# ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004.75 : 004.89
DOI 10.21685/2072-3059-2020-1-1

*В. Н. Дубинин, А. В. Дубинин, Ч.-В. Янг, В. В. Вяткин*

## ИСПОЛЬЗОВАНИЕ ЯЗЫКА SPARQL В ОНТОЛОГИЧЕСКОМ МОДЕЛИРОВАНИИ МУЛЬТИАГЕНТНЫХ СИСТЕМ В СЕМАНТИЧЕСКОМ WEB

**Аннотация**

*Актуальность и цели.* Широкое распространение концепции мультиагентных систем и технологий семантического Web, а также начало их активного совместного использования в проектировании и реализации информационных систем требует разработки новых адекватных методов и средств их проектирования.

*Материалы и методы.* Исследования выполнены с использованием онтологического и агентно-ориентированного подходов к моделированию, технологий семантического Web, а также положений теории графовых трансформаций и теории сетей Петри.

*Результаты.* Предложен подход к онтологическому моделированию мультиагентных систем семантического Web. Данный подход продемонстрирован на примере мультиагентной системы в сети Smart Grid для выполнения маркетинговых операций. В рамках предложенного подхода разработан метод реализации сетей Петри на основе онтологий и языка SPARQL Update.

*Выводы*. Предложенный мультимодельный подход к онтологическому моделированию систем является эффективным, поскольку позволяет описание не только статики системы, но и ее динамики, что было продемонстрировано на примерах. Язык SPARQL Update является наиболее удобным и мощным средством реализации данного подхода.

**Ключевые слова**: онтология, моделирование, мультиагентные системы, семантический Web, трансформация графов, RDF, сети Петри, Smart Grid, SPARQL Update.

*V. N. Dubinin, A. V. Dubinin, C.-W. Yang, V. V. Vyatkin*

## THE USE OF SPARQL LANGUAGE IN ONTOLOGICAL MODELING OF MULTI-AGENT SYSTEMS IN SEMANTIC WEB

**Abstract**.

*Background*. The widespread use of multi-agent systems conception and technologies of the semantic Web, as well as the beginning of their active joint use in the design and implementation of information systems, requires the development of new adequate methods and tools for their design.

*Materials and methods*. The studies were performed using ontological and agent-based approaches to modelling, semantic Web technologies, as well as provisions of the theories of graph transformations and Petri nets.

*Results*. An approach to ontological modelling of multi-agent systems of the semantic Web is proposed. This approach is demonstrated by an example of a multi-agent system in the Smart Grid for marketing operations. Within the framework of the proposed approach, a method for implementing Petri nets based on ontologies and the SPARQL Update language has been developed.

*Conclusions*: The proposed multi-model approach to ontological modelling of systems is effective because it allows the description of not only the statics of the system, but also its dynamics, which was demonstrated by examples. The SPARQL Update language is the most convenient and powerful tool for implementing this approach.

**Keywords**: ontology, modeling, multi-agent systems, semantic Web, graph transformation, RDF, Petri nets, Smart Grid, SPARQL Update.

## Введение

Концепция семантического Web, введенная в 1998 г. Тимом Бернерсом-Ли, положила начало грандиозной работы в направлении создания Web нового поколения [1]. Семантический Web представляет собой расширение существующего «синтаксического» Web, позволяющее агентам понимать смысл информации в Интернете. В технологиях семантического Web значения представляются с помощью онтологий. Онтология включает две части – T-Box и A-Box, определяющие интенсиональные и экстенсиональные знания соответственно. Для описания онтологий наиболее часто используются языки RDFS и OWL [2]. В любом случае онтологию на нижнем уровне можно представить с использованием RDF. Основным понятием RDF является триплет, а в целом RDF-описание представляет собой граф (RDF-граф) [2]. Для извлечения информации из RDF-хранилищ был предложен ряд языков, среди которых выделяется язык SPARQL, одобренный консорциумом W3C [3]. В 2013 г. была принята текущая версия SPARQL 1.1, одним из важнейших расширений которой является включение в состав языка средств манипулирования RDF-данными (SPARQL 1.1 Update).

В настоящее время при построении интернет-приложений большое применение находит агентно-ориентированный подход [4]. Известно, что в мультиагентной системе агенты имеют несколько важных характеристик: автономность, ограниченность представления и децентрализация. В состав мультиагентной системы могут входить не только программные агенты, но также аппаратные агенты, роботы, люди или команды людей. Структура агента, как правило, предполагает наличие у него определенного интеллекта. Предполагается, что в среде семантического Web агенты могут иметь или использовать онтологические базы знаний, что и определяет специфику соответствующей предметной области.

В данной статье рассматриваются подходы к моделированию мультиагентных систем на основе технологий семантического Web, причем такое мо-

делирование возможно на различных уровнях абстракции, что позволяет использовать подход как для симуляции, так и для решения задач прототипирования и реализации.

