УДК 004.89, 004.682, 004.832.2

СОВМЕСТНОЕ ПРИМЕНЕНИЕ МЕТОДОВ РАСПРОСТРАНЕНИЯ ОГРАНИЧЕНИЙ И СТРУКТУРНОЙ ДЕКОМПОЗИЦИИ ДЛЯ АПРИОРНОГО АНАЛИЗА ЗАПРОСОВ К ОНТОЛОГИЯМ

А.А. Зуенко[1], П.А. Ломов[2]

Институт информатики и математического моделирования
ФИЦ «Кольский научный центр Российской академии наук», Апатиты, Россия,
[1]zuenko@iimm.ru, [2]lomov@iimm.ru

Аннотация

В отличие от реляционных СУБД, ориентированных на поддержку ссылочной целостности интенсивно меняющихся данных, при использовании RDF-репозиториев для хранения онтологий нет необходимости оперировать такими атомарными структурами, как элементарный кортеж таблицы, а можно использовать более подходящие структуры для группирования и обобщения информации. В статье онтология рассматривается как совокупность отношений (унарных и бинарных), выраженных с помощью специализированных матрицеподобных структур – *C*-систем, что позволяет ставить и решать задачи вывода на онтологии как задачи удовлетворения ограничений. Рассматриваемая в статье задача априорного анализа и упрощения SPARQL-запросов решается для онтологий, которые разработаны с применением онтологических паттернов содержания, что обеспечивает предсказуемость структуры потенциальных запросов. Каждому паттерну сопоставляется совокупность шаблонов SPARQL-запросов. Разработан метод априорного анализа и преобразования шаблонов SPARQL-запросов в форму, позволяющую ускорить последующее исполнение конкретизированных пользовательских запросов. Метод основан на совместном применении методов структурной декомпозиции и авторских методов удовлетворения нечисловых ограничений. Применение методов структурной декомпозиции дает возможность разбивать шаблон SPARQL-запроса на части, распараллеливать выполнение подзадач, что особенно актуально при обращении к RDF-репозиториям большого объёма. Для соединения отношений, выраженных в виде совокупности *C*-систем, применяются авторские методы удовлетворения нечисловых ограничений, которые представляют собой модификации известных методов достижения совместности в вершинах и по дугам. Предлагаемый подход к представлению онтологии, а также к организации процедур вывода на онтологиях, позволяет снизить потребный объём памяти на хранение онтологии и обеспечить приемлемую скорость выполнения SPARQL-запросов.

Ключевые слова: *онтология, паттерны онтологического проектирования, задача удовлетворения ограничений, программирование в ограничениях, SPARQL, RDF-репозиторий.*

Цитирование: *Зуенко А.А.* Совместное применение методов распространения ограничений и структурной декомпозиции для априорного анализа запросов к онтологиям. / А.А. Зуенко, П.А. Ломов // Онтология проектирования. – 2018. – Т.8, №4(30). – С.571-593. – DOI: 10.18287/2223-9537-2018-8-4-571-593.

Введение

В современных информационных системах для обеспечения унифицированного представления знаний предметной области в виде совокупности связанных между собой объектов, процессов и явлений применяют онтологии. В качестве хранилищ онтологий, как правило, выступают RDF-репозитории, представляющие онтологии в виде набора RDF-триплетов «субъект-свойство-объект». В качестве языка запросов к RDF-репозиториям в этом случае используется язык SPARQL, позволяющий описывать запросы с помощью комбинаций шаблонов триплетов.

Распространённой проблемой применения RDF-хранилищ в информационных системах является замедление выполнения SPARQL-запросов при достижении большого (более 10 миллионов триплетов) объема хранимых данных. Такое снижение производительности связано с необходимостью перебирать большое количество триплетов при выполнении запроса, содержащего несколько конъюнкций и/или дизъюнкций шаблонов. По этой причине, как правило, выполняют предварительную группировку триплетов в таблицы (*cluster-property table*), соответствующие:

- одному свойству (предикату);
- свойству (предикату) для сущностей одного класса;
- комбинациям субъектов, предикатов и объектов, которые часто встречаются в большинстве запросов.

Наряду с этим, при обработке запроса задействуется оптимизатор, выбирающий наиболее дешёвый план выполнения запроса на основе функции оценки стоимости, которая учитывает количество промежуточных результатов, порождаемых связкой шаблонов триплетов в том или ином плане выполнения.

Для повышения скорости выполнения SPARQL-запросов к онтологиям был разработан способ компактного представления онтологии в виде совокупности специализированных матрицеподобных структур (*C*-систем), а также основанный на этом представлении метод обработки SPARQL-запросов, реализованный в рамках парадигмы программирования в ограничениях [1]. При работе с онтологиями большого размера компактное представление онтологии и применение авторских методов распространения нечисловых ограничений позволило существенно ускорить выполнение SPARQL-запросов по сравнению с традиционным подходом к их обработке, базирующимся на динамическом программировании. Предложенный метод был успешно апробирован на онтологии интегрированного пространства знаний (ИПЗ) [2]. Однако он охватывал только этап непосредственного исполнения запроса, и не затрагивал этап его априорного (до стадии исполнения) анализа.

Особенностью представленных в статье исследований является то, что априорный анализ SPARQL-запросов предлагается выполнять для онтологий, разработанных с применением онтологических паттернов содержания (*Content ontology Design Pattern*, CDP) [3]. Паттерны представляют собой небольшие целостные фрагменты онтологии, формализующие обобщенные ситуации предметной области (например, участие в событии, исполнении роли, наличие частей у объекта и др.). Их использование позволяет обеспечить качество разрабатываемой онтологии, так как каждый паттерн является проверенным решением, доказавшим свою эффективность. По мнению авторов, помимо упрощения и ускорения процесса разработки онтологии, применение паттернов создает предпосылки для повышения эффективности запросов к онтологиям. Каждому онтологическому паттерну ставится в соответствие набор квалификационных вопросов (*Competency questions*) на естественном языке, которые указывают, какую информацию можно получить с помощью представляемого им фрагмента онтологии. При формировании списка квалификационных вопросов предполагается, что именно вопросы из этого списка наиболее часто адресуются информационной системе. Другими словами, квалификационные вопросы являются наиболее частотными, типовыми. Они могут быть легко преобразованы в соответствующие шаблоны SPARQL-запросов, содержащие неизвестные целевые атрибуты и атрибуты, значения которых присваиваются при конкретизации запроса пользователем и обеспечивают фильтрацию данных. Таким образом, структура подобных шаблонов SPARQL-запросов, описывающая используемые связи и концепты онтологии, известны заранее, а конкретные константы (экземпляры концептов онтологии) подставляются в шаблон в качестве условий фильтрации конечным пользователем. Чтобы анализ структуры запроса, построенного на основе такого шаблона, не производился при

каждом исполнении запроса целесообразно априорно единожды выполнить процедуру проверки и упрощения шаблона запроса и многократно использовать результаты анализа для повышения эффективности исполнения конкретизированных запросов данного типа.

Оригинальность предлагаемых исследований состоит в том, что задачу априорного анализа семантической корректности шаблона запроса, а также задачу его упрощения предлагается решать в форме задач удовлетворения ограничений.

Представленный в статье подход основан на интеграции структурных алгоритмов теории удовлетворения ограничений с предлагаемыми алгоритмами распространения нечисловых ограничений и нацелен на априорный анализ структуры запроса, её упрощение, а также на раннее выявление некорректных запросов и подготовку структур данных для эффективного выполнения конкретизированных запросов.

1 Методы решения задач удовлетворения ограничений

Рассмотрим общую постановку задачи удовлетворения ограничений и подходы к ее решению. Согласно [4], задача удовлетворения ограничений (*Constraint Satisfaction Problem*, CSP) состоит из трёх компонент:

- X – множество переменных $\{X_1, X_2, \ldots, X_n\}$;
- D – множество доменов $\{D_1, D_2, \ldots, D_n\}$, где D_i является областью определения переменной X_i;
- C – множество ограничений $\{C_1, C_2, \ldots, C_m\}$, которые предписывают допустимые комбинации значений переменных.

Состояние задачи описывается как присваивание значений некоторым (частичное присваивание) или всем переменным (полное присваивание): $\{X_i = v_i, X_j = v_j, \ldots\}$. Решением задачи CSP является полное присваивание, которое удовлетворяет всем ограничениям.

Далее рассматриваются только задачи удовлетворения ограничений с конечными областями определения переменных.

Методы решения задач удовлетворения ограничений могут быть разбиты на *три класса* [5, 6]. Первый класс содержит различные варианты алгоритмов *поиска в глубину с возвратами*, которые строят решение путем расширения частичного присваивания шаг за шагом, используя различные эвристики и применяя разумные стратегии возврата из тупиковых вершин. Ко второму классу относятся *алгоритмы распространения ограничений*, которые исключают из пространства поиска некоторые элементы, не входящие в решение, обеспечивая снижение размерности задачи (см. например [7, 8]). Эти алгоритмы не строят сами по себе решение, поскольку исключают не все элементы, не входящие в решение. Они применяются или для препроцессинга задачи до использования алгоритмов другого типа, или перемежаются с шагами алгоритма другого типа (например, поиска с возвратами) для повышения производительности последнего. Наиболее популярными методами в рамках данного класса являются методы достижения совместности в вершинах (*Node consistency*) и по дугам (*Arc consistency*) [9, 10]. При этом полагается, что переменным соответствуют вершины некоторого графа, а бинарным ограничениям – его дуги. Точное определение понятия совместности довольно громоздко, поэтому здесь не приводится. Алгоритмы распространения ограничений позволяют свести исходную задачу CSP $<X, D, C>$ к более простой задаче $<X', D', C'>$, где каждая область переменных в D' является подмножеством соответствующей области в D, а ограничения C' содержат то же множество решений CSP, что и ограничения C, но вид ограничений C', как правило, существенно проще.

Наконец, *структурные алгоритмы* (см. например, [11, 12]) используют информацию о структуре первичного или двойственного графа ограничений задачи. Алгоритмы этого клас-

са производят декомпозицию исходной задачи CSP на слабо связанные подзадачи, которые могут быть решены с помощью методов предыдущих двух классов.

Рассмотрим основные подходы, которые лежат в основе структурных алгоритмов. Другими словами, опишем способы, позволяющие использовать для быстрого поиска решений структуру самой задачи, представленную в виде графа ограничений.

