Компьютерные инструменты в образовании, 2015
№ 4: 17–25
УДК: 004.93.51
http://ipo.spb.ru/journal

ВЫДЕЛЕНИЕ МАКСИМАЛЬНОЙ ОБЩЕЙ ПРЕДИКАТНОЙ ПОДФОРМУЛЫ С ПОМОЩЬЮ ОБРАТНОГО МЕТОДА МАСЛОВА*

Петухова Нина Дмитриевна

Аннотация

Статья посвящена изложению алгоритма выделения максимальной общей с точностью до имён переменных подформулы двух элементарных конъюнкций атомарных предикатных формул. Предлагаемый алгоритм использует введённую ранее автором модификацию обратного метода Маслова, а также Муравьиные тактики и параллельные вычисления. Проблема выделения максимальной общей с точностью до имён переменных подформулы предикатных формул имеет достаточно широкое применение при построении эффективных алгоритмов решения задач искусственного интеллекта, допускающих формализацию средствами исчисления предикатов. Приведены асимптотические оценки числа шагов работы описанного алгоритма.

Ключевые слова: *искусственный интеллект, логико-предметное распознавание образов, исчисление предикатов, сложность алгоритмов, обратный метод Маслова, параллельные вычисления, неполная выводимость.*

1. ВВЕДЕНИЕ

Во многих задачах искусственного интеллекта исследуемый объект характеризуется не только своими глобальными признаками, задающими его свойства как единого целого, но и свойствами его частей и отношений между ними. При распознавании контурного изображения объект может быть представлен как множество его вершин и вида соединений этих вершин. В задачах диагностики объект задаётся набором своих составляющих, а также зависимостями между этими элементами, позволяющими объекту функционировать.

Такой подход ещё в 70-е годы XX века был сформулирован в литературе (см. например [1, 2]). Однако практического применения он почти не получил в силу высокой вычислительной сложности возникающих при этом задач, которые являются NP-трудными [3].

Для уменьшения числа шагов решения таких задач были предложены различные стратегии доказательства логического следования формул (например применение обратного метода Маслова [4] и использование муравьиных алгоритмов [5]) и специальные виды описаний классов (многоуровневые описания классов [6]).

При построении многоуровневых описаний классов используется понятие максимальной общей с точностью до имён переменных подформулы двух элементарных

* Работа выполнена при поддержке гранта РФФИ 14-08-01276-а.

конъюнкций [7]. Выделение такой подформулы также является NP-трудной задачей и имеет обширное практическое применение. Например, обработка изображений и сигналов [8, 9], качественный анализ в химии.

В работе обратный метод Маслова применяется к процедуре выделения максимальной общей подформулы.

2. ПОСТАНОВКА ЗАДАЧИ

В настоящей работе исследуемый объект ω рассматривается как множество его элементов $\omega = \{a_1, ..., a_k\}$, на котором заданы исходные признаки-предикаты $P_1, ..., P_n$, характеризующие свойства элементов из ω и отношения между ними. Описанием $S(\omega)$ объекта ω является набор постоянных атомарных формул или их отрицаний, истинных на нём. Если множество всех исследуемых объектов разбито на K классов $\Omega = \cup_{k=1}^{K} \Omega_k$, то описанием класса может служить дизъюнкция элементарных конъюнкций атомарных формул $A_k(\bar{x})$, истинная только на объектах k-го класса.

При такой формализации средствами языка исчисления предикатов задача определения того, принадлежит ли объект ω или его часть k-му классу, сводится к доказательству логического следования

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

где ограниченный квантор существования $\exists \bar{x}_{\neq}$ означает, что требуется проверить существование только попарно различных значений для переменных из списка $\bar{x}$ [3].

