Вестник СПбГУ. Математика. Механика. Астрономия. 2024. Т. 11 (69). Вып. 4
УДК 510.63, 004.82, 004.85
MSC 03B70

Алгоритм выделения общих свойств объектов, описанных на языке исчисления предикатов с одним предикатным символом*

Ц. Чжоу, Т. М. Косовская

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

Для цитирования: *Чжоу Ц., Косовская Т. М.* Алгоритм выделения общих свойств объектов, описанных на языке исчисления предикатов с одним предикатным символом // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. 2024. Т. 11 (69). Вып. 4. С. 733–743. https://doi.org/10.21638/spbu01.2024.409

В задачах искусственного интеллекта, где исследуются сложные структурированные объекты, которые задаются свойствами своих элементов и отношениями между этими элементами, удобно использовать формулы исчисления предикатов, точнее элементарные конъюнкции атомарных предикатных формул. При этом встает задача выделения общих свойств таких объектов. Общие свойства сложных составных объектов задаются максимальными по количеству литералов формулами с переменными в качестве аргументов, которые с точностью до имен аргументов совпадают с подформулами исследуемых объектов, т. е. изоморфны этим подформулам. Задача проверки элементарных конъюнкций предикатных формул на изоморфизм является GI-полной задачей, т. е. полиномиально эквивалентна «открытой» задаче проверки двух графов на изоморфизм. Ранее авторами были разработаны алгоритмы проверки таких формул на изоморфизм. В настоящей статье предложен алгоритм нахождения наибольшей по количеству литералов подформулы с переменными в качестве аргументов, изоморфной подформулам двух элементарных конъюнкций предикатных формул, содержащих единственный предикатный символ. В такой постановке рассматриваемая задача полиномиально эквивалентна NP-трудной задаче выделения наибольшего общего подграфа двух графов. В частности, задача выделения наибольшего общего подграфа двух ориентированных графов является частным случаем рассматриваемой в статье задачи при условии, что предикатный символ двуместный. Задача выделения наибольшего общего подграфа двух графов является частным случаем рассматриваемой в статье задачи при условии, что предикатный символ двуместный и вместе с литералом $P(x, y)$ всегда присутствует литерал $P(y, x)$. Доказаны оценки вычислительной сложности предложенного алгоритма. Алгоритм реализован на языке *Python*.

Ключевые слова: изоморфизм элементарных конъюнкций предикатных формул, максимальная общая подформула, унификатор предикатных формул.

1. Введение. Во многих прикладных задачах искусственного интеллекта исследуются сложные структурированные объекты (ССО), представляющие собой множество элементов объекта, на котором заданы свойства этих элементов и отношения между ними [1, 2].

*Работа выполнена при поддержке Санкт-Петербургского государственного университета (проект № 95438429) и стипендии Китая на учебу за рубежом (грант № 202108620001).

В этом случае описание объекта может быть задано элементарной конъюнкцией атомарных предикатных формул. При исследовании множества ССО важным этапом является выделение общих свойств объектов из заданного множества. К таким задачам относятся, например, задача описания класса объектов [3], создание многоуровневого описания классов, существенно уменьшающего вычислительную сложность распознавания [4], построение онтологии в множестве ССО [5] и многие другие.

Основным этапом в процессе выделения наибольшего общего свойства двух объектов является проверка двух элементарных конъюнкций предикатных формул на изоморфизм. Эта задача полиномиально эквивалентна «открытой» задаче «Изоморфизм графов» [6], для которой не доказана ее NP-полнота и не найден полиномиальный по времени алгоритм решения.

Более того, проблема изоморфизма графов представляет собой сужение рассматриваемой проблемы, если в элементарной конъюнкции имеется только один предикат и он является двуместным.

Решению этой задачи посвящены различные исследования. Так, например, в [7] предложен алгоритм VF2 сопоставления графов с использованием набора правил выполнимости, что соответствует проверке унификаторов на противоречивость, описанной ниже. В [8] представлен новый алгоритм проверки изоморфизма графов с заданными свойствами, названный VF3, который на несколько порядков улучшает время выполнения по сравнению со своим предшественником.

В работе [9] были предложены возможные алгоритмы как для проверки формул на изоморфизм, так и для выделения наибольшего общего свойства двух объектов. Описания алгоритмов даны очень кратко.

В [10] представлены подробные алгоритмы проверки элементарных конъюнкций на изоморфизм и доказаны оценки числа шагов этих алгоритмов.

В настоящей работе предлагается подробный алгоритм выделения наибольшей общей подформулы для элементарных конъюнкций литералов, содержащих ровно один предикатный символ. Этот алгоритм является основой для алгоритма выделения наибольшей общей подформулы для элементарных конъюнкций литералов, содержащих произвольное количество предикатных символов.

Представленный алгоритм реализован на языке *Python*.

2. Основные определения

Определение 1 [9]. *Две элементарные конъюнкции атомарных формул исчисления предикатов* $F1(a_1, a_2, \ldots, a_n)$ *и* $F2(b_1, b_2, \ldots, b_n)$ *называются изоморфными*

$$F1(a_1, a_2, \ldots, a_n) \sim F2(b_1, b_2, \ldots, b_n),$$

если существуют такая элементарная конъюнкция $R(x_1, x_2, \ldots, x_n)$ *и подстановки аргументов* $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$ *и* $b_{j_1}, b_{j_2}, \ldots, b_{j_n}$ *формул* $F1(a_1, a_2, \ldots, a_n)$ *и* $F2(b_1, b_2, \ldots, b_n)$ *соответственно вместо всех вхождений переменных* $x_1, x_2, \ldots, x_n$ *формулы* $R(x_1, x_2, \ldots, x_n)$, *что результаты этих подстановок* $R(a_{i_1}, a_{i_2}, \ldots, a_{i_n})$ *и* $R(b_{j_1}, b_{j_2}, \ldots, b_{j_n})$ *совпадают с формулами* $F1(a_1, a_2, \ldots, a_n)$ *и* $F2(b_1, b_2, \ldots, b_n)$ *соответственно с точностью до порядка литералов.*