Структура или топология задач CSP может описываться с помощью различных графовых моделей: (первичного) *графа ограничений*, *гиперграфа ограничений*, *двойственного графа ограничений*.

Первичный граф ограничений CSP (V, D, C) – это неориентированный граф $G = (V, E)$, вершины V которого соответствуют переменным CSP, причём две вершины соединяются ребром в графе G, если соответствующие переменные участвуют в одном и том же ограничении.

Граф ограничений может быть использован для разбиения всей задачи на совокупность более простых с точки зрения вычислительной сложности подзадач.

Если в составе задачи CSP можно выделить полностью независимые друг от друга подзадачи, то их можно решать отдельно друг от друга, а полученные решения впоследствии скомбинировать в решение исходной задачи.

Для разбиения задачи CSP на независимые подзадачи $\{CSP_i\}$, можно рассмотреть связные компоненты графа ограничений. Предположим, что каждая подзадача CSP_i содержит c переменных из общего количества n переменных, где c – константа. В таком случае получится n/c подзадач, и для решения каждой из них потребуется, самое большее, объем работы d^c, где d – размер доменов переменных. Таким образом, общий объём работы измеряется величиной $O(d^c \cdot n/c)$, которая линейно зависит от n; без такой декомпозиции общий объём работы измерялся бы величиной $O(d^n)$, которая экспоненциально зависит от n. Поэтому полностью независимые подзадачи являются очень привлекательными. Однако они встречаются редко.

В большинстве случаев подзадачи любой задачи CSP связаны друг с другом по переменным. В простейшем случае граф ограничений является деревом: любые две переменные связаны не больше чем одним путём. Алгоритм решения CSP с древовидной структурой обладающий низкой вычислительной сложностью (задача CSP может быть решена за линейное время [13]), включает следующие этапы.

1) выбрать в качестве корня дерева любую переменную и упорядочить переменные от корня к листьям таким образом, чтобы родительская вершина каждой вершины в дереве предшествовала этой вершине в таком упорядочении. Обозначить эти переменные по порядку как $X_1,\ldots, X_n$. Теперь каждая переменная, кроме корня, имеет только одну родительскую переменную.
2) в цикле по j от n до 2 применять проверку совместности по дугам (X_i, X_j), где X_i – родительская вершина вершины X_j, удаляя значения из области определения D_i по мере необходимости.
3) в цикле по j от 1 до n присваивать X_j любое значение, совместное со значением, присвоенным X_i, где X_i – родительская вершина вершины X_j.

Поскольку существует эффективный алгоритм для деревьев, следует рассмотреть вопрос о том, можно ли каким-то образом приводить к древовидным структурам более общие графы ограничений. Существуют два основных способа решения этой задачи; один из них основан на удалении вершин, а другой – на слиянии вершин друг с другом и образовании супервершин. Первый подход предусматривает присваивание значений некоторым переменным так, чтобы оставшиеся переменные образовывали дерево. В рамках первого подхода наиболее известными методами являются *метод множеств разрыва цикла* [14] и *метод нахождения связных компонент графа ограничений* [15, 16].

К типичным методам, реализованным в рамках второго подхода, относятся: метод древовидной декомпозиции (*tree decomposition*) [15, 17, 18] и метод древовидной кластеризации (*tree-clustering*) [17].

1.1 Метод множеств разрыва цикла

Метод множеств разрыва цикла (*cycle cutset*) [14] основан на том, что присваивание значения переменной устраняет её из дальнейшего рассмотрения в этой ветви дерева поиска. Это сильно меняет связность оставшейся части графа ограничений. Общий алгоритм решения указанным способом описан ниже [4].

1) выбрать подмножество *S* из множества переменных задачи CSP, такое, что граф ограничений после удаления *S* становится деревом. Подмножество *S* называется множеством разрыва цикла.
2) для каждого возможного присваивания переменным в *S*, которое удовлетворяет всем ограничениям в *S*, выполнить следующее:

- удалить из областей определения оставшихся переменных любые значения, несовместные с данным присваиванием для S;
- если оставшаяся задача CSP имеет решение, вернуть это решение вместе с присваиванием для *S*.

Если множество разрыва цикла имеет размер *c*, то общее время работы алгоритма составляет $O(d^{c} \cdot (n-c) d^{2})$. Несмотря на то, что задача поиска наименьшего множества разрыва цикла является NP-трудной, известны эффективные приближенные алгоритмы её решения.

1.2 Древовидная декомпозиция

Любая древовидная декомпозиция должна удовлетворять следующим требованиям:

- каждая переменная из первоначальной задачи должна появляться, по меньшей мере, в одной из подзадач;
- если две переменные первоначальной задачи связаны ограничением, то они должны появляться вместе (наряду с этим ограничением), по меньшей мере, в одной из подзадач;
- если какая-то переменная появляется в двух подзадачах в дереве, то должна появляться в каждой подзадаче вдоль пути, соединяющего эти подзадачи.

Ширина дерева древовидной декомпозиции графа на единицу меньше размера наибольшей подзадачи; ширина дерева самого графа определяется как минимальная ширина дерева среди всех его древовидных декомпозиций.

Если граф имеет ширину дерева *w* и дана соответствующая древовидная декомпозиция, то соответствующая задача может быть решена за время $O(nd^{w+1})$. Это означает, что задачи CSP с графами ограничений, характеризующимися конечной шириной дерева, могут быть решены за полиномиальное время. Задача поиска декомпозиции с минимальной шириной дерева является NP-трудной, но существуют эвристические методы, которые хорошо работают на практике.

Описанные выше методы структурной декомпозиции имеют следующие общие черты. Каждая подзадача общей задачи CSP решается независимо; если какая-либо из них не имеет решения, то вся задача также не имеет решения. Если удается решить все подзадачи, то предпринимается попытка составить глобальное решение.

2 Компактное представление онтологии с помощью нечисловых матриц и вывод на онтологии как распространение нечисловых ограничений

Любую онтологию можно рассматривать как совокупность отношений, что даёт возможность ставить и решать задачи вывода на онтологиях как задачи удовлетворения ограничений. Однако табличное представление отношений не позволяет обеспечить достаточную эффективность процедур вывода. Поэтому разработка способов «сжатого» представления многоместных отношений представляется актуальной проблемой.

Многоместные отношения могут быть выражены более компактно, чем полным перечислением своих кортежей. Любое отношение может быть представлено как объединение декартовых произведений некоторых множеств. Эта мысль наглядно проиллюстрирована на рисунке 1, где слева показано отношение, записанное в виде обычной таблицы, а справа – «сжатая» форма записи многоместного отношения в виде специализированной матрицы.

Здесь области определения переменных: X – $\{a, b, c, d\}$, Y – $\{1, 2, 3, 4, 5\}$.

Фактически при матричной записи на рисунке 1b между компонентами одной строки опускается знак операции × (декартово произведение), а между строками явно не записывается знак операции ∪ (объединение множеств).

X	Y
c	1
c	2
c	4
c	5
b	2
b	4
d	1
d	5

(a)

X	Y
$\{c\}$	$\{1, 2, 4, 5\}$
$\{b\}$	$\{2, 4\}$
$\{d\}$	$\{1, 5\}$

(b)

Рисунок 1 – Табличное представление ограничения (a);
представление ограничения в виде специализированной матрицы (b)

Предлагаемый математический аппарат для «сжатого» представления многоместных отношений используют два типа матрицеподобных структур [19, 20]. Первый тип – это C-системы. C-система – это нечисловая матрица, где в качестве ячеек выступают не отдельные элементы, а множества.

Распишем C-систему, представленную на рисунке 1b в явном виде, то есть в форме алгебраического выражения над множествами:

$$T[XY] = \begin{bmatrix} \{c\} & \{1,2,4,5\} \\ \{b\} & \{2,4\} \\ \{d\} & \{1,5\} \end{bmatrix} = \{c\} \times \{1,2,4,5\} \cup \{b\} \times \{2,4\} \cup \{d\} \times \{1,5\}.$$

Графически каждый C-кортеж соответствует некоторой области в признаковом пространстве (декартову произведению), а вся C-система – объединению этих областей.

Другой тип матрицеподобных структур, обеспечивающий компактное представление многоместных отношений, – это D-система. Данная матрица записывается в обратных скобках. Ниже приводится D-система $P[XY]$, которая эквивалентна рассмотренной ранее C-системе $T[XY]$, поскольку обе эти структуры описывают одну и ту же таблицу, представленную на рисунке 1a:

$$P[XY] = \begin{bmatrix} \{c,d\} & \{2,4\} \\ \{b,c\} & \{1,5\} \\ \varnothing & \{1,2,4,5\} \end{bmatrix} = (\{c,d\} \times \{1,2,3,4,5\} \cup \{a,b,c,d\} \times \{2,4\}) \cap$$

$$\cap (\{b,c\} \times \{1,2,3,4,5\} \cup \{a,b,c,d\} \times \{1,5\}) \cap (\{a,b,c,d\} \times \{1,2,4,5\}).$$

Пустая компонента «∅» – это фиктивная компонента, не содержащая значений.

D-система расписывается как сложное алгебраическое выражение. Каждая строка *D*-системы описывает область в признаковом пространстве. Эта область имеет более сложную структуру, чем декартово произведение (объединение декартовых произведений). Вся *D*-система есть пересечение областей, соответствующих отдельным строкам.

Напомним, что эффективный алгоритм решения задачи CSP для случая, когда первичный граф ограничений представляет собой дерево, опирается на такие методы распространения ограничений, как достижение совместности в вершинах (*Node consistency*) и по дугам (*Arc consistency*) [9, 10]. Использование специализированных матриц для представления качественных зависимостей привело к необходимости модифицировать алгоритмы обеспечения совместности. Уточним особенности процесса редуцирования пространства поиска (распространения ограничений) на основе матричного представления ограничений. Для этого приведём утверждения, позволяющие реализовывать эквивалентные преобразования совокупности ограничений для случая, когда ограничения представлены в виде набора *C*-систем [1, 21]. Целью преобразований является приведение CSP к более простому виду, где содержится меньшее количество *C*-систем, строк *C*-систем, столбцов (атрибутов) *C*-систем, значений в доменах атрибутов и т.п.

***Утверждение* 1** (У1). Если все строки (кортежи) *C*-системы пусты, то есть содержат хотя бы по одной пустой компоненте каждая, то *C*-система пуста (соответствующая задача CSP несовместна).