## 1. Метод онтологического моделирования

Для моделирования мультиагентных систем, функционирующих в среде семантического Web, предлагается использовать подход, основанный на управлении онтологиями (ontology-driven approach) [5]. При этом система представляется как онтология или совокупность взаимосвязанных онтологий, а функционирование системы – как последовательность изменения онтологий. Онтология описывает структуру системы взаимодействующих агентов, сообщения, которыми они обмениваются между собой, а также наборы данных и знаний, с которыми работают агенты. Также можно сказать, что онтология в данном случае определяет обобщенное состояние системы. В нашем случае будут использоваться только RDF(S)-онтологии, и, таким образом, состояние системы представляет собой экземпляр RDF-графа. В процессе функционирования, как правило, будет изменяться онтология A-Box, однако при радикальных изменениях, происходящих в системе, возможно также и изменение онтологии T-Box.

Для реализации трансформации онтологий предлагается использование языка SPARQL Update, являющегося на данный момент самым мощным средством преобразования RDF-графов [3]. Сам по себе язык SPARQL Update является самодостаточным для решения поставленной задачи. Однако для повышения уровня формализации и, таким образом, для увеличения качества и надежности проектов предлагается использование формальных моделей в процессе проектирования. В первую очередь это графы, системы трансформации (перезаписи) графов, а также сети Петри (СП), с помощью которых можно описать динамику функционирования систем. Следует отметить, что существует ряд работ по моделированию мультиагентных систем как с использованием систем трансформации графов (например, [6]), так и СП (например, [7]).

Попытаемся неформально установить соответствие между системами трансформации графов и языком SPARQL Update. В классических системах трансформации графов (например, AGG [8]), как правило, используется поочередное выполнение правил, в результате которого изменяется основной граф. При этом порядок выполнения правил подчинен какой-либо стратегии (случайная выборка, приоритетная, по слоям, в соответствии с управляющей схемой и т.д.). В SPARQL-системе запросы типа DELETE-INSERT-SELECT представляют, по сути, специфичное правило трансформации. При этом SELECT-предложение определяет левую часть правила, а предложения DELETE и INSERT – правую часть. Выполнение одного запроса SPARQL Update можно представить как двухфазную трансформацию RDF-графа. Назовем текущий RDF-граф непосредственно перед выполнением запроса SPARQL Update исходным. Результирующий граф первоначально равен исходному.

На первой фазе производится сопоставление графа, определяемого SELECT-предложением (шаблона графа), с исходным RDF-графом. Для найденного варианта сопоставления с использования DELETE и INSERT

предложений модифицируется результирующий граф путем удаления и/или включения соответствующих вершин и дуг графа. Эта процедура повторяется для всех вариантов сопоставлений SELECT-графа с исходным графом. Важно, что трансформации на первой фазе не изменяют исходный граф. На второй фазе текущим становится результирующий граф. Очевидно, что в рамках одного запроса SPARQL Update фактически выполняется серия правил перезаписи графов.

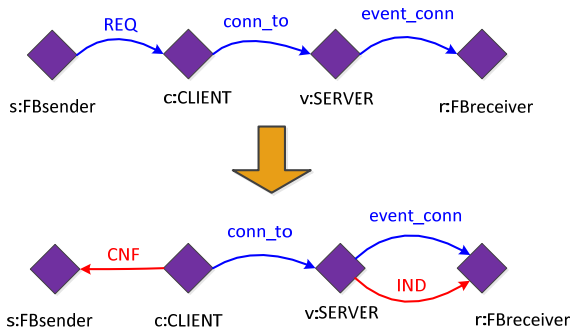
Следует обратить внимание на сложность реализации произвольного управления выполнением правил трансформации графа в SPARQL-системе. Правило (вернее, серия правил) в SPARQL-системе выполняется однократно. Поскольку все транзакции SPARQL являются линейными (так как операторы управления выполнением транзакций отсутствуют), то в лучшем случае можно выполнить последовательность из серий правил перезаписи графов. Для организации циклических стратегий управления выполнением правил необходима разработка внешних программ (клиентов). Самым простым подходом к организации взаимодействия подобного клиента со SPARQL-сервером является использование протокола HTTP. Вместе с тем стоит отметить большую выразительную мощность языка SPARQL (в некоторых аспектах) по сравнению с системами трансформации графов за счет возможности использования подзапросов, механизмов фильтрации, группирования и сортировки, указания необязательных частей графа, возможности одновременной работы с несколькими графами.

Возможны два способа использования формальных моделей в предложенном онтологическом подходе: А1 – как основного средства описания с последующей реализацией на онтологических языках (включая SPARQL); А2 – как вспомогательного средства для описания, улучшения восприятия и, возможно, дальнейшего анализа с использованием специализированных средств. При этом А0 назовем подходом, в котором формальные модели вообще не используются.