При этом полученные подстановки $\lambda_{R\ F1} = \{x_1 : a_{i_1}, x_2 : a_{i_2}, \ldots, x_n : a_{i_n}\}$ *и* $\lambda_{R\ F2} = \{x_1 : b_{i_1}, x_2 : b_{i_2}, \ldots, x_n : b_{i_n}\}$ *называются унификаторами формул* $F1(a_1, a_2, \ldots, a_n)$ *и* $F2(b_1, b_2, \ldots, b_n)$ *с формулой* $R(x_1, x_2, \ldots, x_n)$ *соответственно.*

В этом определении символами $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_n$ обозначены аргументы, которые могут быть как константами, так и переменными (или более сложными термами). Символами $x_1, x_2, \ldots, x_n$ обозначены переменные.

При проверке изоморфизма формул $F1(a_1, a_2, \ldots, a_n)$ и $F2(b_1, b_2, \ldots, b_n)$ в качестве формулы $R(x_1, x_2, \ldots, x_n)$ удобно взять одну из заданных формул с заменой констант на переменные (например, $F1(x_1, x_2, \ldots, x_n)$) и проверять на изоморфизм формулы $R(x_1, x_2, \ldots, x_n)$ и $F2(b_1, b_2, \ldots, b_n)$. Унификатором формул $R(x_1, x_2, \ldots, x_n)$ и $F1(a_1, a_2, \ldots, a_n)$ будет очевидная подстановка $\lambda_{R\ F1} = \{x_1 : a_1, x_2 : a_2, \ldots, x_n : a_n\}$, примененная к $R(x_1, x_2, \ldots, x_n)$. Ниже в качестве $R(x_1, x_2, \ldots, x_n)$ будем брать более короткую (с меньшим количеством литералов) формулу с заменой констант на переменные, а вторую из формул обозначать как $F(b_1, b_2, \ldots, b_n)$.

Определение 2 [11]. *Элементарная конъюнкция $R(x_1, x_2, \ldots, x_n)$ называется наибольшей общей (с точностью до имен переменных) подформулой двух элементарных конъюнкций $F1(a_1, a_2, \ldots, a_n)$ и $F2(b_1, b_2, \ldots, b_n)$, если она изоморфна некоторым подформулам этих элементарных конъюнкций, но после добавления в нее хоть одного литерала становится не изоморфной ни одной подформуле либо формулы $F1(a_1, a_2, \ldots, a_n)$, либо формулы $F2(b_1, b_2, \ldots, b_n)$.*

Определение 3. *Пусть $R(x_1, x_2, \ldots, x_n)$ и $F(b_1, b_2, \ldots, b_n)$ — две элементарные конъюнкции предикатных формул, причем $R(x_1, x_2, \ldots, x_n)$ содержит только переменные в качестве аргументов.*

Подстановка $\{\overline{x} : \overline{b}\}$, где $\overline{x}$ — список некоторых переменных из $R(x_1, x_2, \ldots, x_n)$, $\overline{b}$ — список некоторых различных констант из $F(b_1, b_2, \ldots, b_n)$, называется частичным унификатором формул $R(x_1, x_2, \ldots, x_n)$ и $F(b_1, b_2, \ldots, b_n)$, если результат применения этой подстановки к формуле $R(x_1, x_2, \ldots, x_n)$ содержит подформулу, совпадающую с некоторой подформулой $F(b_1, b_2, \ldots, b_n)$ с точностью до порядка литералов. Частичный унификатор обозначать как $\lambda_{pR\ F}$.

Для дальнейшего описания алгоритма, которому посвящена эта статья, потребуются некоторые обозначения.

Обозначение. *Пусть элементарная конъюнкция $F(a_1, a_2, \ldots, a_n)$ содержит только один t-местный предикатный символ P, причем каждый аргумент входит в литерал не более одного раза. Выберем два аргумента a_i и a_j формулы $F(a_1, a_2, \ldots, a_n)$. Обозначим посредством $lits_{u,v}(a_i, a_j)_F$ множество литералов из $F(a_1, a_2, \ldots, a_n)$, в которых оба аргумента a_i и a_j находятся в позициях u и v соответственно.*

Например, для формулы $F(a, b, c, d, e, f) = P(a, b, c)\ \&\ P(a, c, e)\ \&\ P(b, d, f)\ \&\ P(c, b, a)\ \&\ P(d, e, f)\ \&\ P(a, c, b)$ множество $lits_{1,2}(a, c)_F = \{P(a, c, e),\ P(a, c, b)\}$.

Всего количество таких множеств составляет $A_t^2 \cdot \frac{n(n-1)}{2}$, но большинство множеств будут пустыми. Количество непустых множеств не превосходит $C_t^2 \cdot L = \frac{1}{2}t\ (t-1)\ L$, где L — количество литералов формулы $F(b_1, b_2, \ldots, b_n)$.

При нахождении наибольшей общей подформулы будут генерироваться возможные унификаторы двух подформул. При их генерировании важно, чтобы они не содержали противоречивые подстановки.

Определение 4. *Две подстановки называются противоречивыми, если для одной и той же переменной x нашли две разные константы a_1 и a_2, т. е. $\{x :$*

$a_1, x : a_2\}$, *или для разных переменных* x_1 *и* x_2 *нашли одинаковую константу* a, *т. е.* $\{x_1 : a, x_2 : a\}$.

