov, zamknutykh otnositel'no gruppy preobrazovanii [Partial deduction of predicate formula as an instrument for recognition of an object with incomplete description]. *Vestnik of Saint-Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes*, 2009, iss. 3, pp. 45–55. (In Russian)

5. Zhuravlev Yu. I. Ob algoritmah raspoznavaniya s predstavitelnymi naborami (o logichskih algoritmah) [Recognition algorithms with representative sets (logical algorithms)]. *Zhurnal vychislitelnoi matamatiki i matematicheskoi fiziki* [*Computational Mathematics and Mathematical Physics*], 2002, vol. 2, no. 9, pp. 1425–1435. (In Russian)

6. Kossovskaya T. M. Dokazatel'stva otsenok chisla shagov resheniia nekotorykh zadach raspoznavaniia obrazov, imeiushchikh logicheskie opisaniia [Proofs of the number of steps bounds for solving of some pattern recognition problems with logical description]. *Vestnik of Saint Petersburg University. Series 1. Mathematics. Mechanics. Astronomy*, 2007, no. 4, pp. 82–90. (In Russian)

7. Russel S. J., Norvig P. *Artificial Intelligence. A Modern Approach.* Pearson Education, Inc., 2003, 1412 p. (Russ. ed.: Russel S. J., Norvig P. *Iskusstvennyi intellekt: sovremennyi podhod*. Moscow, "Viliams" Publ., 2006, 1408 p.)

8. Kossovskaya T. M. Mnogourovnevye opisaniia klassov dlia umen'sheniia chisla shagov resheniia zadach raspoznavaniia obrazov, opisyvaemykh formulami ischisleniia predikatov [Level descriptions of classes for decreasing step number of pattern recognition problem solving described by predicate calculus formulas]. *Vestnik of Saint Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes*, 2008, iss. 1, pp. 64–72. (In Russian)

9. Kosovskaya T. M. Podkhod k resheniiu zadachi postroeniia mnogourovnevogo opisaniia klassov na iazyke ischisleniia predikatov [An approach to the construction of a level description of classes by means of a predicate calculus language]. *SPIIRAS Proceedings*, 2014, no. 3(34), pp. 204–217. (In Russian)

10. Kleene S. C. *Mathematical logic.* New York, Wiley, Dover Publ., 1967. 398 p. (Russ. ed.: Kleene S. *Matematicheskaia logika*. Moscow, Mir Publ., 1973, 480 p.)

11. Kossovskaya T. M. Chastichnaia vyvodimost' predikatnykh formul kak sredstvo raspoznavaniia ob"ektov s nepolnoi informatsiei [Partial deduction of predicate formula as an instrument for recognition of an object with incomplete description]. *Vestnik of Saint Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes*, 2009, iss. 1, pp. 74–84. (In Russian)

12. Rybalov A. N. Ob odnom generichеskom otnoshenii rekursivno perechislimykh mnozhestv [A Generic relation on Recursively Enumerable Sets]. *Algebra and Logic*, 2016, vol. 55, no. 5, pp. 587–596. (In Russian)

13. Kosovskaya T. Distance between objects described by predicate formulas. *International Book Series. Information Science and Computing. Book 25. Mathematics of Distances and Applications*. Sofia, Bulgaria, ITHEA Publ., 2012, pp. 153–159.

14. Kosovskaya T. M. Samoobuchaiushchaiasia set' s iacheikami, realizuiushchimi predikatnye formuly [Self-training network with the nells implementing predicate formulas]. *SPIIRAS Proceedings*, 2015, no. 6(43), pp. 94–113. (In Russian)

For citation: Kosovskaya T. M., Petrov D. A. 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*, 2017, vol. 13, iss. 3, pp. 250–263. DOI: 10.21638/11701/spbu10.2017.303

Статья рекомендована к печати проф. Л. А. Петросяном.

Статья поступила в редакцию 30 сентября 2016 г.

Статья принята к печати 8 июня 2017 г.