Для доказательства приведённого логического следования достаточно уметь доказывать аналогичное следование для случая, когда $A_k(\bar{x})$ — элементарная конъюнкция атомарных формул. Доказательство (1) равносильно доказательству формулы

$$(\&S(\omega)) \Rightarrow \exists \bar{x}_{\neq} A_k(\bar{x}),$$

при любых наборах значений констант из ω, которую, используя равносильные преобразования, можно свести к формуле

$$\forall a_1, ..., a_k \exists (x_1, ..., x_n)_{\neq} (\&_{i=1}^{\delta} D_i(a_1, ..., a_k, x_1, ..., x_n)), \tag{2}$$

где $D_i(a_1, ..., a_k, x_1, ..., x_n)$ имеет вид $\vee \neg S(\omega) \vee P_{k_i}(x_1, ..., x_{n_i})$, $\vee \neg S(\omega)$ означает дизъюнкцию отрицаний всех формул из $S(\omega)$, δ — количество атомарных формул в элементарной дизъюнкции. Доказательство (2) и будет рассматриваться в этой статье.

При описании задач поиска логического вывода в исчислении предикатов одним из наиболее продуманных с точки зрения сокращения перебора методов является обратный метод, предложенный С.Ю. Масловым.

3. АЛГОРИТМ IAPTA (Inverse Ant Parallel Tactic Algorithm) РЕШЕНИЯ ЗАДАЧ ЛОГИКО-ПРЕДМЕТНОГО РАСПОЗНАВАНИЯ ОБРАЗОВ

В [5] сформулирован алгоритм IAPTA решения задач логико-предметного распознавания образов, основанный на тактиках муравьиных алгоритмов, параллельного вычисления и обратного метода Маслова. А также получены асимптотические оценки работы этого алгоритма.

Для дальнейшего чтения работы приведём основные определения.

Определение 1. *Список Γ неповторяющихся формул вида $D_i(a_1, ..., a_k, x_1, ..., x_n)$ называется Γ-набором для формул типа* (2).

Определение 2. *F-набор называется пустым, если все формулы, входящие в него, не имеют переменных и тавтологичны.*

Определение 3. *F-набор называется тупиковым, если в него входит хотя бы одна формула, не имеющая переменных и являющаяся ложной или не являющаяся ни тавтологией, ни противоречием.*

3.1. Алгоритм IAPTA

1. Строим δ-членный F-набор, формулы в котором не повторяются. То есть переписываем без конъюнкций все дизъюнкты вида $\vee\neg S(\omega) \vee P_{k_i}(x_1,...,x_{n_i})$ при $i = 1,...,\delta$. Создаем популяцию из δ процессоров.
   Каждой паре потенциально контрарных формул $P_{k_i}(x_1,...,x_{n_i})$, и $\neg P_{k_i}(a_{j_1},...,a_{j_{n_i}})$, входящих в один F-набор, назначаем приоритет их отождествления равным 1. Остальные приоритеты назначаем равными 0.
2. Копируем δ-членный F-набор $\delta - 1$ раз. Таким образом, получаем ровно δ одинаковых F-наборов. Назначаем i-му процессору ($i = 1,...,\delta$) свою начальную формулу $P_{k_i}(x_1,...,x_{n_i})$, с которой данный процессор начинает свой итерационный цикл, и потенциально контрарную ей постоянную формулу из $S(\omega)$, имеющую приоритет, равный 1. Если какие-то два процессора начинают работу с формулой, начинающейся с одного и того же предикатного символа (таких формул не более δ), то назначаем для них разные формулы из $S(\omega)$, потенциально контрарные данной. Если для каких-то двух процессоров не существует разных потенциально контрарных формул, то формула не выводима[1]. Алгоритм заканчивает работу. Иначе переходим к п. 3.3.
3. Параллельно работают δ процессоров; i-ый процессор ($i = 1,...,\delta$) осуществляет присвоение значений переменным.
   - 3.1 Если в рабочей формуле данного процессора нет переменных, то в качестве рабочей для этого процессора выбираем формулу из следующей элементарной дизъюнкции, содержащую хоть одну переменную.
   - 3.2 Ищем среди формул в $S(\omega)$ формулу $\neg P_{k_i}(a_{j_1},...,a_{j_{n_i}})$, имеющую приоритет, равный 1, и потенциально контрарную формуле $P_{k_i}(t_1,...,t_{n_i})$, с которой работает этот процессор[2]. Если нашли подходящую формулу, то переходим к п. 3.3. Если ее нет, то переходим к п. 4.
   - 3.3 Решаем систему уравнений вида $t_l = a_{j_l} (l = 1,...,n_i)$, унифицирующую список переменных и констант со списком констант. В случае если эта система имеет решение, то переходим к п. 3.4. Если система решений не имеет, то понизим приоритет этого действия до 0 и переходим к п. 3.2.
   - 3.4 Записываем результаты, полученные разными процессорами, и проверяем их на непротиворечивость.
   - 3.5 Заменяем в F-наборе каждого процессора вхождения переменных из списка на их значения, полученные в п. 3.3 и 3.4, если успешно пройдена проверка на непротиворечивость.