3. Постановка задачи. Задана пара элементарных конъюнкций атомарных предикатных формул $F1(a_1, a_2, \ldots, a_m)$ и $F2(b_1, b_2, \ldots, b_n)$ с единственным предикатным символом и константами в качестве аргументов[1]. В каждом литерале имена всех аргументов различны.

Найти максимальную по количеству литералов элементарную конъюнкцию $R(x_1, x_2, \ldots, x_k)(k \leq \min(m, n))$, для которой в $F1(a_1, a_2, \ldots, a_m)$ и $F2(b_1, b_2, \ldots, b_n)$ имеются подформулы, изоморфные $R(x_1, x_2, \ldots, x_k)$.

4. Описание алгоритма нахожения наибольшей общей подформулы двух заданных элементарных конъюнкций. Не умаляя общности, можно считать, что $F1(a_1, a_2, \ldots, a_m)$ содержит меньшее количество литералов, чем $F2(b_1, b_2, \ldots, b_n)$. Заменив в $F1(a_1, a_2, \ldots, a_m)$ константы на переменные, получаем формулу $R(x_1, x_2, \ldots, x_m)$ с унификатором $\lambda_{R\ F1} = \{x_1 : a_1, x_2 : a_2, \ldots, x_m : a_m\}$. Формулу $F2(b_1, b_2, \ldots, b_n)$ будем обозначать как $F(b_1, b_2, \ldots, b_n)$.

В процессе работы алгоритма текущие значения (и количество их аргументов) формул $R(x_1, x_2, \ldots, x_m)$ и $F(b_1, b_2, \ldots, b_n)$ будут меняться, поэтому, если это не вызовет разночтения, будем записывать R и F.

Кроме того, в процессе работы алгоритма потребуется хранить найденные (но не обязательно максимальные) подформулы формулы F, изоморфные подформулам R. Для их хранения будем использовать множество R_isom.

1. Для каждой позиции аргументов (u, v) составляем списки $lits_{u,v}(x_{i_1}, x_{j_1})_R$ и $lits_{u,v}(b_{i_2}, b_{j_2})_F$ литералов из R и F, в которых оба аргумента x_{i_1} и x_{j_1} (соответственно b_{i_2} и b_{j_2})находятся в позициях u и v соответственно.

2. Если список $lits_{u,v}(x_i, x_j)_R$ непуст, то, используя $lits_{u,v}(x_i, x_j)_R$ и $lits_{u,v}(b_i, b_j)_F$, выбираем пары аргументов в R и F, имеющие наибольшее количество одновременных вхождений в один литерал. Убираем литералы с выбранными аргументами из множеств $lits_{u,v}(x_i, x_j)_R$ и $lits_{u,v}(b_i, b_j)_F$.

Литералы, в которые входят выбранные пары аргументов, ставим в очереди для R и для F.

Организовываем двойной цикл по очередям для R и F. В цикле выполняем пп. 3–8.

3. Берем из очередей пару литералов L_R и L_F для первоначальных значений формул R и F.

4. Находим частичный унификатор L_R и L_F. Значение этого частичного унификатора используется в качестве корневого узла дерева нахождения унификаторов.

5. Применяем найденный частичный унификатор к R. Помечаем использованные в этом унификаторе константы как «*used*».

6. В R находим литералы, в которых нет переменных. Используя ранее найденный частичный унификатор, восстанавливаем в R конъюнкцию этих литералов с переменными и записываем ее (вместе с унификатором) в множество R_isom. Так как эти литералы обязательно присутствуют в F, то удаляем их из R и F.

7. Проверяем, есть ли в R переменные.

[1]Имена констант в разных формулах могут совпадать, причем константы с одинаковыми именами могут стоять на разных местах.

Если переменных нет или есть в R, но в F нет литералов, совпадающих с литералами из $lits\_maxR$ в тех позициях, в которых R имеет константы, то записываем в множество $FOUND$ пары литералов, для которых искали формулу R.

(a) Если очередь непуста, то переходим к п. 3.

(b) Если очередь пуста, то переходим к п. 2[2].

8. Организуем два вложенных цикла по литералам из текущих значений R и F.

(a) Создаем множество $lits\_maxR$, выделяя в формуле R все литералы, одновременно содержащие максимальное количество констант, помеченных как «*used*». Удаляем эти литералы $lits\_maxR$ из формулы R. Если множество $lits\_maxR$ непусто, то переходим к п. 8b, иначе, к п. 3.

(b) В формуле F находим литералы $lits\_maxF$, упорядоченный список констант в которых совпадает со списком констант из L_R, за исключением позиций, в которых L_R содержит переменные. В этих позициях найденный литерал не содержит констант, помеченных как «*used*». Если множество $lits\_maxF$ непусто, то переходим к п. 8c, иначе, к п. 3.

(c) Находим частичные унификаторы $\lambda_{lits\_maxR,lits\_maxF}$ конъюнкций литералов для R и F: $lits\_maxR$ и $lits\_maxF$.

i. Проверка непротиворечивости возможных унификаторов реализуется с помощью алгоритма *CONS-T*[3] формирования непротиворечивого дерева возможных унификаторов. Добавляем к уже имеющемуся унификатору подстановки, которые не противоречат подстановкам в имеющемся унификаторе. Составляем копию меток «*used*», в которую добавляем найденные константы. Эта копия будет действительна только внутри одной ветки обхода дерева, формируемого в п. 8c.