Следует отметить, что в случаях А1 и А2 возможен *мультимодельный* подход, когда в моделировании используются разноплановые модели. Например, возможно одновременное использование онтологических и графотрансформационных моделей, СП, а также других моделей переходов состояний. При этом существует два подхода к интеграции моделей: Б1 – на уровне онтологий; Б2 – на уровне реализации, причем для каждой из моделей на языке SPARQL разрабатывается собственный интерпретатор. В этом случае для перехода от одной модели к другой необходима разработка «переходников», производящих соответствующие преобразования. Например, конкретной дуге RDF-графа может быть поставлена в соответствие маркированная позиция ординарной СП, в то время как шаблону дуги RDF-графа, содержащему переменные, может быть поставлена в соответствие позиция высокоуровневой СП.

В качестве простейшего примера использования подхода А1 рассмотрим правило перезаписи RDF-графа, приведенное на рис. 1, моделирующее нормальную передачу данных с использованием коммуникационных функциональных блоков (ФБ) CLIENT/SERVER в международном стандарте IEC 61499 [9]. При приходе события-запроса REQ на ФБ CLIENT по сети TCP/IP производится передача данных на ФБ SERVER, при этом на его выходе появ-

ляется событие-индикация IND. На рис. 1 ромбами представлены экземпляры классов (иначе, индивиды).




```
DELETE
  {s REQ c.}
INSERT
  {c CNF s.
   v IND r.}
WHERE
  {s REQ c.
   c conn_to v.
   v event_conn r.}
```

Рис. 1. Простое правило перезаписи RDF-графа
и реализующий его SPARQL-запрос (справа)

На рис. 1 используются составные имена, где первая часть имени определяет имя индивида, а вторая – имя класса, к которому он принадлежит. На основе данного формального правила был построен SPARQL-запрос, представленный в правой части рис. 1. Для упрощения в запросе не указывается факт принадлежности индивидов классам, а также префиксы пространств имен. Следует сразу отметить, что выразительная мощь правила значительно возрастает, оно универсализируется при использовании переменных в графе шаблона. Для модификации правила в данном направлении необходимо перед именами индивидов поставить знак «?», например: *?s REQ ?c*.

В качестве формальной модели в подходе A1 могут быть выбраны СП. На рис. 2 приведена ординарная СП, а на рис. 3 – высокоуровневая СП, моделирующие правило из рис. 1.

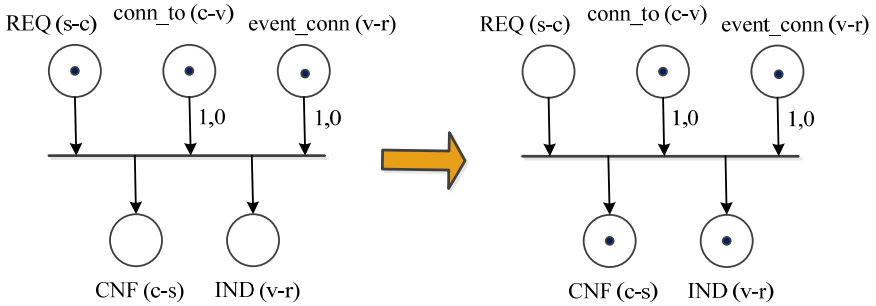



Рис. 2. Моделирование правила из рис. 1 с использованием ординарных сетей Петри

Из особенностей моделирования можно выделить следующее. Позиция ординарной СП моделирует RDF-триплет, а позиция высокоуровневой СП – шаблон триплета с переменными. Причем в последнем случае учитывается маркировка исходящих дуг. Маркировка позиции ординарной СП показывает наличие/отсутствие конкретного RDF-триплета в хранилище, а маркировка позиции высокоуровневой СП – сами RDF-триплеты, соответствующие шаблону триплета. SPARQL-правило моделируется переходом СП. Правило раз-

решенности перехода СП соответствует паттерну графа в разделе WHERE запроса SPARQL, а правило срабатывания – разделам INSERT и DELETE.

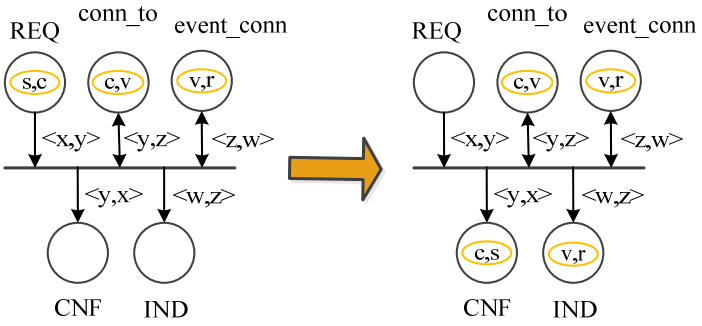



Рис. 3. Моделирование правила из рис. 1 с использованием высокоуровневых сетей Петри

Следует также отметить, что с использованием SPARQL возможно моделирование самомодифицирующихся СП, при функционировании которых меняется не только маркировка, но и структура сети. Реализация высокоуровневых СП на языке логического программирования Пролог, имеющего такую же теоретическую основу, что и онтологии (логика первого порядка), подробно рассмотрена в работе [10].