[1] Невозможно подобрать такой набор значений переменных, чтобы вывести пустой F-набор, так как в нем будет как минимум одна формула, в которой останутся переменные, которым будет невозможно присвоить значения.

[2] Первоначально $(t_1,...,t_{n_i})$ — список переменных $(x_1,...,x_{n_i})$. При последующих итерациях это список переменных и констант, подставленных вместо переменных после процедуры унификации.

3.6 Если для какого-либо процессора получился пустой F-набор, то алгоритм заканчивает работу. Формула выводима, и найден набор значений переменных, существование которых утверждалось в формуле.
3.7 Если получился тупиковый F-набор, то переходим к п. 4.
3.8 Если для всех процессоров приоритеты всех действий равны 0, то формула не выводима. Алгоритм заканчивает работу.
3.9 Если в F-наборе какого-либо процессора существуют формулы, имеющие переменные, которым еще не присвоены значения, то переходим к п. 3.1.

4. Возвратная часть алгоритма.
4.1 Отменяем последнее действие п. 3.5, если это возможно, и переходим к п. 3.2.
4.2 Если для какого-либо процессора отмена последнего действия п. 3.5 невозможна, то алгоритм заканчивает работу.
4.3 Если все процессоры закончили работу, но пустой F-набор не получен, то алгоритм заканчивает работу. Формула не выводима.

3.2. Оценка числа шагов работы алгоритма IAPTA

Определение 4. *Одним шагом работы алгоритма называется любое из следующих трех действий: присвоение переменной значения (решение одного уравнения вида x=a); проверка на графическое совпадение атомарных формул; подстановка значений переменных в атомарные формулы, имеющие вхождения данных переменных.*

Пусть δ — количество дизъюнктивных членов в исходной формуле; l — наибольшее количество аргументов в атомарной формуле; $s+1$ — количество атомарных формул в каждом из дизъюнктов (так как каждый дизъюнкт имеет вид $\vee\neg S(\omega) \vee P_{k_i}(x_1,...,x_{n_i})$, то во всех дизъюнктах одинаковое число атомарных формул); s_k — количество атомарных формул в $S(\omega)$ с предикатным символом P_k.

В [5] получены следующие оценки работы алгоритма IAPTA.

Теорема 1. (Нижняя оценка числа шагов работы алгоритма.) *Количество шагов доказательства логического следования вида*

$$S(\omega) \Rightarrow \exists \bar{x}_{\neq} A_k(\bar{x})$$

и нахождения значений переменных, для которых это следование имеет место, при использовании алгоритма IAPTA, не менее $2\delta(s+2)+2$.

Теорема 2. (Верхняя оценка числа шагов работы алгоритма.) *Количество шагов доказательства логического следования вида*

$$S(\omega) \Rightarrow \exists \bar{x}_{\neq} A_k(\bar{x})$$

и нахождения значений переменных, для которых это следование имеет место, при использовании алгоритма IAPTA, составляет $O(l(max_k s^k)^\delta)$.

3.3. Понятие неполной выводимости

Понятие неполной выводимости предикатной формулы было введено в [10] для распознавания объектов с неполной информацией. При этом рассматривается задача проверки того, что из истинности всех формул множества $S(\omega)$ следует истинность $A(\bar{x})$ или некоторой её максимальной подформулы $\tilde{A}(\bar{y})$ на наборе различных констант из ω, где список переменных является подсписком списка переменных $\bar{x}$.