Берем первый унификатор из только что найденных и выполняем те же действия, которые описаны в пп. 5 и 6 этого алгоритма, т. е. применяем найденный частичный унификатор к R. Записываем конъюнкцию (вместе с унификатором) в множество R_isom, из R и F удаляем литералы.

ii. Проверяем: в R нет переменных или в R есть переменные, но в F нет литералов, совпадающих с литералами из $lits\_maxR$ в тех позициях, в которых R имеет константы.

Если да, то для текущего значения R находим $lits\_maxR$, $lits\_maxF$ и соответствующие унификаторы следующего уровня.

В противном случае поиск в этой ветви заканчивается. Отмена замен в R. Добавляем соответствующий унификатор к R_ISOM. Прибавляем пару литералов, для которой (из п. 3) искали R, к множеству $FOUND$. Возврат к сестринской (на том же уровне) вершине дерева (п. 8ci). Если такой нет, то возврат на предыдущий уровень дерева (п. 3). Берем следующую пару литералов, которой нет в множестве $FOUND$.

5. Описание алгоритма *CONS-T* формирования непротиворечивого дерева возможных унификаторов. На вход алгоритма подаются текущие значения формул R и F и два их частичных унификатора: $UP1$ $\{\overline{x} : \overline{a}\}$ (унификатор, находящийся в рассматриваемой вершине) и $UP2$ $\{\overline{y} : \overline{b}\}$ (возможный унификатор для переменных, еще не имеющих значения), где $\overline{x}$ и $\overline{y}$ — списки переменных; $\overline{a}$ и $\overline{b}$ — списки констант.

[2]Таким образом, переход к п. 8 происходит, если в R есть переменные и в F имеются литералы, совпадающие с литералами из $lits\_maxR$ в тех позициях, в которых R имеет константы.

[3]Алгоритм описан ниже.

1. Проверяем, есть ли в списках $\overline{x}$ и $\overline{y}$ одинаковые переменные, причем им соответствуют различные константы из списков $\overline{a}$ и $\overline{b}$, или в списках $\overline{a}$ и $\overline{b}$ есть одинаковые константы, соответствующие разным переменным из списков $\overline{x}$ и $\overline{y}$. В любом из этих случаев эти частичные унификаторы противоречивы.

2. Из $UP2$ удаляем все подстановки, противоречащие хоть одной подстановке $UP1$.

3. Внутри $UP2$ находим все частичные унификаторы, в которых одной и той же переменной соответствуют разные константы или одна и та же константа соответствует разным переменным.

4. Частичные унификаторы, которые не противоречат подстановкам ни $UP1$, ни $UP2$, перемещаем в копии $UP1$.

5. К частичному унификатору из $UP1$ добавляем копии $UP1$, получаем при этом дочерние вершины по отношению к исходной вершине в дереве.

Процесс формирования дерева нахождения возможных унификаторов представлен на рисунке.

Ввиду недостатка места на рисунке использованы следующие обозначения, в которых первая цифра обозначает номер пункта алгоритма:

3. L_{R_i} означает, что из очереди QR берем i-й литерал, L_{F_j} — из очереди QF — j-й литерал;

λ_ilevel_j — j-й частичный унификатор i-го уровня. Если нет индекса j, значит, на i-м уровне есть только одни унификатор;

4. λ_1level — находим частичный унификатор первого уровня;

5. $used$ — помечаем использованные в унификаторе λ_ilevel_j константы как «$used$», замены в R — применяем найденный частичный унификатор λ_ilevel_j к R;

6. const — множество литералов с константами в R. $R := R \setminus \text{const}$ — удаляем литералы с константами из R. $F := F \setminus \text{const}$ — удаляем из F тот же литерал, что и в R. Записываем в R_ISOM — с помощью ранее найденного частичного унификатора λ_ilevel_j восстанавливаем в R конъюнкцию литералов const с переменными и записываем ее (вместе с унификатором) в множество R_isom;