***Утверждение* 2** (У2). Если все компоненты некоторого атрибута (столбца *C*-системы) являются полными, то данный атрибут можно удалить из *C*-системы (удаляются все компоненты, стоящие в соответствующем столбце), а пара «удаляемый атрибут – его домен» сохраняется в векторе частичного решения.

***Утверждение* 3** (У3). Если домен некоторого атрибута *C*-системы содержит значения, не встречающиеся в соответствующем столбце, то эти значения удаляются из данного домена.

***Утверждение* 4** (У4). Если строка *C*-системы содержит хотя бы одну пустую компоненту (строка пуста), то строка удаляется.

***Утверждение* 5** (У5). Если компонента некоторого атрибута содержит значение, не принадлежащее соответствующему домену, то это значение удаляется из компоненты.

***Утверждение* 6** (У6). Если одна строка *C*-системы полностью доминирует (покомпонентно содержит) другую строку, то доминируемая строка удаляется из *C*-системы.

Аналогичные правила были сформулированы для случая, когда требуется распространить ограничения, выраженные в виде совокупности *D*-систем [22-24].

Часть из приведенных утверждений позволяет исключать значения из доменов и компонент атрибутов (У3, У5) или даже сами столбцы-атрибуты (У2), а часть дает возможность исключать из рассмотрения лишние строки (У4, У6).

Признак успешного завершения процесса поиска – элиминация из *C*-системы *всех* строк и столбцов без образования пустых строк. Другими словами, результирующее состояние в этом случае будет характеризоваться только совокупностью непустых усеченных доменов в векторе решения.

Признаком несовместности CSP является пустота *C*-системы (У1).

2.1 Модификации алгоритмов достижения вершинной и дуговой совместностей

Рассмотрим реализацию алгоритма *достижения вершинной совместности* для случая, когда CSP выражена в виде совокупности *C*-систем. Унарное ограничение моделируется *C*-системой, состоящей из одной строки, в которой имеется в точности одна нефиктивная компонента. Ввиду явного перечисления всех допустимых значений для указанного атрибута в нефиктивной компоненте *C*-системы, существенно упрощается реализация алгоритма *обеспечения вершинной совместности*: требуется просто пересечь два множества, одно из которых описывает старый домен атрибута, а второе соответствует нефиктивной компоненте. В результате пересечения образуется новый домен атрибута, откуда исключены значения, не входящие в единственную непустую компоненту. При этом само ограничение исключается из дальнейшего рассмотрения. Особенностью процедур распространения ограничений с использованием предлагаемых матрицеподобных структур является то, что остальные ограничения задачи CSP, записанные в виде *C*-систем, модифицируются, «настраиваясь» на новые домены переменных. В результате «настройки» из компонент *C*-системы могут вычеркиваться строки, столбы, значения из компонент, часть компонент может стать фиктивными. В процессе «настройки» могут образовываться новые унарные ограничения. Таким образом, процесс обеспечения вершинной совместности носит итеративный характер: «усечение доменов переменных» – «настройка *C*-систем на новые домены» – «усечение доменов переменных» и т.д.

Алгоритм обеспечения *совместности по дугам* для случая *C*-систем реализуется также довольно просто. Каждое бинарное отношение, моделируемое той или иной *C*-системой, на графе ограничений можно изобразить двумя дугами: дугой (*X, Y*) и дугой (*Y, X*), где *X, Y* – некоторые переменные (в нашей терминологии – атрибуты). По сути, дуги задают направление проверки бинарного отношения: в первом случае исключаются «лишние» значения из домена переменной *X* (домен переменной *X* обозначается D_X), а во втором – из домена переменной *Y* (домен переменной *Y* обозначается D_Y). Дугу (*X, Y*) можно сделать совместной, удаляя из D_X все значения, для которых не существует таких значений в D_Y, чтобы удовлетворялось соответствующее бинарное ограничение. В рамках предлагаемого подхода к обработке нечисловых ограничений, для обеспечения совместности одной дуги (*X, Y*) достаточно перечислить во вновь формируемом домене переменной *X* все значения, встречающиеся в компонентах того столбца *C*-системы, который соответствует атрибуту *X*. Данный алгоритм также носит итеративный характер, в процессе «настройки» на новые домены могут появляться новые унарные ограничения, а прежние бинарные ограничения упрощаются или вовсе исключаются из рассмотрения, обеспечивая сокращение пространства поиска.

2.2 Моделирование запросов к онтологиям

В качестве примера выберем один из паттернов онтологии ИПЗ [2], используемый для представления процессов исследований и соотнесенных с ними объектов. UML-схема данного паттерна приведена на рисунке 2. В соответствии с данным паттерном «Процесс исследования» (*Investigation*) содержит (отношение *participates-in*) персон (*Person*), реализующих роль «Исследователь» (*Researcher role*), и сущности, реализующие роли «Объект исследования» (*Investigation object role*), и «Предмет исследования» (*Investigation subject role*). Процесс исследования предполагает некоторое заключение (*Conclusion*), которое является «Информационным объектом» (*Information content entity*) о некоторой сущности предметной области (*Entity*). Для паттерна заданы следующие квалификационные вопросы: «Какой предмет в контексте данного объекта планировалось исследовать?», «Кто участвовал в исследованиях данной сущности?», «Какие результаты были получены в данном исследовании?» и др.

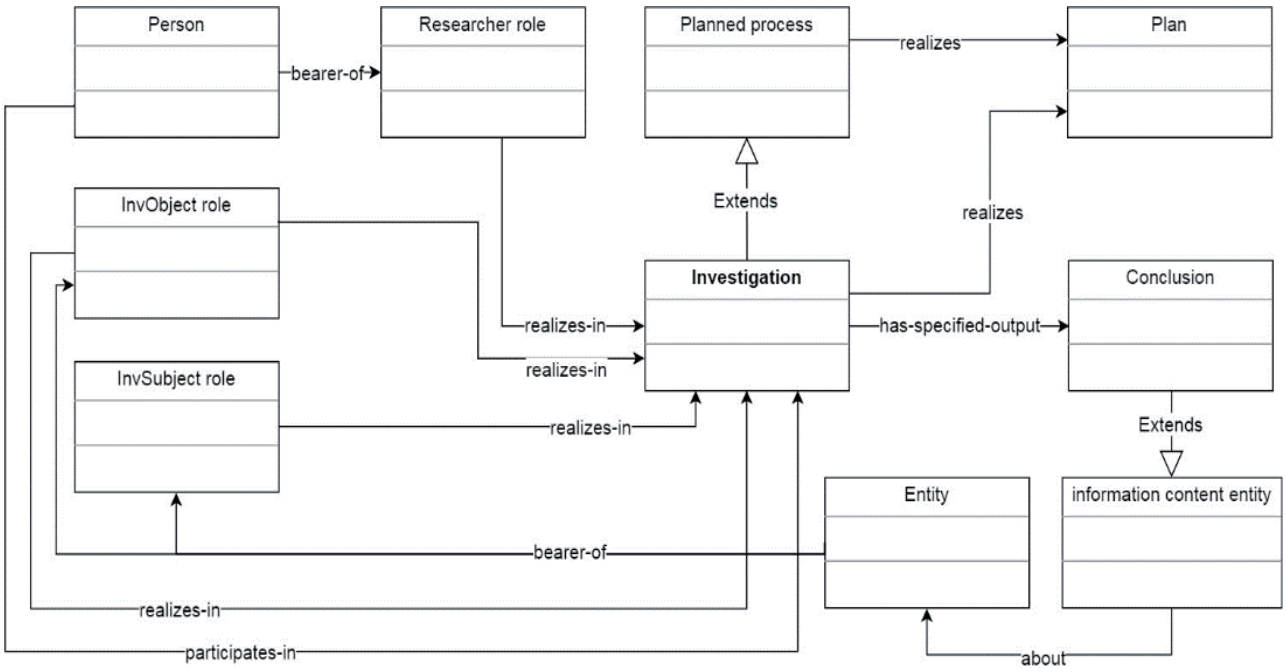

Рисунок 2 – UML-схема онтологического паттерна содержания «Исследование» (*Investigation*)

Рассмотрим применение предлагаемого подхода на примере анализа шаблона SPARQL-запроса, соответствующего одному из квалификационных вопросов – «Какие выводы и о каких предметах исследования сделал исследователь *X*?»:

```
PREFIX IKS: < http://www.iimm.ru/ontologies/oiks>
PREFIX INV: < http://www.iimm.ru/ontologies/cdp/investigation-process>
SELECT ?conclusion ?entity
WHERE {
X_person      IKS:bearer-of             ?role.
?role         rdf:type                  INV:Researcher-role.
?role         IKS:realizes-in           ?inv.
?entity       IKS:bearer-of             ?objRole.
?objRole      IKS:realizes-in           ?inv.
?inv          IKS:has-specified-output  ?conclusion.
?conclusion   IKS:about                 ?entity.
}.
```

Рассмотрим его выполнение на конкретном наборе данных (см. рисунок 3).

Для наглядности во фрагменте присутствуют только те классы и отношения, которые представлены в рассматриваемом SPARQL-запросе.

Для классов, экземпляров классов, и связей фрагмента онтологии, показанного на рисунке 3, введём специальные обозначения. Каждому классу фрагмента онтологии сопоставим определённую переменную (атрибут), а каждому экземпляру класса определённое значение данной переменной так, как это показано в таблице 1. Тогда в пределах рассматриваемого фрагмента получим следующие соответствия «атрибут – домен атрибута»:

X – $\{a_1, a_2, a_3, a_4, a_5, a_6\}$, Y – $\{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$, Z - $\{c_1, c_2, c_3, c_4, c_5, c_6\}$,

W – $\{d_1, d_2, d_3, d_4, d_5, d_6\}$, L – $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$,

M - $\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}\}$.