Пусть a и $\tilde{a}$ — количество атомарных формул в $A(\bar{x})$ и в $\tilde{A}(\bar{y})$ соответственно, m и $\tilde{m}$ — количество предметных переменных в $A(\bar{x})$ и в $\tilde{A}(\bar{y})$ соответственно.

Числа q и r вычисляются по формулам $q = \tilde{a}/a$, $r = \tilde{m}/m$ и характеризуют степень совпадения формул $A(\bar{x})$ и в $\tilde{A}(\bar{y})$. При этом $0 < q \leqslant 1,\ 0 < r \leqslant 1$. Кроме того, $q = r = 1$ тогда и только тогда, когда $\tilde{A}(\bar{y})$ совпадает с $A(\bar{x})$.

Процедура выделения максимальной общей подформулы двух заданных элементарных конъюнкций $A(\bar{x})$ и в $\tilde{A}(\bar{y})$ с точностью до имён переменных сведена в [10] к проверке логического следования:

$$A(\bar{x}) \Rightarrow \exists \bar{y} \tilde{A}(\bar{y})$$

с нахождением такой подформулы $A'(\bar{y'})$ формулы $A(\bar{y})$, что имеет место следствие:

$$A(\bar{x}) \Rightarrow \exists \bar{y'} \tilde{A}(\bar{y'}), \tag{3}$$

а также общего унификатора λ формул $A(\bar{x})$ и $\tilde{A}'(\bar{y'})$, то есть такой подстановки имён переменных $\bar{y'}$ вместо некоторых имён переменных из списка $\bar{x}$, что $\tilde{A}'(\bar{y'})$ является подформулой $A(\bar{x})$ после применения унификатора λ.

3.4. Применение алгоритма IAPTA к решению задачи выделения максимальной общей подформулы

В [11] решение рассмотренных задач сведено к доказательству логического следования:

$$S(\omega) \Rightarrow \exists \bar{x}_{\neq} A_k(\bar{x}),$$

причем множество ω рассматривалось как множество констант.

При выделении максимальной общей подформулы двух заданных элементарных конъюнкций $A(\bar{x})$ и $\tilde{A}(\bar{y})$ требуется найти такую подформулу $\tilde{A}'(\bar{y'})$ формулы $\tilde{A}(\bar{y})$, что имеет место следствие (3). Очевидно, что в следствии (3) наборы $\bar{x}$ и $\bar{y'}$ являются наборами переменных. Тем не менее, если в алгоритме IAPTA заменить действие «присвоение значений переменным» на «отождествление переменных», этот алгоритм (после описанных ниже модификаций алгоритма) можно применить и для решения задачи выделения максимальной общей подформулы.

4. АЛГОРИТМ PHIAPTA (Partial Hatchability Inverse Ant Parallel Tactic Algorithm) ВЫДЕЛЕНИЯ МАКСИМАЛЬНОЙ ОБЩЕЙ ПОДФОРМУЛЫ

1. Строим δ-членный F-набор, формулы в котором не повторяются. То есть переписываем без конъюнкций все дизъюнкты вида $A(\bar{x}) \vee P_{k_i}(y'_1, ..., y'_{n_i})$ при $i = 1, ..., \delta$. Создаем популяцию из δ процессоров. Каждой паре потенциально контрарных формул $P_{k_i}(y'_1, ..., y'_{n_i})$ и $\neg P_{k_j}(x_1, ..., x_{n_j})$, входящих в один F-набор, назначаем приоритет их отождествления равным 1. Остальные приоритеты назначаем равными 0.
2. Копируем δ-членный F-набор $\delta - 1$ раз. Таким образом, получаем ровно δ одинаковых F-наборов. Назначаем i-му процессору ($i = 1, ..., \delta$) свою начальную формулу $P_{k_i}(y'_1, ..., y'_{n_i})$, с которой данный процессор начинает свой итерационный цикл, и потенциально контрарную ей формулу вида $\neg P_{k_j}(x_1, ..., x_{n_j})$ из $A(\bar{x})$, имеющую приоритет, равный 1. Для каждого процессора назначаем переменную $l_i = 0$ ($i = 1, ..., \delta$), означающую длину фрагмента формулы $A(\bar{x})$ в формуле $\tilde{A}'(\bar{y'})$.