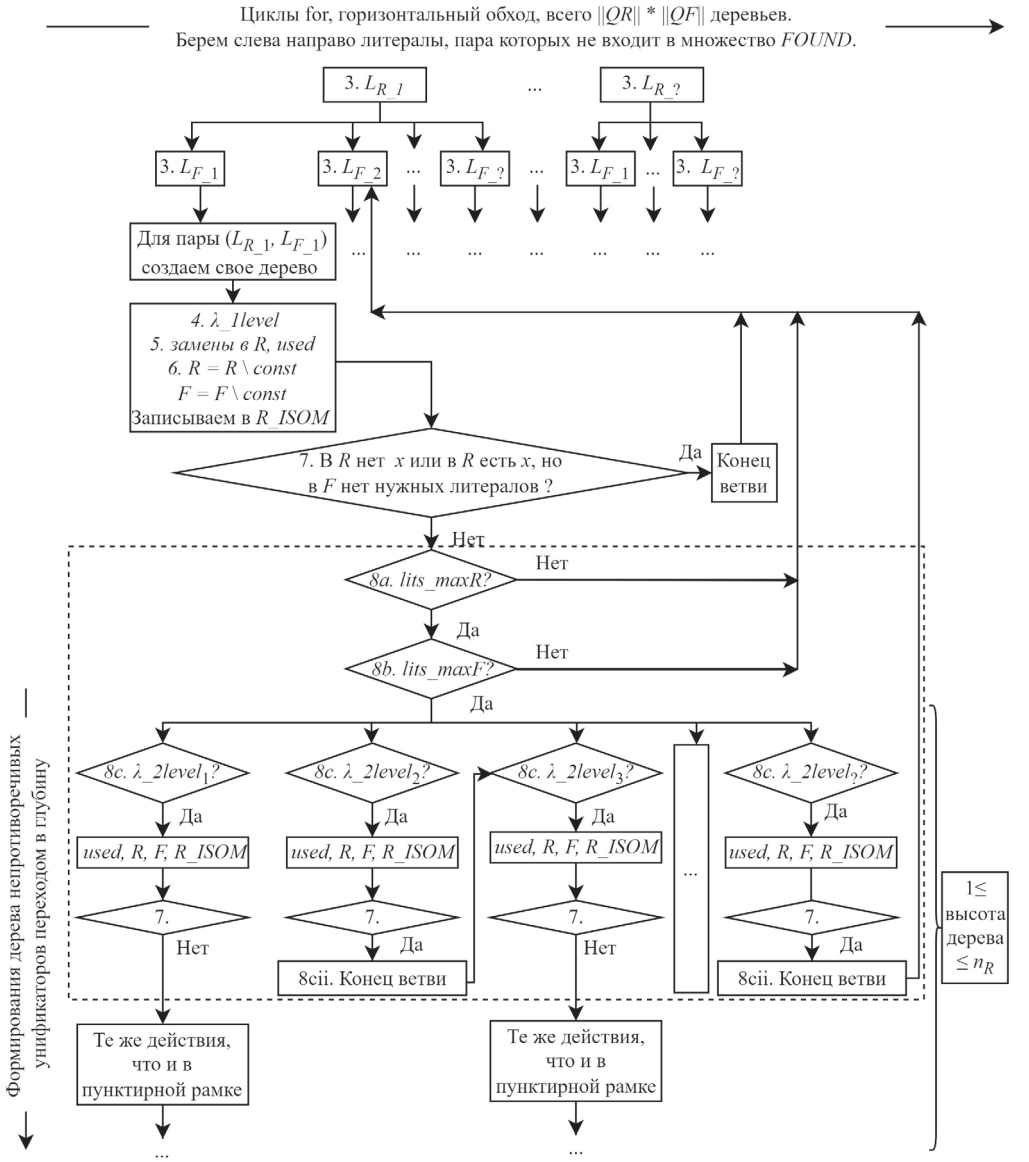
7. В R нет x или в R есть x, но в F нет нужных литералов? — В R нет переменных или в R есть переменные, но в F нет литералов, совпадающих с литералами из $lits\_maxR$ в тех позициях, в которых R имеет константы?

8a. $lits\_maxR$? — множество $lits\_maxR$ непусто? 8b. $lits\_maxF$? — множество $lits\_maxF$ непусто? 8c. λ_ilevel_j? — унификатор λ_ilevel_j непуст? Если в пунктах 8a, 8b, 8c множества $lits\_maxR$ и $lits\_maxF$ унификатор λ_ilevel_j непусты, выполняются следующие пункты. Иначе возвращаемся к сестринской (на предыдущем уровне) вершине дерева или на предыдущий уровень дерева. $used, R, F, R\_ISOM$ — такие же, как в пунктах 5 и 6 выше.

Конец ветви — Конец работы в этой ветви. Отмена замен в R. Добавляем унификатор к R_ISOM. Прибавляем пару литералов, для которой искали R, к множеству $FOUND$.

$1 \leq$ высота дерева $\leq n_R$ — в худшем случае, высота дерева не превосходит количество переменных n_R в R, присвоить каждой переменной в R одно из значений (констант) из F. В лучшем случае, высота дерева равна 1, для всех переменных сразу находим значения.

Замечание. Следует отметить, что предложенный алгоритм находит все наибольшие (по включению) общие подформулы заданных формул. Однако мож-



Процесс формирования непротиворечивого дерева возможных унификаторов.

но прекратить работу алгоритма, как только одна из наибольших формул будет найдена.

6. Вычислительная сложность основного алгоритма. Пункты 1–8b алгоритма выполняются за полиномиальное (не более квадратичного) число шагов. Основной вклад в оценку вычислительной сложности вносит реализация пункта 8ci, а именно проверка на противоречивость унификаторов алгоритма.

В худшем случае после первого выполнения пункта 8ci только для одной переменной найдено значение. Здесь происходит исчерпывающий поиск: формирование

и обход дерева с высотой, равной количеству переменных n_R в R и степенями ветвления $n_R \cdot n_F$, $(n_F-1)\cdot(n_R-1)\cdot n_F \cdot n_R$, $(n_F-2)\cdot(n_R-2)\cdot(n_F-1)\cdot(n_R-1)\cdot n_F \cdot n_R$,..., $(n_F-i)\cdot(n_R-i)\cdot\ldots\cdot n_F\cdot n_R$, где $0 \le i \le n_R-1, n_R \le n_F$. Таким образом, верхняя граница вычислительной сложности алгоритма равна $n_F \cdot n_R + [(n_F-1)\cdot(n_R-1)\cdot n_F \cdot n_R] + [(n_F-2)\cdot(n_R-2)\cdot(n_F-1)\cdot(n_R-1)\cdot n_F\cdot n_R] + \ldots + [(n_F-i)\cdot(n_R-i)\cdot\ldots\cdot n_F\cdot n_R]$, т. е. $O((n_R \cdot n_F)^{n_R})$, где n_R, n_F — количество аргументов в формулах R, F соответственно.

7. Наибольший общий подграф. Задача нахождения наибольшего общего (с точностью до имен вершин) подграфа двух графов является одной из основных NP-трудных задач. Она представляет собой сужение рассматриваемой проблемы, если в элементарной конъюнкции есть только один предикат и он двухместный.

Предложенный в работе алгоритм можно использовать для нахождения наибольшего общего графа. Пусть G_1, G_2 — ориентированные графы, тогда вычислительная сложность выделения наибольшего общего подграфа G_1, G_2 будет $O((n_1 \cdot n_2)^{min(n_1,n_2)})$, где n_1, n_2 — количество вершин в графах G_1, G_2 соответственно.