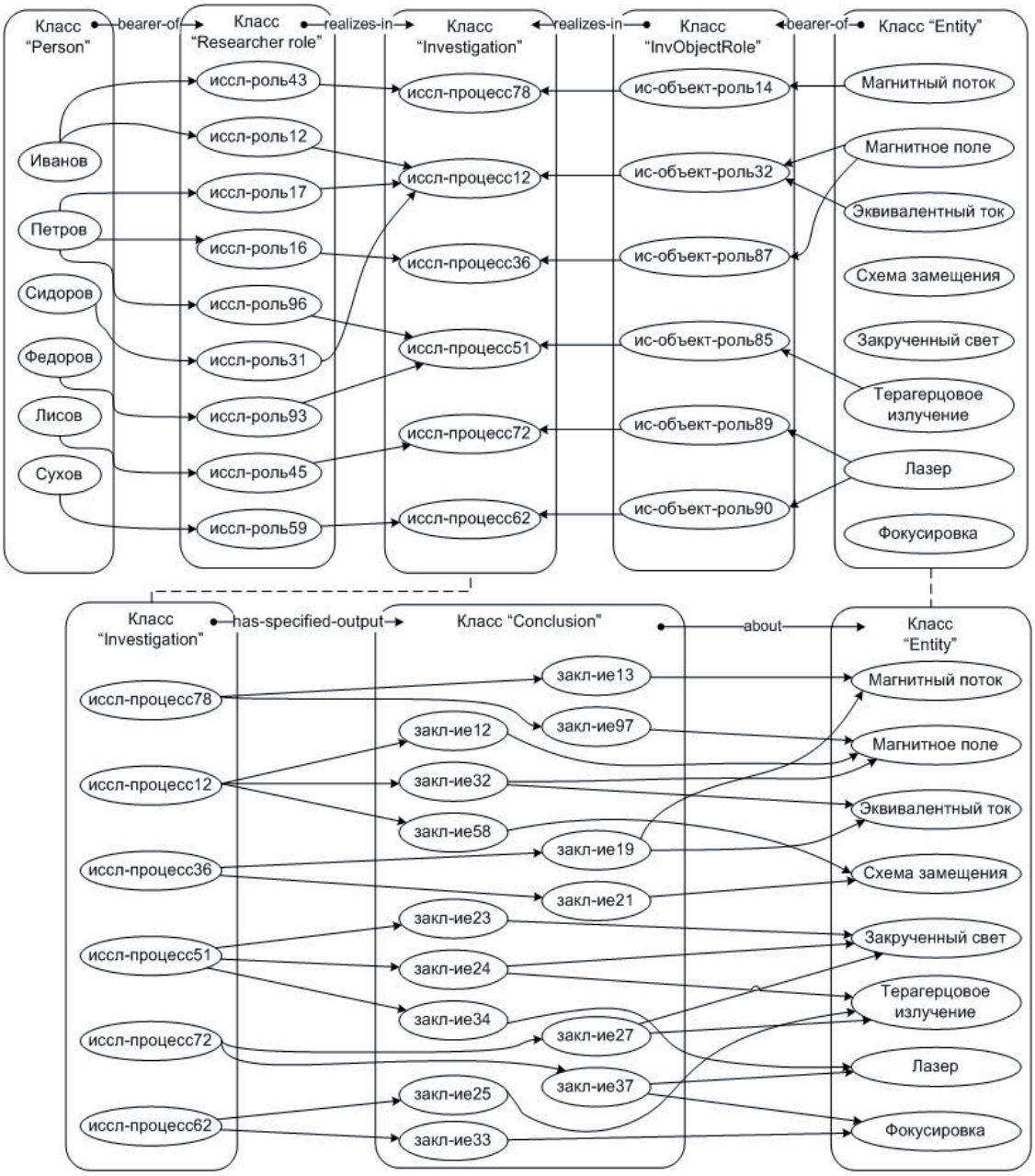

Рисунок 3 – Фрагмент данных в RDF-репозитории

Теперь каждой связи онтологии, которая фигурирует в рассматриваемом шаблоне SPARQL-запроса, сопоставим ограничение:

- $R_1[XY]$ – отношение «bearer-of» между классами X («PERSON») и Y («RESEARCHER_ROLE»);
- $R_2[YZ]$ – отношение «realizes-in» между классами Y («RESEARCHER_ROLE») и Z («INVESTIGATION»);
- $R_3[WZ]$ – отношение «realizes-in» между классами W («INV_OBJ_ROLE») и Z («INVESTIGATION»);
- $R_4[LW]$ – отношение «bearer-of» между классами L («ENTITY») и W («INV_OBJ_ROLE»);
- $R_5[ZM]$ – отношение «has-specified-output» между классами Z («INVESTIGATION») и M («CONCLUSION»);
- $R_6[ML]$ – отношение «about» между классами M («CONCLUSION») и L («ENTITY»).

Таблица 1 - Обозначение классов и экземпляров классов

Названия классов и экземпляров классов	Обозначение классов	Обозначение значений переменных
КЛАСС «PERSON»	X	
«Иванов»		a_1
«Петров»		a_2
«Сидоров»		a_3
«Федоров»		a_4
«Лисов»		a_5
«Сухов»		a_6
КЛАСС «RESEARCHER_ROLE»	Y	
«иссл-роль43»		b_1
«иссл-роль12»		b_2
«иссл-роль17»		b_3
«иссл-роль16»		b_4
«иссл-роль96»		b_5
«иссл-роль31»		b_6
«иссл-роль93»		b_7
«иссл-роль45»		b_8
«иссл-роль59»		b_9
КЛАСС «INVESTIGATION»	Z	
«иссл-процесс78»		c_1
«иссл-процесс12»		c_2
«иссл-процесс36»		c_3
«иссл-процесс51»		c_4
«иссл-процесс72»		c_5
«иссл-процесс62»		c_6
КЛАСС «INV_OBJ_ROLE»	W	
«исс-объ-роль14»		d_1
«исс-объ-роль32»		d_2
«исс-объ-роль87»		d_3
«исс-объ-роль85»		d_4
«исс-объ-роль89»		d_5
«исс-объ-роль90»		d_6
КЛАСС «ENTITY»	L	
«Магнитный поток»		e_1
«Магнитное поле»		e_2
«Эквивалентный ток»		e_3
«Схема замещения»		e_4
«Закрученный свет»		e_5
«Терагерцовое излучение»		e_6
«Лазер»		e_7
«Фокусировка»		e_8
КЛАСС «CONCLUSION»	M	
«закл-ие13»		f_1
«закл-ие97»		f_2
«закл-ие12»		f_3
«закл-ие32»		f_4
«закл-ие58»		f_5
«закл-ие19»		f_6
«закл-ие21»		f_7
«закл-ие23»		f_8
«закл-ие24»		f_9
«закл-ие34»		f_{10}
«закл-ие27»		f_{11}
«закл-ие37»		f_{12}
«закл-ие25»		f_{13}
«закл-ие33»		f_{14}

На рисунке 3 перечисленные отношения R_1–R_6 обозначены стрелками. Учитывая конкретные экземпляры классов, можно записать перечисленные выше отношения в виде предлагаемых матриц ограничений, а именно в виде *C*-систем:

- Отношение $R_1[XY]$

$$\begin{bmatrix} \{a_1\} & \{b_1, b_2\} \\ \{a_2\} & \{b_3, b_4, b_5\} \\ \{a_3\} & \{b_6\} \\ \{a_4\} & \{b_7\} \\ \{a_5\} & \{b_8\} \\ \{a_6\} & \{b_9\} \end{bmatrix}$$

- Отношение $R_2[YZ]$

$$\begin{bmatrix} \{b_1\} & \{c_1\} \\ \{b_2, b_3, b_6\} & \{c_2\} \\ \{b_4\} & \{c_3\} \\ \{b_5, b_7\} & \{c_4\} \\ \{b_8\} & \{c_5\} \\ \{b_9\} & \{c_6\} \end{bmatrix}$$

- Отношение $R_3[WZ]$

$$\begin{bmatrix} \{d_1\} & \{c_1\} \\ \{d_2\} & \{c_2\} \\ \{d_3\} & \{c_3\} \\ \{d_4\} & \{c_4\} \\ \{d_5\} & \{c_5\} \\ \{d_6\} & \{c_6\} \end{bmatrix}$$

- Отношение $R_4[LW]$

$$\begin{bmatrix} \{e_1\} & \{d_1\} \\ \{e_2\} & \{d_2, d_3\} \\ \{e_3\} & \{d_2\} \\ \{e_6\} & \{d_4\} \\ \{e_7\} & \{d_5, d_6\} \end{bmatrix}$$

- Отношение $R_5[ZM]$

$$\begin{bmatrix} \{c_1\} & \{f_1, f_2\} \\ \{c_2\} & \{f_3, f_4, f_5\} \\ \{c_3\} & \{f_6, f_7\} \\ \{c_4\} & \{f_8, f_9, f_{10}\} \\ \{c_5\} & \{f_{11}, f_{12}\} \\ \{c_6\} & \{f_{13}, f_{14}\} \end{bmatrix}$$

- Отношение $R_6[ML]$

$$\begin{bmatrix} \{f_1, f_6\} & \{e_1\} \\ \{f_2, f_3, f_4\} & \{e_2\} \\ \{f_4, f_6\} & \{e_3\} \\ \{f_5, f_7\} & \{e_4\} \\ \{f_8, f_9, f_{11}\} & \{e_5\} \\ \{f_9, f_{11}, f_{14}\} & \{e_6\} \\ \{f_{10}, f_{13}\} & \{e_7\} \\ \{f_{12}, f_{14}\} & \{e_8\} \end{bmatrix}$$

На примере матрицы R_1 поясним, что обозначает подобная запись. Первая строка данной матрицы соответствует двум стрелкам (дугам) на рисунке 3, проведённым от экземпляра «Иванов» (значение a_1) класса «PERSON» (переменная *X*) к экземплярам «иссл-роль43» (значение b_1) и «иссл-роль12» (значение b_2) класса «RESEARCHER_ROLE» (переменная *Y*). По аналогии, каждой строке всех перечисленных *C*-систем соответствует некоторое множество дуг между экземплярами соответствующих классов онтологии, изображённых на рисунке 3.

Реализация рассматриваемого шаблона запроса с помощью матриц ограничений будет сводиться к вычислению соединения бинарных отношений:

$$R_1[XY] \bowtie R_2[YZ] \bowtie R_3[WZ] \bowtie R_4[LW] \bowtie R_5[ZM] \bowtie R_6[ML].$$

В настоящей работе задачу соединения нескольких отношений, выраженных в виде совокупности нечисловых матриц, предлагается рассматривать как задачу удовлетворения ограничений и решать её с помощью упомянутых авторских методов распространения нечисловых ограничений.