Если какие-то два процессора начинают работу с формулой, начинающейся с одного и того же предикатного символа (таких формул не более δ), то назначаем для них разные формулы из $A(\bar{x})$, потенциально контрарные данной. Если для каких-то двух процессоров не существует разных потенциально контрарных формул, то формула $\tilde{A}'(\bar{y}')$ не является подформулой $A(\bar{x})$. Алгоритм заканчивает работу. Иначе переходим к п. 3.3.

3. Параллельно работают δ процессоров; i-ый процессор ($i = 1,...,\delta$) осуществляет отождествление переменных следующим образом.
   - 3.1 Если в рабочей формуле данного процессора нет переменных, которые еще не проходили процедуру отождествления, то в качестве рабочей для этого процессора выбираем формулу из следующей элементарной дизъюнкции, содержащую хоть одну «не отождествленную» переменную.
   - 3.2 Ищем среди формул в $A(\bar{x})$ формулу $\neg P_{k_i}(x_{j_1},...,x_{j_{n_j}})$, имеющую приоритет, равный 1, и потенциально контрарную формуле $P_{k_i}(t'_1,...,t'_{n_i})$ (из $\tilde{A}'(\bar{y}')$), с которой работает этот процессор. Если нашли подходящую формулу, то переходим к п. 3.3. Если ее нет, то переходим к п. 4.
   - 3.3 Решаем систему уравнений вида $t'_l = x_{j_l}$ ($l = 1,...,n_i$), унифицирующую списки переменных. В случае если эта система имеет решение[3], то увеличиваем длину текущего фрагмента l_i на единицу, запоминаем это число и текущее отождествление и переходим к п. 3.4. Если система решений не имеет, то понизим приоритет этого действия до 0 и переходим к п. 3.2.
   - 3.4 Записываем результаты, полученные разными процессорами, и проверяем их на непротиворечивость.
   - 3.5 Заменяем в F-наборе каждого процессора вхождения переменных из списка на их значения, полученные в п. 3.3 и 3.4, если успешно пройдена проверка на непротиворечивость.
   - 3.6 Если для какого-либо процессора все переменные из $\tilde{A}'(\bar{y}')$ получили новое значение, то алгоритм заканчивает работу. Формула $\tilde{A}'(\bar{y}')$ полностью содержится в формуле $A(\bar{x})$.
   - 3.7 Если в F-наборе какого-либо процессора отсутствуют формулы, имеющие переменные, которые еще проходили процедуру отождествления, то переходим к п. 4.
   - 3.8 Если для всех процессоров приоритеты всех действий равны 0, то формула $\tilde{A}'(\bar{y}')$ не является подформулой $A(\bar{x})$.
   - 3.9 Если в F-наборе какого-либо процессора существуют формулы, имеющие переменные, которые еще проходили процедуру отождествления, то переходим к п. 3.1.
4. Возвратная часть алгоритма.
   - 4.1 Отменяем последнее действие п. 3.5, если это возможно, уменьшаем соответствующую величину l_i на 1, запоминаем это число и текущее отождествление и переходим к п. 3.2.
   - 4.2 Если для какого-либо процессора отмена последнего действия п. 3.5 невозможна, то процессор заканчивает работу.
   - 4.3 Если все процессоры закончили работу, то наибольшее число из всех l_i на всех шагах является размером наибольшей подформулы $\tilde{A}'(\bar{y}')$ в $A(\bar{x})$. И соответ-

[3] В данном случае система решений не имеет, только тогда, когда происходит повторное отождествление какой-либо переменной (в правой или левой части уравнений).