8. Применение основного алгоритма. Пусть имеются две формулы:

$F1(a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_{10}) =$
$P(a_1,a_3,a_2)$ & $P(a_2,a_1,a_5)$ & $P(a_2,a_5,a_8)$ & $P(a_2,a_1,a_8)$ & $P(a_3,a_4,a_1)$ &
$P(a_3,a_5,a_1)$ & $P(a_3,a_9,a_4)$ & $P(a_3,a_9,a_5)$ & $P(a_3,a_9,a_1)$ & $P(a_4,a_3,a_6)$ &
$P(a_4,a_6,a_5)$ & $P(a_5,a_2,a_3)$ & $P(a_5,a_2,a_4)$ & $P(a_5,a_3,a_7)$ & $P(a_5,a_3,a_{10})$ &
$P(a_5,a_4,a_7)$ & $P(a_5,a_4,a_{10})$ & $P(a_5,a_7,a_2)$ & $P(a_5,a_{10},a_2)$ & $P(a_6,a_4,a_9)$ &
$P(a_6,a_7,a_4)$ & $P(a_6,a_9,a_7)$ & $P(a_7,a_5,a_6)$ & $P(a_7,a_6,a_{10})$ & $P(a_8,a_2,a_{10})$ &
$P(a_9,a_6,a_3)$ & $P(a_9,a_{10},a_6)$ & $P(a_9,a_{10},a_3)$ & $P(a_{10},a_7,a_9)$ & $P(a_{10},a_5,a_9)$ &
$P(a_{10},a_8,a_7)$ & $P(a_{10},a_8,a_5)$ & $P(a_{10},a_8,a_9)$,

$F2(b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8) =$
$P(b_1,b_4,b_2)$ & $P(b_2,b_1,a_6)$ & $P(b_2,b_6,b_3)$ & $P(b_2,b_1,b_3)$ & $P(b_3,b_2,b_8)$ &
$P(b_4,b_5,b_1)$ & $P(b_4,b_6,b_1)$ & $P(b_4,b_7,b_5)$ & $P(b_4,b_7,b_6)$ & $P(b_4,b_7,b_1)$ &
$P(b_5,b_4,b_7)$ & $P(b_5,b_7,b_6)$ & $P(b_6,b_2,b_5)$ & $P(b_6,b_2,b_4)$ & $P(b_6,b_5,b_8)$ &
$P(b_6,b_4,b_8)$ & $P(b_6,b_8,b_2)$ & $P(b_7,b_5,b_4)$ & $P(b_7,b_8,b_5)$ & $P(b_7,b_8,b_4)$ &
$P(b_8,b_3,b_6)$ & $P(b_8,b_6,b_7)$ & $P(b_8,b_3,b_7)$.

Так как формула $F2(\overline{b})$ имеет меньшее количество литералов, то берем ее в качестве $R(\overline{x})$ с заменой b_i на x_i.

В результате работы программы были получены 134 пары изоморфных подформул приведенных формул. Ни одна из выделенных формул $R(\overline{x}_i)$ $(i = 1,\ldots,134)$ не является подформулой никакой другой.

Наибольшая по количеству литералов подформула $R(\overline{x}_i)$, задающая общее свойство двух заданных, содержит 19 литералов и имеет вид

$R(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) =$
$P(x_1,x_4,x_2)$ & $P(x_2,x_1,x_3)$ & $P(x_2,x_1,x_6)$ & $P(x_2,x_6,x_3)$ & $P(x_3,x_2,x_8)$ & $P(x_4,x_5,x_1)$ & $P(x_4,x_6,x_1)$ & $P(x_4,x_7,x_1)$ & $P(x_4,x_7,x_5)$ & $P(x_4,x_7,x_6)$ & $P(x_6,x_2,x_4)$ & $P(x_6,x_2,x_5)$ & $P(x_6,x_4,x_8)$ & $P(x_6,x_5,x_8)$ & $P(x_6,x_8,x_2)$ & $P(x_7,x_8,x_4)$ & $P(x_8,x_3,x_6)$ & $P(x_8,x_3,x_7)$ & $P(x_8,x_6,x_7)$.

Подформула формулы $F2$ получается применением замены x_i на b_i при $i = 1,2,3,4,5,6,7,8$ и имеет вид $F'2(b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8) = R(b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8)$.

$F2(b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8) =$

$P(b_1, b_4, b_2)$ & $P(b_2, b_1, b_3)$ & $P(b_2, b_1, b_6)$ & $P(b_2, b_6, b_3)$ & $P(b_3, b_2, b_8)$ & $P(b_4, b_5, b_1)$ & $P(b_4, b_6, b_1)$ & $P(b_4, b_7, b_1)$ & $P(b_4, b_7, b_5)$ & $P(b_4, b_7, b_6)$ & $P(b_6, b_2, b_4)$ & $P(b_6, b_2, b_5)$ & $P(b_6, b_4, b_8)$ & $P(b_6, b_5, b_8)$ & $P(b_6, b_8, b_2)$ & $P(b_7, b_8, b_4)$ & $P(b_8, b_3, b_6)$ & $P(b_8, b_3, b_7)$ & $P(b_8, b_6, b_7)$.