3 Структурная декомпозиция и априорный анализ запросов

Эффективность и целесообразность применения различных методов ускорения обработки запросов, во многом, определяется наличием информации о возможных типах запросов, а также задачами, для решения которых используется хранилище онтологии. Наличие априорной информации о запросах к онтологии обуславливается методикой её разработки, включающей, как правило, этап тестирования возможности получения ответов на контрольные вопросы. Формирование контрольных вопросов осуществляется с учетом будущих задач, на решение которых направлена онтология. Использование онтологических паттернов содер-

жания при разработке онтологий гарантирует наличие контрольных (квалификационных) вопросов. Набор квалификационных вопросов является обязательным атрибутом паттерна, поскольку позволяет точно указать разработчику онтологии, какую информацию о каких сущностях и их взаимосвязях можно будет получить. Тем самым, обеспечивается возможность подбора того паттерна, который потенциально будет позволять отвечать на большинство пользовательских запросов.

Наличие набора наиболее ожидаемых вопросов позволяет проводить априорный анализ структуры соответствующих SPARQL-запросов к онтологии. Однако это не ограничивает пользователя в отношении формулировок запросов иного вида, в частности, пользовательский запрос может соответствовать комбинации квалификационных. Также возможна ситуация, когда пользовательский запрос не соотносится ни с одним из квалификационных вопросов или их комбинацией.

Особенности методов априорного анализа запросов также сильно зависят от характера использования хранилища онтологий. Существует несколько вариантов использования хранилищ онтологий. Одним из вариантов является хранилище для транзакционных систем (*Online Transaction Processing,* OLTP). Заметим, что на данный момент более эффективными и распространёнными хранилищами для OLTP-систем признаются реляционные базы данных, а не RDF-репозитории. Поэтому данный вариант подробно в работе не рассматривается. Еще одним вариантом является применение онтологий и RDF-репозиториев для управления нормативно-справочными данными (*Reference Data Management,* RDM), управления мастер-данными (*Master Data Management,* MDM) и/или организации систем интерактивной аналитической обработки (*Online Analytical Processing,* OLAP). В этих случаях частота изменения данных довольно невысока.

Таким образом, среди всего множества вариантов можно выделить два крайних случая. Первый вариант предполагает ориентацию хранилища на оперативную модификацию информации и обработку *ad hoc* - запросов. Второй вариант подразумевает низкую частоту изменений данных и предопределённый набор квалификационных вопросов, соответственно изначально заданный перечень шаблонов SPARQL-запросов.

И в первом, и во втором случае априорный анализ запроса способен существенно ускорить процесс его исполнения. В первом случае имеется возможность проанализировать и упростить структуру конкретного запроса, выявить некорректности в условиях фильтрации запроса. Второй вариант позволяет выполнить более глубокий априорный анализ. Каждый квалификационный вопрос транслируется в соответствующий шаблон SPARQL-запроса. Единожды выполнив анализ и упрощение шаблона запроса, появляется возможность многократно использовать результаты предварительных вычислений для повышения эффективности обработки однотипных конкретизированных запросов. Тот факт, что данные практически не изменяются, позволяет для каждого шаблона запроса заранее подготовить структуры данных с целью ускорения выполнения конкретизированных запросов.

Итак, можно выделить следующие основные этапы априорного анализа запросов:

1) Применение алгоритмов структурной декомпозиции для разбиения шаблона запроса (или конкретного запроса) на части. Выполняется при анализе как шаблонов запросов, так и при анализе конкретных запросов.
2) Анализ совместности условий фильтрации шаблона запроса (или конкретного запроса). Данный этап актуален, прежде всего, при анализе конкретных запросов.
3) Подготовка структур данных для быстрого выполнения конкретных запросов, описываемых шаблоном. Данный этап присутствует только при анализе шаблонов запросов.

Дальнейшее объяснение предлагаемого подхода к априорному анализу запросов к онтологиям выполнено на примере рассмотренного выше шаблона SPARQL-запроса, который

соответствует квалификационному вопросу «Какие выводы и о каких предметах исследования сделал исследователь *X*?» и ситуации, когда редко осуществляется модификация данных в репозитории.

Представим данный шаблон SPARQL-запроса в виде графа ограничений (рисунок 4).

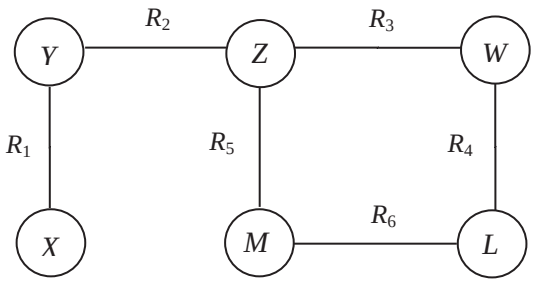

Рисунок 4 – Граф задачи CSP, представляющий шаблон запроса

3.1 Этап структурной декомпозиции

Применение описанных ранее алгоритмов структурной декомпозиции позволяет разбить шаблон запроса/запрос на слабо связанные части шаблона/подзапросы. На последующих этапах это позволяет независимо вычислять результаты подзадач, а затем выполнять их соединение для получения ответа на исходную задачу. На рисунке 5 приведен пример структурной декомпозиции графа задачи CSP, которая описывает шаблон запроса (рисунок 4). Разбиение исходной задачи CSP на подзадачи выполняется путем присваивания значений переменной Z.

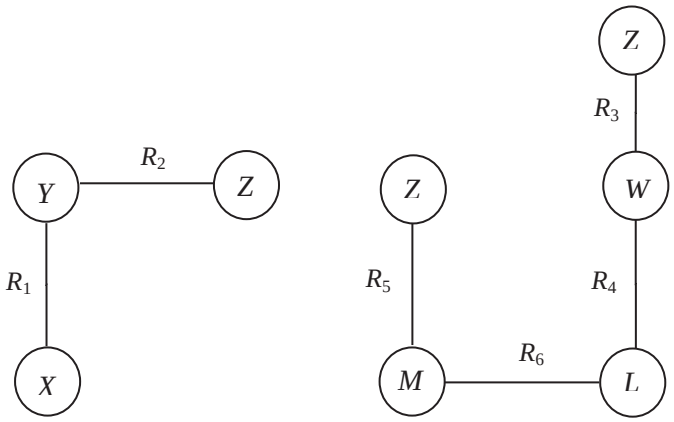

Рисунок 5 – Пример структурной декомпозиции шаблона запроса после присваивания значений переменной *Z*

В этом случае для решения исходной задачи требуется рассмотреть две слабо связанные подзадачи CSP:

1) R_3[WZ] ⋈ R_4[LW] ⋈ R_5[ZM] ⋈ R_6[ML];
2) R_1[XY] ⋈ R_2[YZ].

На рисунке 6 показан пример того, как можно осуществить разбиение ранее описанного шаблона запроса альтернативным способом с применением метода древовидной декомпозиции. В этом случае для решения исходной задачи требуется рассмотреть три слабо связанные подзадачи CSP:

1) R_1[XY] ⋈ R_2[YZ];
2) R_3[WZ] ⋈ R_4[LW];
3) R_5[ZM] ⋈ R_6[ML].

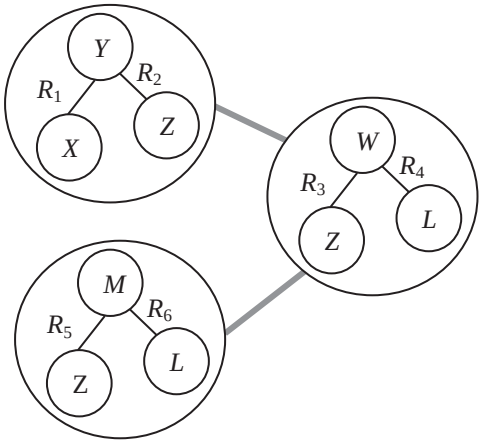

Рисунок 6 – Пример разбиения шаблона запроса на части методом древовидной декомпозиции

3.2 Этап подготовки структур данных для быстрого выполнения конкретизированных запросов

Поскольку наиболее трудоемкой операцией в ходе исполнения SPARQL-запросов к RDF-хранилищам является соединение триплетов, то для шаблонов запросов, в ряде случаев, имеет смысл вычислять подобные соединения заранее, получая одно многоместное отношение (ограничение). Для каждой подзадачи CSP, полученной в процессе структурной декомпозиции шаблона запроса, можно заранее вычислить соответствующее многоместное отношение. Затем, эти отношения можно соединить в одно. Приведём описанные вычисления, взяв за основу декомпозицию шаблона, показанную на рисунке 5.

Рассмотрим первую из двух подзадачу CSP: $R_3[WZ] \bowtie R_4[LW] \bowtie R_5[ZM] \bowtie R_6[ML]$.

Продемонстрируем, как осуществляется упрощение данной задачи CSP на основе алгоритмов распространения нечисловых ограничений. Поскольку в рассматриваемой задаче CSP отсутствуют унарные ограничения, то проанализируем имеющиеся ограничения на совместность по дугам. Последовательно анализируя ограничения R_1–R_3, выясняем, что они совместны по дугам. Анализ ограничения R_4[LW] показывает, что условие совместности по дугам для него не выполняется: первый столбец данной матрицы не содержит значений e_4, e_5 и e_8 переменной L. Другими словами значения, e_4, e_5 и e_8 не имеют поддержки в R_4[LW], поэтому данные значения должны быть исключены из домена атрибута L. Согласно утверждению У3, домен атрибута L сужается до множества $\{e_1, e_2, e_3, e_6, e_7\}$.

После того, как ограничение R_4[LW] стало совместно по дугам, активируется ограничение R_6[ML], также содержащее в схеме атрибут L, домен которого был изменен. Теперь выполняется «настройка» ограничения R_6[ML] на новый домен переменной L. При этом, по правилу У5 из второго столбца матрицы R_6[ML] удаляются значения e_4, e_5 и e_8, образуя пустые компоненты, а затем, опираясь на У4, удаляются строки 4, 5 и 8, соответственно. В результате матрица R_6[ML] примет вид: отношение $R'_6[ML]$

$$\begin{matrix} M & L \end{matrix}$$
$$\begin{bmatrix} \{f_1, f_6\} & \{e_1\} \\ \{f_2, f_3, f_4\} & \{e_2\} \\ \{f_4, f_6\} & \{e_3\} \\ \{f_9, f_{11}, f_{14}\} & \{e_6\} \\ \{f_{10}, f_{13}\} & \{e_7\} \end{bmatrix}.$$

Данное ограничение несовместно по дугам, так как в первом столбце матрицы отсутствуют значения f_5, f_7, f_8, f_{12}, которые содержатся в домене переменной M. Согласно утверждению У3, домен атрибута M сужается до множества $\{f_1, f_2, f_3, f_4, f_6, f_9, f_{10}, f_{11}, f_{13}, f_{14}\}$.