Построение интерпретатора СП на языке SPARQL, необходимом при использовании подхода А1Б2, рассмотрено в разд. 3. Подход А0 и подход А2 с использованием графовых трансформаций как вспомогательной формальной модели представлен в разд. 2.

## 2. Онтологическое моделирование агентов сети Smart Grid

В настоящее время в области электроэнергетики все большее распространение и признание получает концепция *Smart Grid*. Под сетью *Smart Grid* понимается разумная сеть электроснабжения будущего, которая включает коммуникационную инфраструктуру и двунаправленный поток электроэнергии и предоставляет информацию в режиме реального времени для всех вовлеченных объектов [11].

Для управления сетью *Smart Grid* на нижних уровнях (уровне управления и SCADA) могут использоваться физические агенты, построенные в соответствии со стандартами IEC 61850 и IEC 61499 [12], в то время как на верхних уровнях (например, ERP) – программные агенты. Ниже в иллюстративных целях рассматривается простой пример мультиагентной системы для осуществления маркетинговых операций в Smart Grid. При этом предполагается, что в системе функционируют агенты разных уровней. Онтология T-Box, описывающая структуру взаимодействующих агентов, представлена на рис. 4.

Классы агентов представлены следующим образом: агенты энергетического рынка – в виде треугольника, физические управляющие агенты – в виде квадрата. Классы, относящиеся к физическим объектам, представляются в виде прямоугольников, окружностей и звездочек. В собственно класс физических сущностей как подклассы входят класс единичных физических сущ-

ностей и класс их агрегаций. Примерами физических сущностей являются дата-центры, гидроэлектростанции, солнечные батареи. Примерами агрегаций сущностей могут служить фермы (агрегаторы) солнечных батарей, виртуальные электростанции.

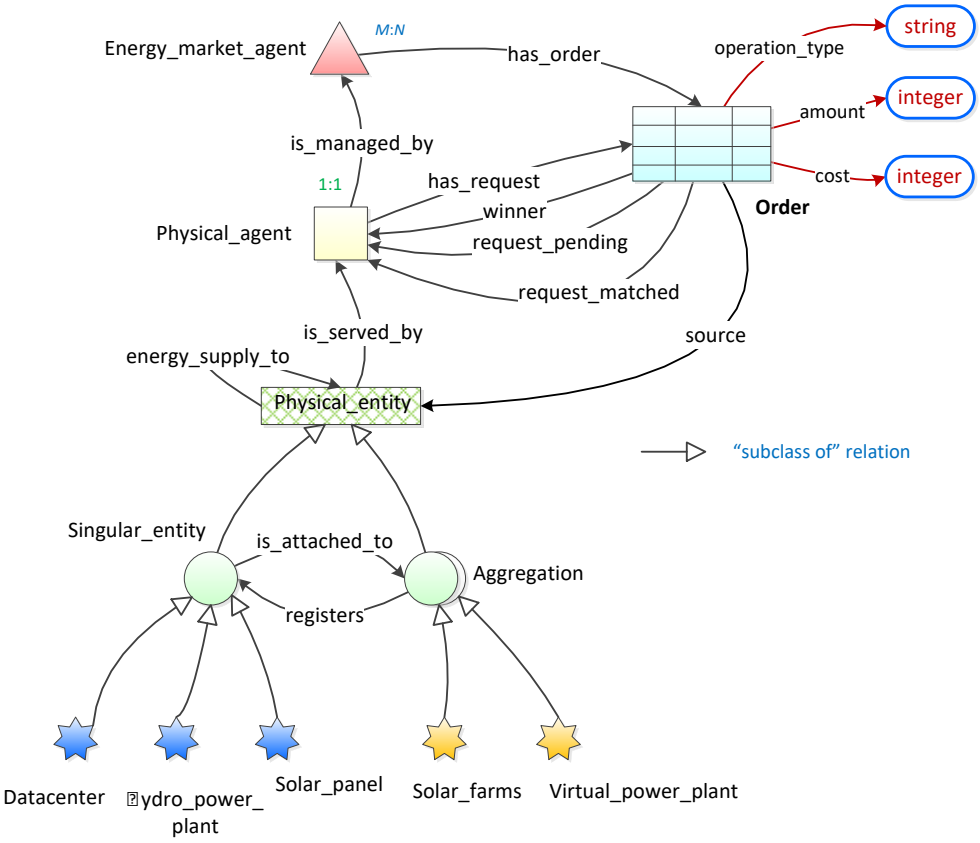



Рис. 4. Онтология, описывающая систему взаимодействующих агентов в Smart Grid

В онтологии также представлен набор данных *Order* для хранения заказов на рынке электроэнергии. Для представления типов данных в онтологии используются овалы. Объектное свойство *source* позволяет определить физическую сущность, которая генерирует запрос. Свойства по данным имеют следующую интерпретацию: *operation_type* – тип операции (продажа или покупка); *amount* – объем продаваемой или покупаемой электроэнергии; *cost* – стоимость единицы электроэнергии.