Унификатором формулы R и $F1$ является замена x_i на a_j при $i = 1, 2, 3, 4, 5, 6, 7, 8$, $j = 1, 2, 8, 3, 4, 5, 9, 10$ и имеет вид

$F'1(a_1, a_2, a_8, a_3, a_4, a_5, a_9, a_{10}) =$

$P(a_1, a_3, a_2)$ & $P(a_2, a_1, a_8)$ & $P(a_2, a_1, a_5)$ & $P(a_2, a_5, a_8)$ & $P(a_8, a_2, a_{10})$ & $P(a_3, a_4, a_1)$ & $P(a_3, a_5, a_1)$ & $P(a_3, a_9, a_1)$ & $P(a_3, a_9, a_4)$ & $P(a_3, a_9, a_5)$ & $P(a_5, a_2, a_3)$ & $P(a_5, a_2, a_4)$ & $P(a_5, a_3, a_{10})$ & $P(a_5, a_4, a_{10})$ & $P(a_5, a_{10}, a_2)$ & $P(a_9, a_{10}, a_3)$ & $P(a_{10}, a_8, a_5)$ & $P(a_{10}, a_8, a_9)$ & $P(a_{10}, a_5, a_9)$.

А также замена x_i на a_j при $i = 1, 2, 3, 4, 5, 6, 7, 8$, $j = 9, 3, 1, 10, 7, 5, 8, 2$ и имеет вид

$F''1(a_9, a_3, a_1, a_{10}, a_7, a_5, a_8, a_2) =$

$P(a_9, a_{10}, a_3)$ & $P(a_3, a_9, a_1)$ & $P(a_3, a_9, a_5)$ & $P(a_3, a_5, a_1)$ & $P(a_1, a_3, a_2)$ & $P(a_{10}, a_7, a_9)$ & $P(a_{10}, a_5, a_9)$ & $P(a_{10}, a_8, a_9)$ & $P(a_{10}, a_8, a_7)$ & $P(a_{10}, a_8, a_5)$ & $P(a_5, a_3, a_{10})$ & $P(a_5, a_3, a_7)$ & $P(a_5, a_{10}, a_2)$ & $P(a_5, a_7, a_2)$ & $P(a_5, a_2, a_3)$ & $P(a_8, a_2, a_{10})$ & $P(a_2, a_1, a_5)$ & $P(a_2, a_1, a_8)$ & $P(a_2, a_5, a_8)$.

Если две формулы $F1$ и $F2$ являются описаниями двух изображений, то тот факт, что в более «длинной» формуле $F1$ имеется две подформулы, изоморфные выделенному общему свойству двух формул, соответствует тому, что, например, в $F2$ присутствует одно изображение некоторой детали, а в $F1$ изображение этой детали присутствует дважды.

9. Заключение. В статье разработан алгоритм выделения наибольшей общей (с точностью до имен аргументов) подформулы двух элементарных конъюнкций. Реализация была осуществлена на языке программирования *Python* [12].

Выделение таких подформул является важной актуальной задачей поиска общих свойств сложно структурированных объектов, описываемых с помощью языка исчисления предикатов, при решении таких задач, как:

- построение уровневого описания классов, позволяющего существенно снизить вычислительную сложность распознавания ССО [3, 4];
- нечеткое распознавание ССО [13];
- создание онтологии для ССО [5].

Литература

1. Нильсон Н. *Искусственный интеллект: методы поиска решений*, пер. с англ. Москва, Мир (1973).

2. Рассел С., Норвиг П. *Искусственный интеллект: современный подход, 4-е изд.*, пер. с англ. Москва, Санкт-Петербург, Диалектика (2021).

3. Kosovskaya T. Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity. In: *Intelligent System*. Open access peer-reviewed chapter. Edited by Chatchawal Wongchoosuk Kasetsart University (2018). Доступно на: https://www.intechopen.com/books/intelligent-system/predicate-calculus-as-a-tool-for-ai-problems-solution-algorithms-and-their-complexity (дата обращения: 19.12.2023).

4. Косовская Т. М. Многоуровневые описания классов для уменьшения числа шагов решения задач распознавания образов, описываемых формулами исчисления предикатов. *Вестник Санкт-Петербургского университета. Сер. 10. Прикладная математика. Информатика. Процессы управления* **10**, вып. 2, 62–70 (2008).

5. Косовская Т. М., Косовский Н. Н. Выделение общих свойств объектов для создания логических онтологий. *Вестник Санкт-Петербургского университета. Прикладная математи-*

ка. Информатика. Процессы управления **18**, вып. 1, 37–51 (2022). https://doi.org/10.21638/11701/spbu10.2022.103

6. Косовская Т. М., Косовский Н. Н. Полиномиальная эквивалентность задач изоморфизм предикатных формул и изоморфизм графов. *Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия* **6** (64), вып. 3, 430–439 (2019). https://doi.org/10.21638/11701/spbu01.2019.308

7. Cordella L. P., Foggia P., Sansone C., Vento M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** (10), 1367–1372 (2004). https://doi.org/doi:10.1109/tpami.2004.75

8. Carletti V., Foggia P., Saggese A., Vento M. Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **40** (4), 804–818 (2018). https://doi.org/doi:10.1109/tpami.2017.2696940

9. 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* **13**, вып. 3, 250–263 (2017). https://doi.org/10.21638/11701/spbu10.2017.303

10. Kosovskaya T., Zhou J. Algorithms for checking two elementary conjunctions isomorphism. *Proceedings of the 14^{th} International Conference "Computer Science and Information Technologies" (CSIT-2023)*, September 25–30, 2023, Yerevan, Armenia. https://doi.org/10.51408/csit2023_01

11. Косовская Т. М. Частичная выводимость предикатных формул как средство распознавания объектов с неполной информацией. *Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления* **10**, вып. 1, 74–84 (2009).

12. Matthes E. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 3^{rd} Edition*. San Francisco, No Starch Press (2023).

13. Kosovskaya T. Fuzzy Recognition by Logic-Predicate Network. *Advances in Science, Technology and Engineering Systems Journal* **5** (4), 686–699 (2020). https://doi.org/10.25046/aj050482

Статья поступила в редакцию 7 февраля 2024 г.;
доработана 19 марта 2024 г.;
рекомендована к печати 23 мая 2024 г.