Теперь активируется ограничение $R_5[ZM]$, в схеме которого присутствует затронутая изменениями переменная M. После «настройки» ограничения $R_5[ZM]$ на новый домен переменной M с использованием утверждения У4, имеем следующее упрощённое ограничение:

Отношение $R'_5[ZM]$

$$\begin{matrix} Z & M \end{matrix}$$
$$\begin{bmatrix} \{c_1\} & \{f_1, f_2\} \\ \{c_2\} & \{f_3, f_4\} \\ \{c_3\} & \{f_6\} \\ \{c_4\} & \{f_9, f_{10}\} \\ \{c_5\} & \{f_{11}\} \\ \{c_6\} & \{f_{13}, f_{14}\} \end{bmatrix}.$$

Процесс распространения ограничений останавливается. Имеем следующую упрощённую задачу CSP. Домены переменных: $X - \{a_1, a_2, a_3, a_4, a_5, a_6\}$, $Y - \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9\}$, $Z - \{c_1, c_2, c_3, c_4, c_5, c_6\}$, $W - \{d_1, d_2, d_3, d_4, d_5, d_6\}$, $L - \{e_1, e_2, e_3, e_6, e_7\}$, $M - \{f_1, f_2, f_3, f_4, f_6, f_9, f_{10}, f_{11}, f_{13}, f_{14}\}$.

Ограничения: матрицы $R_1 - R_4$ остаются без изменений, а матрицы R_5 и R_6 заменяются матрицами R'_5 и R'_6. Для упрощённой подзадачи CSP ($R_3[WZ] \bowtie R_4[LW] \bowtie R'_5[ZM] \bowtie R'_6[ML]$) получим дерево поиска, показанное на рисунке 7.

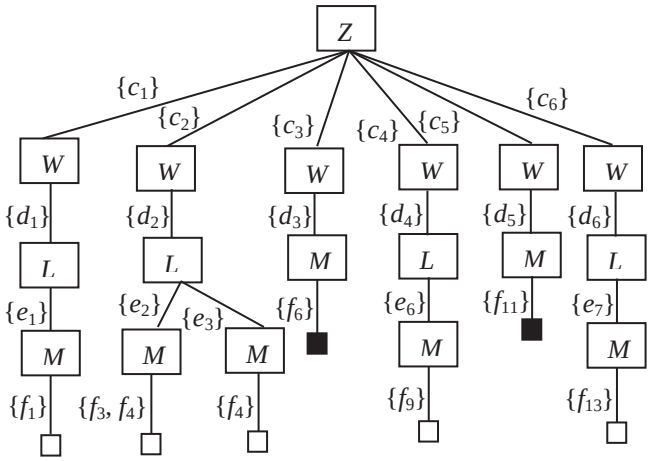

Рисунок 7 – Дерево поиска для упрощённой подзадачи CSP $R_3[WZ] \bowtie R_4[LW] \bowtie R'_5[ZM] \bowtie R'_6[ML]$

Узлы дерева соответствуют переменным, выбираемым на каждом шаге поиска, а дуги помечены различными значениями данных переменных. Каждая ветвь дерева поиска оканчивается либо незакрашенным белым квадратом (случай, когда решение получено), либо квадратом, окрашенным в черный цвет (случай, когда выявлено противоречие и требуется осуществлять возврат к одной из родительских вершин данного узла). Если в дереве поиска у родительского узла имеется только один дочерний, то это означает, что значение переменой, соотнесенной с дочерним узлом, было получено в результате процедур распространения ограничений без организации процедур ветвления.

На основе представленного дерева поиска может быть выписана С-система $R_8[ZWLM]$, описывающая все решения подзадачи CSP:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccc} Z & W & L & M \end{array} \\ \left[\begin{array}{cccc} \{c_1\} & \{d_1\} & \{e_1\} & \{f_1\} \\ \{c_2\} & \{d_2\} & \{e_2\} & \{f_3, f_4\} \\ \{c_2\} & \{d_2\} & \{e_3\} & \{f_4\} \\ \{c_4\} & \{d_4\} & \{e_6\} & \{f_9\} \\ \{c_6\} & \{d_6\} & \{e_7\} & \{f_{13}\} \end{array}\right] \end{array}.$$

Каждая строка *C*-системы соответствует некоторой нетупиковой ветви дерева поиска.

Для второй подзадачи CSP, а именно, для $R_1[XY] \bowtie R_2[YZ]$, оказывается, что она изначально совместна и в вершинах, и по дугам. Для данной задачи получим дерево поиска, показанное на рисунке 8.

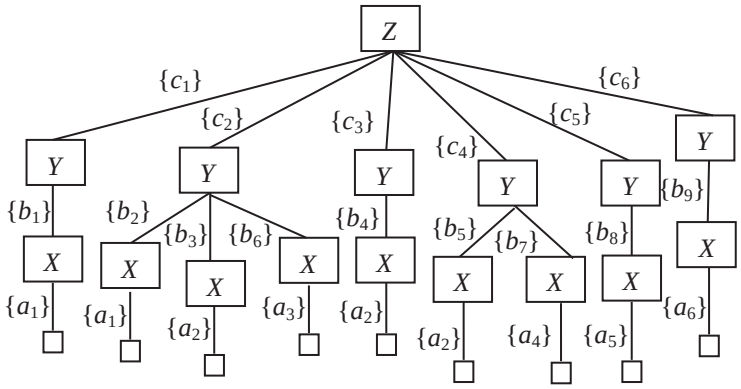

Рисунок 8 – Дерево поиска для подзадачи CSP $R_1[XY] \bowtie R_2[YZ]$

На основе представленного дерева поиска может быть выписана С-система $R_9[XYZ]$:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}\begin{array}{c} \begin{array}{ccc} X & Y & Z \end{array} \\ \left[\begin{array}{ccc} \{a_1\} & \{b_1\} & \{c_1\} \\ \{a_1\} & \{b_2\} & \{c_2\} \\ \{a_2\} & \{b_3\} & \{c_2\} \\ \{a_3\} & \{b_6\} & \{c_2\} \\ \{a_2\} & \{b_4\} & \{c_3\} \\ \{a_2\} & \{b_5\} & \{c_4\} \\ \{a_4\} & \{b_7\} & \{c_4\} \\ \{a_5\} & \{b_8\} & \{c_5\} \\ \{a_6\} & \{b_9\} & \{c_6\} \end{array}\right] \end{array}.$$

Совместно рассматривая ограничения $R_8[ZWLM]$ и $R_9[XYZ]$, можно упростить $R_9[XYZ]$, исключив из этой матрицы те значения переменной Z (соответственно и те строки матрицы), которые отсутствуют в $R_8[ZWLM]$. Получаем следующее отношение $R'_9[XYZ]$:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 7 \\ 9 \end{array}\begin{array}{c} \begin{array}{ccc} X & Y & Z \end{array} \\ \left[\begin{array}{ccc} \{a_1\} & \{b_1\} & \{c_1\} \\ \{a_1\} & \{b_2\} & \{c_2\} \\ \{a_2\} & \{b_3\} & \{c_2\} \\ \{a_3\} & \{b_6\} & \{c_2\} \\ \{a_2\} & \{b_5\} & \{c_4\} \\ \{a_4\} & \{b_7\} & \{c_4\} \\ \{a_6\} & \{b_9\} & \{c_6\} \end{array}\right] \end{array}.$$

Теперь вычислим $R_8[ZWLM] \bowtie R'_9[XYZ]$:

$$\begin{array}{c} \begin{array}{cccccc} X & Y & Z & W & L & M \end{array} \\ \left[\begin{array}{cccccc} \{a_1\} & \{b_1\} & \{c_1\} & \{d_1\} & \{e_1\} & \{f_1\} \\ \{a_1\} & \{b_2\} & \{c_2\} & \{d_2\} & \{e_2\} & \{f_3, f_4\} \\ \{a_1\} & \{b_2\} & \{c_2\} & \{d_2\} & \{e_3\} & \{f_4\} \\ \{a_2\} & \{b_3\} & \{c_2\} & \{d_2\} & \{e_2\} & \{f_3, f_4\} \\ \{a_2\} & \{b_3\} & \{c_2\} & \{d_2\} & \{e_3\} & \{f_4\} \\ \{a_3\} & \{b_6\} & \{c_2\} & \{d_2\} & \{e_2\} & \{f_3, f_4\} \\ \{a_3\} & \{b_6\} & \{c_2\} & \{d_2\} & \{e_3\} & \{f_4\} \\ \{a_2\} & \{b_5\} & \{c_4\} & \{d_4\} & \{e_6\} & \{f_9\} \\ \{a_4\} & \{b_7\} & \{c_4\} & \{d_4\} & \{e_6\} & \{f_9\} \\ \{a_6\} & \{b_9\} & \{c_6\} & \{d_6\} & \{e_7\} & \{f_{13}\} \end{array}\right] \end{array}.$$

Чтобы оценить сложность процедуры поиска решений для двух рассмотренных задач CSP, используем следующую меру – количество значений, участвующих в разметке дуг дерева поиска. Получаем следующие оценки: для первой подзадачи – 25. для второй подзадачи – 24 значения. Суммарная сложность – 49. Если бы исходная задача CSP, представленная на рисунке 4, решалась без применения методов структурной декомпозиции, то сложность решения была равна бы 57 (см. рисунок 9).

В данном случае выигрыш не столь существенен и составляет около 16%, но при решении практических задач большой размерности применение методов структурной декомпозиции позволяет решать те задачи, которые до этого были нереализуемы. К тому же разбиение исходной задачи на слабо связанные подзадачи обеспечивает возможность параллельного решения подзадач.

Для окончательного формирования отношения-заготовки, соответствующего разбираемому шаблону запроса, требуется вычислить проекцию отношения $R_8[ZWLM] \bowtie R'_9[XYZ]$ на атрибуты *X, M, L*. Первый атрибут представляет собой параметр поискового запроса, а два оставшихся – целевые атрибуты поискового запроса.