В системе агентов возможны различные сценарии функционирования, один из сценариев приведен ниже:

а) ожидается, что дата-центр как объект физического уровня будет иметь повышенную нагрузку в течение следующих 24 ч и он отправляет запрос с желаемым количеством требуемой электроэнергии своему агенту на энергетическом рынке (рыночному агенту дата-центра);

б) рыночный агент дата-центра обрабатывает запрос от агента физического уровня и декларирует этот запрос нагрузки на рынке электроэнергии;

в) происходит процесс купли-продажи, при котором каждый агент рынка может рекламировать свои генерирующие мощности и стоимость генерации;

г) после окончания маркетингового процесса «победитель» отправляет желаемую сумму генерируемой энергии своему агенту на физическом уровне;

д) если, например, некоторый агрегатор солнечных батарей становится «победителем» маркетингового процесса, то менеджер агрегатора затем объявляет желаемое количество электроэнергии на свои отдельные солнечные панели, и каждая солнечная батарея будет кооперироваться для достижения этой общей цели.

Пример создания экземпляра o777 запроса приведен ниже. Для этого используется оператор INSERT DATA:

```
INSERT DATA {m:o777 rdf:type m:Order.  m:o777 m:source m:dc99.
m:o777 m:operation_type 'buy'.  m:o777 m:amount 15.  m:o777 m:cost 7. }
```

Фрагмент кода ниже описывает создание экземпляра торгового агента (ma49), связывание его с физическим управляющим агентом pa55, а также со своими заказами:

```
INSERT DATA {m:ma49 rdf:type m:Energy_market_agent.};
INSERT DATA {m:pa55 m:is_managed_by m:ma49.};
INSERT DATA
{m:ma49 m:has_order m:o21.  m:ma49 m:has_order m:o22.
m:ma49 m:has_order m:o23. m:ma49 m:has_order m:o24.}
```

Ниже представлен SPARQL-запрос, описывающий функционирование системы в соответствии со сценарием, приведенным выше:

```
DELETE {?pa m:has_request ?o1.}
INSERT {?o2 m:winner ?pa.  ?o1 m:request_matched ?pa.  ?s2 m:energy_supply_to ?s1.}
WHERE
{SELECT ?o2 ?pa ?o1 ?s2 ?s1
WHERE
{?pa m:has_request ?o1.  ?o1 m:source ?s1.  ?o1 m:operation_type 'buy'.  ?o1 m:amount
?a1.
?o1 m:cost ?c1.  ?pa m:is_managed_by ?ma.  ?ma m:has_order ?o2.
?o2 m:operation_type 'sale'.  ?o2 m:source ?s2.  ?o2 m:amount ?a2.
FILTER(?a2>=?a1)
?o2 m:cost ?c2.
FILTER(?c2<=?c1) }
ORDER BY ?c2 DESC(?a2)
LIMIT 1 }
```

Его смысл можно выразить следующим образом. Если физический агент имеет запрос на покупку электроэнергии, то определяется соответствующий рыночный агент и делается попытка найти среди его заказов такой, который полностью удовлетворяет условиям запроса и является в некотором смысле оптимальным. При этом также фиксируется, что запрос согласован. После этого физический объект – поставщик электроэнергии – напрямую связывается с физическим объектом – потребителем электроэнергии, используя свойство *energy_supply_to*. Наконец, запрос, посланный физическим агентом,

«отвязывается» от него. Условием для признания запроса победителем является минимальная стоимость и максимум объема электроэнергии. Следует отметить, что для сокращения размера SPARQL-запроса описание префиксов пространств имен опущено. В данном случае префикс *m* соответствует онтологии агентов T-Box+A-Box.

Как видно из приведенного выше запроса, отбираются решения (заказы рыночного агента), удовлетворяющие следующим условиям оптимальности:

1) указанный в запросе физического агента объем покупаемой электроэнергии должен быть не больше объема продаваемой электроэнергии, а стоимость покупаемой единицы электроэнергии должна быть больше стоимости продаваемой единицы электроэнергии в заказе рыночного агента;

2) из заказов рыночного агента, удовлетворяющих верхнему условию, отбирается заказ, имеющий минимальную стоимость продаваемой единицы электроэнергии при наибольшем объеме продаваемой электроэнергии.

Несмотря на некоторые различия в семантике выполнения классических систем трансформации графов и запросов SPARQL Update, их синтаксические представления имеют сходства, что определяет смысл использовать графическое представление для последних. Для представления правил предлагается использовать графическую нотацию, предложенную в работе [13]. При этом индивид представляется прямоугольником, в верхней части которого записывается имя класса, а в нижней – или идентификатор экземпляра, или имя переменной (через знак ?). Для представления значения свойства по данным используется овал синего цвета, внутри которого может находиться константа или переменная. Свойства представляются дугами, надписанными или именами свойств, или именами переменных.