Контактная информация:

Чжоу Цзюань — аспирант; https://orcid.org/0009-0007-2350-987X, st103098@student.spbu.ru
Косовская Татьяна Матвеевна — д-р физ.-мат. наук, проф.;
https://orcid.org/0000-0001-8330-4816, kosovtm@gmail.com

Algorithm for extraction common properties of objects described in the predicate calculus language with a single predicate symbol*

J. Zhou, T. M. Kosovskaya

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

For citation: Zhou J., Kosovskaya T. M. 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*, 2024, vol. 11 (69), issue 4, pp. 733–743. https://doi.org/10.21638/spbu01.2024.409 (In Russian)

In artificial intelligence problems, connected with the study of complex structured objects which are described in the terms of properties of their elements and relationships between these elements, it is convenient to use predicate calculus formulas, more precisely elementary conjunctions of atomic predicate formulas. In such a case, the problem of extraction common properties of objects arises. The common properties of complex structured objects are set by formulas with variables as arguments, which, up to the names of the arguments, coincide with the subformules of the objects under study, that is, are isomorphic to these subformules. In the case of a single predicate symbol in the descriptions of objects, the

*The work is supported by the St. Petersburg State University (project no. 95438429) and Chinese scholarship for study abroad (grant no. 202108620001).

problem under consideration is polynomial equivalent to the NP-hard problem of extraction the largest common subgraph of two graphs. An algorithm for finding the largest by the number of literals of a subformula with variables as arguments, isomorphic to the subformules of two elementary conjunctions of predicate formulas containing a single predicate symbol is proposed in this paper. Estimates of the computational complexity of the proposed algorithm are proved. The algorithm is implemented in Python.

Keywords: isomorphism of elementary conjunctions of predicate formulas, maximal common subformula, unifier of predicate formulas.

References

1. Nilson Nils J. *Problem-solving methods in artificial intelligence*. New York, McGraw-Hill Book Company Press (1971). [Rus. ed.: Nilson N. *Iskusstvennyi intellect: metody poisra reshenii*. Moscow, Mir Publ. (1973)].

2. Russel S. J., Norvig P. *Artificial Intelligence: A Modern Approach, Fourth Edition*. London, Pearson Education, Inc. (2021). [Rus. ed.: Russel S. J., Norvig P. *Iskusstvennyi intellekt: sovremennyi podhod*. Moscow, St. Petersburg, Dialektika Publ. (2021)].

3. Kosovskaya T. Predicate Calculus as a Tool for AI Problems Solution: Algorithms and Their Complexity. In: *Intelligent System*. Open access peer-reviewed chapter. Edited by Chatchawal Wongchoosuk Kasetsart University (2018). Available at: https://www.intechopen.com/books/intelligent-system/predicate-calculus-as-a-tool-for-ai-problems-solution-algorithms-and-their-complexity (accessed December 19, 2023).

4. Kosovskaya T. M. Level descriptions of classes for decreasing step number of pattern recognition problem solving described by predicate calculus formulas. *Vestnik of Saint Petersburg University. Ser. 10. Applied Mathematics. Computer Science. Control Processes* **10**, iss. 2, 62–70 (2008). (In Russian)

5. Kosovskaya T. M., Kosovskii N. N. Extraction of common properties of objects for creation of a logic ontology. *Vestnik of Saint Petersburg University.Applied Mathematics. Computer Science. Control Processes* **18**, iss. 1, 37–51 (2022). https://doi.org/10.21638/11701/spbu10.2022.103 (In Russian)

6. Kosovskaya T. M., Kosovskii N. N. Polynomial equivalence of the problems predicate formulas isomorphism and graph isomorphism. *Vestnik of Saint Petersburg University. Mathematics. Mechanics. Astronomy* **6** (64), iss. 3, 430–439 (2019). https://doi.org/10.21638/11701/spbu01.2019.308 (In Russian)

7. Cordella L. P., Foggia P., Sansone C., Vento M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** (10), 1367–1372 (2004). https://doi.org/doi:10.1109/tpami.2004.75

8. Carletti V., Foggia P., Saggese A., Vento M. Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3. *IEEE Transaction on Pattern Analysis and Machine Intelligence* **40** (4), 804–818 (2018). https://doi.org/doi:10.1109/tpami.2017.2696940

9. 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* **13**, iss. 3, 250–263 (2017). https://doi.org/10.21638/11701/spbu10.2017.303 (In Russian)

10. Kosovskaya T., Zhou J. Algorithms for checking two elementary conjunctions isomorphism. *Proceedings of the 14^{th} International Conference "Computer Science and Information Technologies" (CSIT-2023)*, 25–30, September, 2023, Yerevan, Armenia. https://doi.org/10.51408/csit2023_01

11. Kosovskaya T. M. Partial deduction of predicate formula as an instrument for recognition of an object with incomplete description. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes* **10**, iss. 1, 74–84 (2009). (In Russian)

12. Matthes E. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming, 3^{rd} Edition*. San Francisco, No Starch Press (2023).

13. Kosovskaya T. Fuzzy Recognition by Logic-Predicate Network. *Advances in Science, Technology and Engineering Systems Journal* **5** (4), 686–699 (2020). https://doi.org/10.25046/aj050482

Received: February 7, 2024
Revised: March 19, 2024
Accepted: May 23, 2024

Authors' information:

Zhou Juan — https://orcid.org/0009-0007-2350-987X, st103098@student.spbu.ru
Tatiana M. Kosovskaya — https://orcid.org/0000-0001-8330-4816, kosovtm@gmail.com