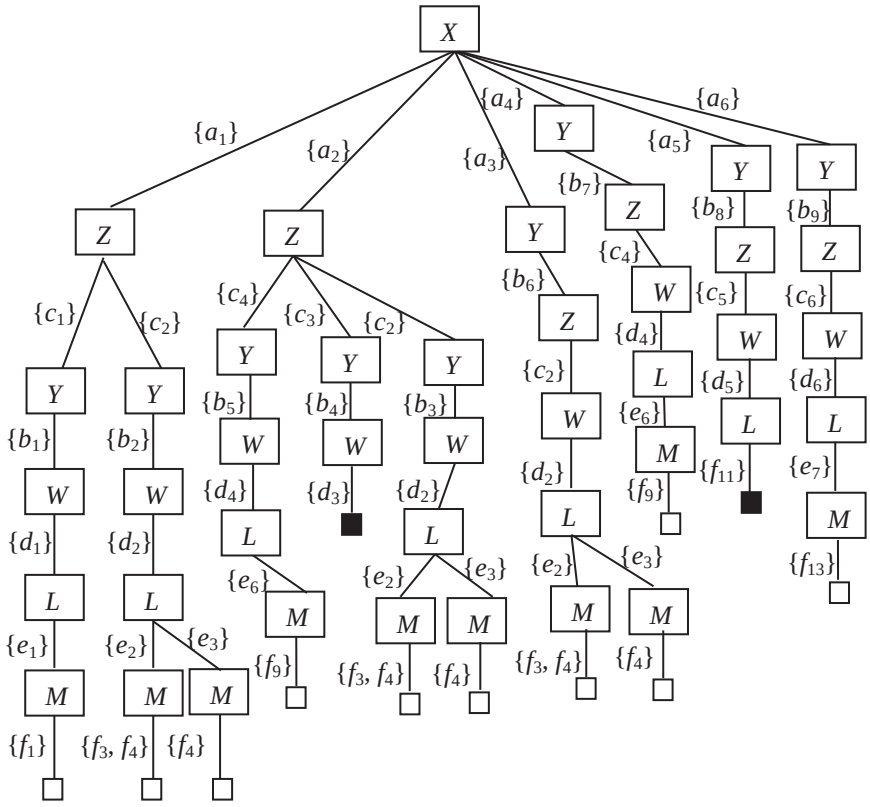

Рисунок 9 – Дерево поиска для исходной задачи CSP, решаемой без применения структурной декомпозиции

В нашем случае имеем следующее отношение-заготовку $Пр_{XYZ}(R_8[ZWLM] \bowtie R'_9[XYZ])$:

$$\begin{matrix} X & M & L \end{matrix}$$
$$\begin{bmatrix} \{a_1\} & \{f_1\} & \{e_1\} \\ \{a_1, a_2, a_3\} & \{f_3, f_4\} & \{e_2\} \\ \{a_1, a_2, a_3\} & \{f_4\} & \{e_3\} \\ \{a_2, a_4\} & \{f_9\} & \{e_6\} \\ \{a_6\} & \{f_{13}\} & \{e_7\} \end{bmatrix}.$$

При вычислении проекции было осуществлено «сжатие» результирующего отношения (алгоритмы «сжатия» здесь не приведены).

Анализируя полученное отношение-заготовку, нетрудно определить, что в ответ на конкретизированный пользовательский запрос «Какие выводы и о каких предметах исследования сделал исследователь Сухов?» (здесь значение параметра X равно a_6) будет получен результат: Сухов получил «закл-ие25» (значение параметра M равно f_{13}) о предмете исследования «Лазер» (значение параметра L равно e_7). Данному результату соответствует последняя строка отношения-заготовки. Процесс вычисления ответа уже не требует анализа и хранения всех шести отношений R_1–R_6, а сводится к анализу матрицы размерности 5×3.

Заключение

В статье онтология рассматриваться как совокупность отношений (унарных и бинарных), выраженных с помощью матрицеподобных структур – С-систем. Такое представление онто-

логии позволяет ставить и решать задачи вывода на онтологии как задачи CSP. В отличие от реляционных СУБД, ориентированных на поддержку ссылочной целостности интенсивно меняющихся данных фиксированного формата, RDF-репозитории не предполагают такой высокой скорости изменения их содержимого. Неформально говоря, базы данных предназначены для обработки данных, а системы онтологического инжиниринга – для работы со знаниями. При использовании онтологий нет необходимости оперировать такими атомарными структурами, как элементарный кортеж таблицы, а можно использовать более подходящие структуры для группирования и обобщения информации. «Компактное» представление отношений в виде совокупности С-систем обеспечивает снижение расходов памяти на хранение онтологии, а также способствует ускорению процедур вывода на онтологии.

Рассмотренная в работе задача априорного анализа и упрощения структуры SPARQL-запросов решается для онтологий, которые разработаны с применением онтологических паттернов содержания, что обеспечивает предсказуемость структуры потенциальных запросов. Фактически, каждому паттерну сопоставляется совокупность шаблонов SPARQL-запросов.

Предлагаемый в статье метод априорного анализа и преобразования шаблонов SPARQL-запросов основан на совместном применении методов структурной декомпозиции и методов удовлетворения нечисловых ограничений. Применение методов структурной декомпозиции даёт возможность разбивать шаблон SPARQL-запроса на части, распараллеливать выполнение подзадач, что особенно актуально при обращении к RDF-репозиториям большого объёма. Декомпозиция общей задачи CSP на подзадачи обычно выполняется таким образом, чтобы для получения решений каждой подзадачи CSP требовалось применить лишь алгоритмы распространения ограничений, характеризующиеся низкой вычислительной сложностью.

Ввиду низкой частоты модификаций RDF-репозитория, появляется возможность на основе анализа шаблонов подготавливать структуры данных для последующего исполнения конкретизированных запросов. Если в процессе такой подготовки для одной из подзадач (подшаблонов) выяснится, что в RDF-репозитории нет релевантной информации, то весь шаблон SPARQL-запроса признаётся некорректным. Если для всех подзадач подобная подготовка проходит успешно, то формируется результирующее «отношение-заготовка» – С-система, которая в дальнейшем используется при исполнении конкретных запросов. Это позволяет избежать излишних «накладных расходов», вызванных многократным выполнением операции соединения одних и тех же отношений при обработке пользовательских запросов, описываемых общим шаблоном.

Для соединения отношений, выраженных в виде совокупности *С*-систем, применяются авторские методы удовлетворения нечисловых ограничений. Они представляют собой модификацию известных методов достижения совместности в вершинах и по дугам на случай, когда качественные ограничения выражены в виде совокупности упомянутых матрицеподобных структур. По сравнению с традиционным подходом к вычислению соединений отношений при обработке SPARQL-запросов, базирующемся на динамическом программировании, предложенные методы обеспечивают существенное ускорение обработки отношений.

Предлагаемый подход к представлению онтологии, а также к организации процедур вывода на онтологиях, позволяет снизить расход памяти на хранение онтологии и обеспечить приемлемую скорость выполнения SPARQL-запросов даже для RDF-репозиториев, характеризующихся большими объёмами информации.

Благодарности

Работа выполнена при финансовой поддержке РФФИ (16-07-00377-а, 16-07-00562-а, 18-07-00615-а).

Список источников

[1] ***Зуенко, А.А.*** Применение методов распространения ограничений для ускорения обработки запросов к онтологиям / А.А. Зуенко, П.А. Ломов, А.Г. Олейник // Труды СПИИРАН. - 2017. - №1(50). - С.112–136.

[2] Ontology of Integrated Knowledge Space. - https://github.com/palandlom/ontology-of-integrated-knowledge-space.

[3] ***Blomqvist, E.*** Experimenting with eXtreme Design / E. Blomqvist, V. Presutti, E. Daga, A. Gangemi // In proceedings of EKAW 2010, LNCS 6317. Springer 2010. Berlin/Heidelberg/New York. - P. 120–134.

[4] ***Russel, S.*** Artificial Intelligence: A Modern Approach. 3rd edition / S. Russel, P. Norvig / Prentice Hall, 2010. - 1132 p.

[5] ***Bartak, R.*** Constraint Programming: In Pursuit of the Holy Grail / R. Bartak // Proceedings of the Week of Doctoral Students (WDS99), Part IV. – Prague: MatFyzPress, 1999. - P. 555–564.

[6] ***Ruttkay, Zs.*** Constraint satisfaction a survey / Zs. Ruttkay // CWI Quarterly. - 1998. - V. 11. - P.163–214.

[7] ***Barto, L.*** Constraint satisfaction problems solvable by local consistency methods / L. Barto, M. Kozik / Journal of the ACM. - 2014. - 61(1), article 3. - P.1–19.

[8] ***Cheng, K.C.*** An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints / K.C. Cheng, R.H. Yap // Constraints. - 2010. - 15 (2). - P.265–304.

[9] ***Mackworth, A.K.*** Consistency in networks of relations / A.K. Mackworth // Artificial Intelligence. - 1977. - 8(1)/ - P.99–118.

[10] ***Mackworth, A.K.*** Arc and path consistency revised / A.K. Mackworth, E.C. Freuder // Artificial Intelligence. - 1985. – 25. - P.65–74.

[11] ***Gottlob, G.*** A comparison of structural CSP decomposition methods / G. Gottlob, N. Leone, F. Scarcello // Artificial Intelligence. - 2000. - 124(2). - P.243–282.

[12] ***Katajainen J.*** Performance tuning an algorithm for compressing relational tables / J. Katajainen, J.N. Madsen // In Proceedings of the 8th Scandinavian Workshop on Algorithm Theory. Volume 2368 of Lecture Notes in Computer Science, Springer-Verlag, 2002. - P.398–407.

[13] ***Miguel, I.*** Solution techniques for constraint satisfaction problems: foundations / I. Miguel, Q. Shen // Artificial Intelligence Review. - 2001. - V. 15. - P.241–265.

[14] ***Dechter, R.*** Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition / R. Dechter // Artificial Intelligence. - 1990. - V. 41. - P.273–312.

[15] ***Dechter, R.*** Network-based heuristics for constraint satisfaction problems / R. Dechter, J. Pearl // Artificial Intelligence. - 1987. - 34(1). - P.1–38.

[16] ***Freuder, E.C.*** A sufficient condition for backtrack-free search / E.C. Freuder // Journal of the ACM. - 1982. - 29(1). - P.24–32.