На рис. 5 приведено графическое представление запроса SPARQL Update, представленного выше. В данном случае левая часть правила перезаписи графа приведена над стрелкой, а правая – под стрелкой. Слева от стрелки – условие применимости правила из предложений FILTER. Справа от стрелки – дисциплина упорядочения и число применений правила. Для большей наглядности для каждого индивида указан его класс (хотя в самом SPARQL-запросе триплеты, описывающие принадлежность индивида классу, отсутствуют). Генерируемые дуги в графе правой части правила отмечены красным цветом. Тестирование запроса SPARQL Update, представленного на рис. 5, было выполнено на сервере Apache Jena Fuseki.

### 3. Онтологическое моделирование сетей Петри

Рассмотрим моделирование и реализацию СП с использованием онтологий. Особенностью рассматриваемых СП являются наличие числовых и ингибиторных дуг, а также приоритетов переходов. Формально данный класс СП определяется следующим кортежем:

$$(P, T, X, Y, Z, W_X, W_Y, W_Z, Q, A, G, m_0),$$

где $P$ – конечное множество позиций; $T$ – конечное множество переходов; $X \subseteq P \times T$ – множество входных дуг переходов с порогом-минимумом (дуги с проверкой на «больше»); $Y \subseteq T \times P$ – множество выходных дуг переходов; $Z \subseteq P \times T$ – множество входных дуг переходов с порогом-максимумом (дуги с проверкой на «меньше»); $W_X$: $X \rightarrow N^+ \times N_0$ – функция весов дуг типа $X$;

$W_Y: Y \to N^+$ – функция весов дуг типа $Y$; $W_Z$: $Z \to N^+$ – функция весов дуг типа $Z$; $Q$: $T \to N_0$ – функция приоритетов переходов (чем меньше значение, тем выше приоритет); $m_0$ – начальная маркировка. Первый компонент веса дуги типа $X$ указывает минимальный порог для срабатывания перехода, а второй – число удаляемых меток из инцидентной дуге позиции.