ствующие этому отождествления являются наибольшей подформулой $\tilde{A}'(\bar{y}')$ в $A(\bar{x})$.

Схематически работа алгоритма PHIAPTA представлена на рис. 1.

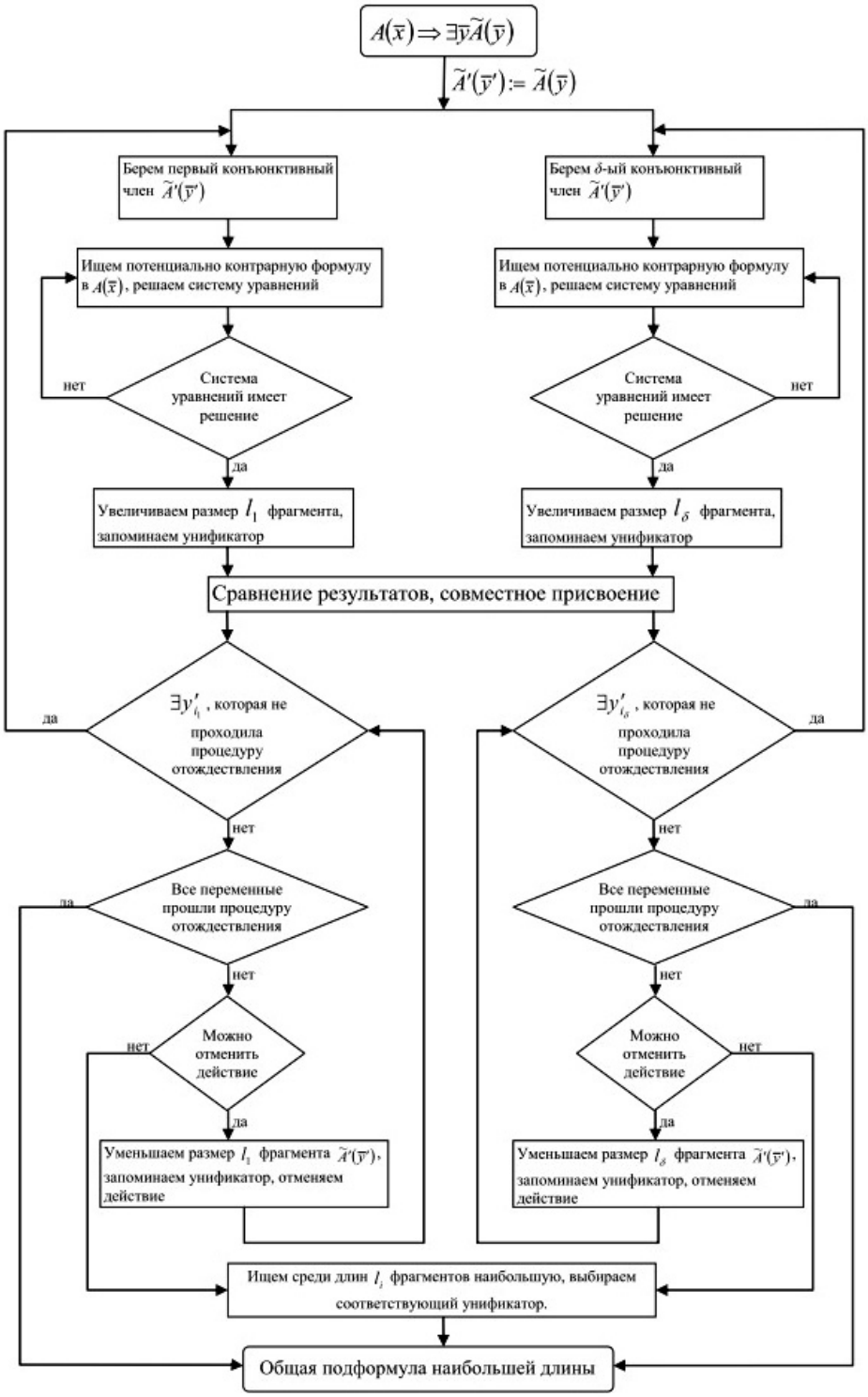

Рис. 1. Схема работы алгоритма PHIAPTA выделения наибольшей общей подформулы

5. ЗАКЛЮЧЕНИЕ

В заключение следует отметить, что, учитывая разнообразие применения частичной выводимости, приведенный выше алгоритм оказывается полезен для решения таких задач, как распознавание объекта с неполной информацией (в частности, изображения частично заслонённого объекта) [10]; определение метрики в пространстве (не фиксированной размерности), описание объектов средствами языка исчисления предикатов и вычисление расстояния между такими описаниями [11]; создание многоуровневого описания классов объектов, позволяющего учитывать общие характеристики объектов класса и существенно уменьшить число шагов решения задач распознавания [7] и построения самокорректирующихся предикатных сетей [6].

Кроме того, несмотря на то, что оценки приведенного алгоритма имеют тот же порядок, что и оценки существующих алгоритмов выделения наибольшей общей подформулы [12], следует отметить, что этот алгоритм на практике имеет большую скорость и легче реализуем программно.

Список литературы

1. *Нильсон Н.* Искусственный интеллект. Методы поиска решений. М.: Мир, 1973.
2. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2006.
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. *Петухова Н.Д., Косовская Т.М.* Решение задач логико-предметного распознавания образов с использованием тактик обратного метода Маслова // Компьютерные инструменты в образовании, 2014 № 3. С. 9–20.
5. *Петухова Н.Д., Косовская Т.М.* Применение тактик муравьиных алгоритмов для решения некоторых задач искусственного интеллекта // Вестник СПбГУ. Сер. 10. 2015. Вып. 3. С. 67–82.
6. *Kosovskaya T.* Self-modificated predicate networks // International Journal on Information Theory and Applications. Vol. 22, No 3. 2015. P. 245–257.
7. *Косовская Т.М.* Подход к решению задачи построения многоуровневого описания классов на языке исчисления предикатов // Труды СПИИРАН, 2014. №3 (34). С. 204–217.
8. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. М.: Техносфера, 2005.
9. *Яне Б.* Цифровая обработка изображений, М.: Техносфера, 2007.
10. *Косовская Т.М.* Частичная выводимость предикатных формул как средство распознавания объектов с неполной информацией // Вестник СПбГУ. Сер. 10. 2009. Вып. 1. С. 74–84.
11. *Kosovskaya T.* Distance between objects described by predicate formulas // International Book Series. Information Science and Computing. Book 25. Mathematics of Distances and Applications (Michel Deza, Michel Petitjean, Krasimir Markov (eds)), ITHEA – Publisher, Sofia, Bulgaria, 2012. P. 153–159.
12. *Косовская Т.М.* Многоуровневые описания классов для уменьшения числа шагов решения задач распознавания образов, описываемых формулами исчисления предикатов // Вестн. С.-Петербург.ун-та. Сер. 10. 2008. Вып.1. С. 64–72.

MAXIMAL COMMON PREDICATE SUB-FORMULA EXTRACTION WITH THE USE OF MASLOV'S INVERSE METHOD

Petukhova N. D.

Abstract

The article is devoted to the describing of an algorithm which extract a maximal common up to the names of variables sub-formula of two elementary conjunctions of atomic predicate formulas. The offered algorithm uses the proposed earlier by the author modification of inverse method of S. Yu. Maslov as well as Ant tactics and concurrent processing. The problem of the extraction of a maximal common up to the names of variables sub-formula of predicate formulas has wide enough application while development of an effective algorithm solving an Artificial Intelligence problem permitting its description in the frameworks of predicate calculus language. Asymptotic estimates of the number of run steps for the described algorithm are formulated.

Keywords: *artificial intelligence, logic-objective approach to pattern recognition, predicate calculus, complexity theory, inverse method of S. Yu. Maslov, concurrent processing, partial deducibility.*

Петухова Нина Дмитриевна,
старший преподаватель кафедры математики Санкт-Петербургского государственного морского технического университета,
ndpetukhova@gmail.com