[17] ***Dechter, R.*** Tree clustering for constraint networks / R. Dechter, J. Pearl // Artificial Intelligence. 1989. 38 (3). - P.353–366.

[18] ***Freuder, E.C.*** A sufficient condition for backtrack-bounded search / E.C. Freuder // Journal of the ACM. - 1985. 32(4). - P.755–761.

[19] ***Кулик, Б.А.*** Алгебраический подход к интеллектуальной обработке данных и знаний / Б.А. Кулик, А.А. Зуенко, А.Я. Фридман. - СПб.: Изд-во Политехн. ун-та, 2010. - 235 с.

[20] ***Zakrevskij, A.*** Integrated Model of Inductive-Deductive Inference Based on Finite Predicates and Implicative Regularities. / A. Zakrevskij // In: Naidenova, X., Ignatov, D. (eds.) Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems.- Hershey, PA: IGI Global, 2013. P.1-12.

[21] ***Зуенко, А.А.*** Качественное моделирование технических систем на основе методов распространения ограничений / А.А. Зуенко // Открытые семантические технологии проектирования интеллектуальных систем - Open Semantic Technologies for Intelligent Systems (OSTIS-2016): Материалы VI Международной научно-технической конф. (18-20 февраля 2016 г., Минск, Беларусь). – Минск: БГУИР, 2016.- С.573–578.

[22] ***Зуенко, А.А.*** Вывод на ограничениях с применением матричного представления конечных предикатов / А.А. Зуенко // Искусственный интеллект и принятие решений. - 2014. - № 3. - С.21–31.

[23] ***Зуенко, А.А.*** Методы ускорения комбинаторного поиска на основе матричного представления качественных зависимостей в задачах удовлетворения ограничений / А.А. Зуенко // Вестник Иркутского государственного технического университета. 2018. Т. 22. № 7.(138). С.69–87.

[24] ***Zuenko, A.A.*** Matrix-Like Structures in Solution of Constraint Satisfaction Problems / A.A. Zuenko // 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC) (18-25 Aug. 2018, Vladivostok, Russia). IEEE. - DOI: 10.1109/RPC.2018.8482223.

COMBINING OF METHODS OF CONSTRAINT PROPAGATION AND STRUCTURAL DECOMPOSITION FOR THE A PRIORI ANALYSIS OF QUERIES TO THE ONTOLOGIES

A.A. Zuenko[1], P.A. Lomov[2]

Institute for Informatics and Mathematical Modeling – Subdivision of the Federal Research Centre Kola Science Centre of the Russian Academy of Sciences (IIMM KSC RAS)
[1]zuenko@iimm.ru, [2]lomov@iimm.ru

Abstract

Unlike relational DBMS that focused on maintaining of the referential integrity of rapidly changing transactional data, while using of RDF repositories which are storages for ontologies, there is no need to operate with such atomic structures as a tuple and you can use more suitable structures for grouping and summarizing of information. The article considers ontology as a set of relations (unary and binary), which are represented by specialized matrix-like structures – *C*-systems. That allows us to consider tasks of inference on ontologies as constraint satisfaction problems. The problem of a priori analysis and simplification of the SPARQL queries, considered in the article, is solved for ontologies that are developed using content ontology design patterns, which ensures the predictability of the structure of potential queries. In fact, each pattern is associated with a set of patterns of SPARQL queries. A method of a priori analysis and transformation of SPARQL queries patterns into a form, which speeds up the subsequent execution of concrete user queries has been developed. The method is based on the combining of the methods of structural decomposition and the author's methods for non-numeric constraints satisfaction. The structural decomposition methods make it possible to split the SPARQL query pattern into parts and parallelize its execution, which is especially important when accessing large volume RDF repositories. Due to the low frequency of modifications of the RDF repository, it is possible, based on the analysis of templates, to prepare data structures for the subsequent execution of concrete queries. This allows you to avoid unnecessary overhead incurred by repeated execution of the operation of joining the same relations when processing user requests described by a common pattern. For the join of relations, expressed as a set of *C*-systems, author's methods of non-numeric constraints satisfaction are applied, which are modifications of the known methods of maintaining node and arc consistency. As a result, the proposed approach to the ontology presentation and organization of inference procedures on ontologies allows to reduce the memory consumption and to ensure an acceptable speed of execution of SPARQL queries even for RDF repositories containing large amounts of data.

Key words: *ontology, ontological design patterns, constraint satisfaction problem, constraint programming, SPARQL, RDF repository.*

Citation: *Zuenko AA., Lomov PA.* Combining of methods of constraint propagation and structural decomposition for the a priori analysis of queries to the ontologies. *Ontology of designing*. 2018; 8(4): 571-593. DOI: 10.18287/2223-9537-2018-8-4-571-593.

Acknowledgment

This work was supported by the Russian Foundation for Basic Research (16-07-00377-a, 16-07-00562-a, 18-07-00615-a).

References

[1] ***Zuenko AA, Lomov PA, Oleinik AG.*** Application of constraint propagation techniques to speed up processing of queries to ontologies [In Russian]. – SPIIRAS Proceedings. 2017; 1(50): 112–136.
[2] Ontology of Integrated Knowledge Space. - https://github.com/palandlom/ontology-of-integrated-knowledge-space.
[3] ***Blomqvist E, Presutti V, Daga E, Gangemi A.*** Experimenting with eXtreme Design. In proceedings of EKAW 2010, LNCS 6317. Springer 2010. Berlin/Heidelberg/New York; 2010: 120–134.
[4] ***Russel S, Norvig P.*** Artificial Intelligence: A Modern Approach. 3rd edition. Prentice Hall; 2010.
[5] ***Bartak R.*** Constraint Programming: In Pursuit of the Holy Grail. Proceedings of the Week of Doctoral Students (WDS99), Part IV. – Prague: MatFyzPress; 1999: 555–564.

[6] ***Ruttkay Zs.*** Constraint satisfaction a survey. CWI Quarterly. 1998; 11: 163–214.
[7] ***Barto L, Kozik M.*** Constraint satisfaction problems solvable by local consistency methods. Journal of the ACM. 2014; 61(1), article 3: 1–19.
[8] ***Cheng KC, Yap RH.*** An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. Constraints. 2010; 15 (2): 265–304.
[9] ***Mackworth AK.*** Consistency in networks of relations. Artificial Intelligence. 1977; 8(1): 99–118.
[10] ***Mackworth AK, Freuder EC.*** Arc and path consistency revised, Artificial Intelligence. 1985; 25: 65–74.
[11] ***Gottlob G, Leone N, Scarcello F.*** A comparison of structural CSP decomposition methods. Artificial Intelligence. 2000; 124(2): 243–282.
[12] ***Katajainen J, Madsen JN.*** Performance tuning an algorithm for compressing relational tables. In Proceedings of the 8th Scandinavian Workshop on Algorithm Theory, volume 2368 of LNCS, Springer-Verlag; 2002: 398–407.
[13] ***Miguel I, Shen Q.*** Solution techniques for constraint satisfaction problems: foundations. Artificial Intelligence Review. 2001; 15: 241–265.
[14] ***Dechter R.*** Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. Artificial Intelligence. 1990; 41: 273–312.
[15] ***Dechter R, Pearl J.*** Network-based heuristics for constraint satisfaction problems. Artificial Intelligence. 1987; 34(1): 1–38.
[16] ***Freuder EC.*** A sufficient condition for backtrack-free search. Journal of the ACM. 1982; 29(1): 24–32.
[17] ***Dechter R, Pearl J.*** Tree clustering for constraint networks. Artificial Intelligence. 1989; 38(3): 353–366.
[18] ***Freuder EC.*** A sufficient condition for backtrack-bounded search. Journal of the ACM. 1985; 32(4): 755–761.
[19] ***Kulik BA, Zuenko AA, Fridman AYa.*** Algebraic approach to intelligent processing of data and knowledge [In Russian]. – St. Petersburg Polytechnic University Publishing House, 2010. -235 p.
[20] ***Zakrevskij A.*** Integrated Model of Inductive-Deductive Inference Based on Finite Predicates and Implicative Regularities. In: Naidenova, X., Ignatov, D. (eds.) Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems.– Hershey, PA:IGI Global; 2013: 1–12.
[21] ***Zuenko AA.*** Qualitative modeling of technical systems based on the propagation methods [In Russian]. Open Semantic Technologies for Intelligent Systems (OSTIS-2016): VI Intern. scientific-tech. Conf. (18-20 February, 2016, Minsk, Belarus) - Minsk: BSUIR; 2016: 573–578.
[22] ***Zuenko AA.*** Constraint inference based on the matrix representation of finite predicates [In Russian]. Artificial Intelligence and Decision Making. 2014; 3: 21–31.
[23] ***Zuenko AA.*** Methods for accelerating combinatorial search based on matrix representation of qualitative relationships in constraint satisfaction problems [In Russian]. Publishers of Irkutsk National Research Technical University. 2018; 22(7): 69–87.
[24] ***Zuenko AA.*** Matrix-Like Structures in Solution of Constraint Satisfaction Problems/ 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC) (18-25 August 2018, Vladivostok, Russia). IEEE; 2018. DOI: 10.1109/RPC.2018.8482223.

Сведения об авторах

Зуенко Александр Анатольевич, 1983 г.р., кандидат технических наук, старший научный сотрудник Института информатики и математического моделирования – обособленного подразделения ФИЦ «Кольский научный центр Российской академии наук». Области научных интересов: программирование в ограничениях; моделирование слабо формализованных предметных областей.

Alexander Anatolievich Zuenko, (b. 1983) PhD, a senior researcher of Institute for Informatics and Mathematical Modeling – Subdivision of the Federal Research Centre "Kola Science Centre of the Russian Academy of Sciences (IIMM KSC RAS). Research interests: constraint programming; poorly formalized subject domains modeling.

Ломов Павел Андреевич, 1984 г.р., к.т.н., с.н.с. Института информатики и математического моделирования – обособленного подразделения ФИЦ «Кольский научный центр Российской академии наук». Области научных интересов: представление знаний, онтологическое моделирование, Semantic web, информационная безопасность.

Pavel Andreevich Lomov, (b. 1984) PhD, a senior researcher of Institute for Informatics and Mathematical Modeling – Subdivision of the Federal Research Centre IIMM KSC RAS.
Research interests: knowledge representation, ontological modeling, Semantic web.