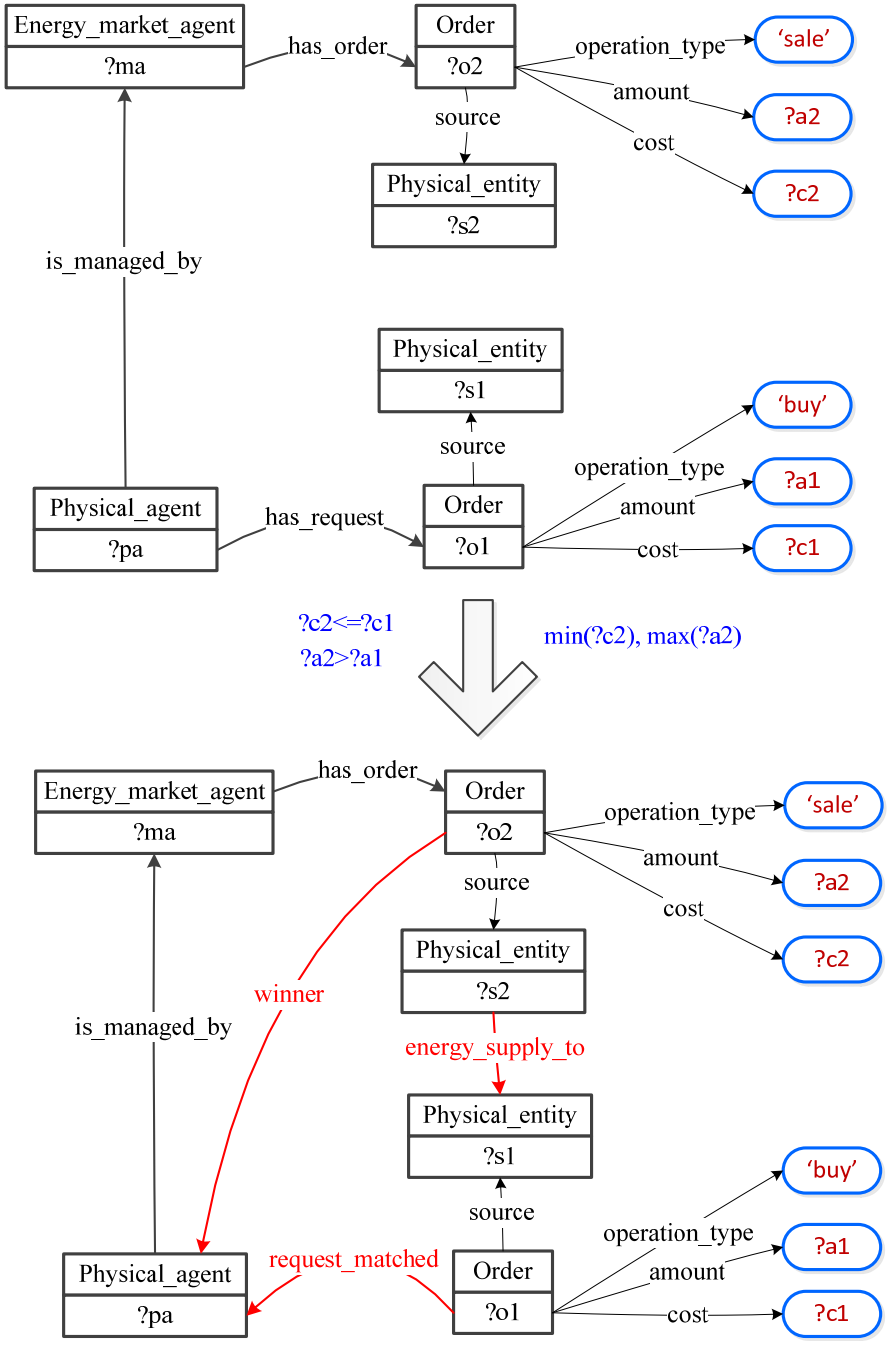



Рис. 5. Представление запроса SPARQL Update
в виде правила трансформации графов

Разработана онтология уровня *T-Box* для данного класса сетевых моделей, представленная на рис. 6.

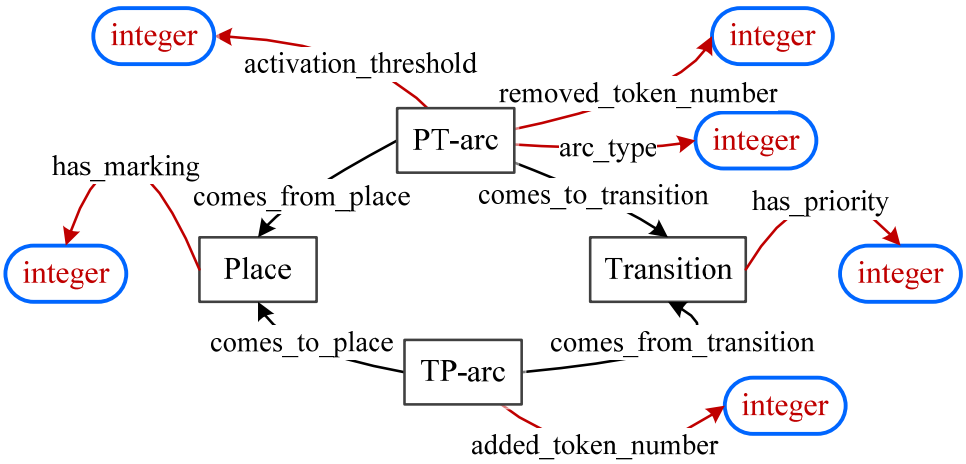



Рис. 6. Онтология T-Box сетей Петри

Она включает: четыре класса – *Place* (Позиция), *Transition* (Переход), *PT-arc* и *TP-arc* (входная и выходная дуга перехода соответственно); четыре объектных свойства – *comes_from_place, comes_from_transition, comes_to_place* и *comes_to_transition*, соответствующие дугам сетевой модели и показывающие источник и приемник дуги; шесть свойств по данным – *has_marking* (маркировка позиции), *arc_type* (тип дуги), *activation_threshold* (порог активизации входной дуги перехода), *removed_token_number* (число удаляемых из позиции меток при срабатывании перехода), *added_token_number* (число добавляемых в позицию меток при срабатывании перехода) и *has_priority* (приоритет перехода). Следует отметить, что существует ряд исследований по разработке онтологий СП разных классов (например [14]), однако все работы рассматривают только онтологическое моделирование структур сетевых моделей, но не касаются их динамики.

С использованием языка SPARQL Update можно промоделировать только одно срабатывание перехода. Однако даже эта задача является нетривиальной по ряду причин:

1) каждый конкретный переход имеет свою конфигурацию (т.е. множества входных и выходных дуг);

2) одна и та же позиция может быть для перехода одновременно входной и выходной. Для разрешения конфликта, когда в позицию одновременно должны добавляться и удаляться метки, класс входовыходных позиций рассматривается отдельно. Соответственно новая маркировка этих позиций вычисляется по отдельной формуле.

SPARQL-запрос, моделирующий срабатывание перехода СП, имеет следующую структуру:

1) фрагмент кода для удаления триплетов маркировки;

2) фрагмент кода для добавления триплетов маркировки;

3) подзапрос для определения активного перехода;

4) фрагмент кода для подсчета новой маркировки входной или входовыходной позиции;

5) фрагмент кода для подсчета новой маркировки выходной позиции.

Первые два фрагмента:

```
DELETE
{?p1 m:has_marking ?n1.    # Удалить прежнюю маркировку входных и входовыходных позиций
?p2 m:has_marking ?n2.}    # Удалить прежнюю маркировку выходных позиций
INSERT
{?p1 m:has_marking ?n1new.  # Включить новую маркировку входных и входовыходных позиций
?p2 m:has_marking ?n2new.}  # Включить новую маркировку выходных позиций
```

Под активным переходом понимается разрешенный переход с максимальным приоритетом, который будет срабатывать. Переход разрешен в том случае, когда число входных дуг перехода равно числу разрешенных входных дуг перехода. При вычислении подзапроса используется группирование (GROUP BY), фильтрация (FILTER), упорядочение (ORDER BY) и ограничение числа результатов (LIMIT). В самом подзапросе используются два других подзапроса. Подзапрос для определения активного перехода:

```
{SELECT ?t
WHERE
{?t rdf:type m:Transition.
?t m:has_priority ?pr.
{SELECT ?t (COUNT(?a) AS ?na)   # Подсчет числа входных дуг перехода
WHERE
{?a m:comes_to_transition ?t}
GROUP BY ?t}
{SELECT ?t (COUNT(?a) AS ?ne)   # Подсчет числа разрешенных входных дуг перехода
WHERE
{?a m:comes_to_transition ?t. ?a m:comes_from_place ?p.
?p m:has_marking ?n. ?a m:activation_threshold ?k. ?a m:arc_type ?s.
FILTER (?s='n' && ?n>=?k || ?s='i' && ?n<=?k)}
GROUP BY ?t }
FILTER (?na=?ne)}   # Отбор: У разрешенного перехода все дуги должны быть разрешенными
ORDER BY ?pr     # Упорядочить разрешенные переходы по убыванию приоритетов
LIMIT 1}
```

Ниже представлен фрагмент кода, определяющий, является ли входная позиция ?p1 также и выходной, и вычисляющий новую маркировку позиции ?p1. В данном фрагменте переменная ?k1 определяет число удаляемых меток:

```
OPTIONAL    # Определяем, является ли входная позиция ?p1 также и выходной
{?a2 rdf:type m:TP-arc.
?a2 m:comes_from_transition ?t.
?a2 m:comes_to_place ?p1.
?a2 m:added_token_number ?k2a.}
BIND ((IF (bound(?k2a),?n1-?k1+?k2a, ?n1-?k1)) AS ?n1new)  # Подсчет новой маркировки
```

## Заключение

В работе представлен подход к онтологическому моделированию мультиагентных систем с использованием языка SPARQL. Направлением дальнейших исследований является развитие мультимодельного подхода к моделированию и проектированию мультиагентных систем и киберфизических систем с использованием моделей переходов состояний, графодинамических систем и технологий семантического Web.

### *Библиографический список*


1. Semantic Web. W3C Consortium. – URL: https://www.w3.org/standards/semanticweb/
2. **Allemang, D.** Semantic Web for the Working Ontologist. Modeling in RDF, RDFS and OWL / D. Allemang, J. Hendler. – Morgan Kaufmann Publishers, 2008. – 349 p.
3. **DuCharme, B.** Learning SPARQL: Querying and Updating with SPARQL 1.1 / B. DuCharme. – O'Reilly, 2011. – 237 p.
4. **Wooldridge, M.** An Introduction to MultiAgent Systems / M. Wooldridge. – John Wiley & Sons Ltd, 2002. – 366 p.
5. Ontology-Driven Software Development / J. Z. Pan, S. Staab, U. Aßmann, J. Ebert, Y. Zhao. – Berlin : Springer, 2012. – 355 p.
6. **Knirsch, P.** A Note on Modeling Agent Systems by Graph Transformation / P. Knirsch, H.-J. Kreowski // Lecture Notes in Computer Science. – 2000. – Vol. 1779. – P. 79–86.
7. **Celaya, J. R.** Modeling and Analysis of Multi-agent Systems using Petri Nets / J. R. Celaya, A. A. Desrochers, R. J. Graves // Journal of computers. – 2009. – Vol. 4, № 10. – P. 981–996.
8. **Taenzer, G.** AGG: A Tool Environment for Algebraic Graph Transformation / G. Taenzer // Lecture Notes in Computer Science. – 2000. – Vol. 1779. – P. 481–490.
9. **Vyatkin, V.** IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design / V. Vyatkin. – Third Edition. – ISA, o3neida, 2015. – 261 p.
10. **Дубинин, В. Н.** Децентрализованное управление распределенными данными в локальной вычислительной сети: спецификация, моделирование, реализация и применение в распределенных вычислениях / В. Н. Дубинин. – Пенза : Изд-во Пенз. гос. технол. ун-та, 1997. – 78 с. – Деп. в ВИНИТИ 10.06.97. – № 1946-В97.
11. **Ipakchi, A.** Grid of the future / A. Ipakchi, F. Albuyeh // IEEE Power and Energy Magazine. – 2009. – Vol. 7, № 2. – P. 52–62.
12. **Zhabelova, G.** Multi-agent Smart Grid Automation Architecture based on IEC 61850/61499 Intelligent Logical Nodes / G. Zhabelova, V. Vyatkin // IEEE Transactions on Industrial Electronics. – 2012. – Vol. 59, № 5. – P. 2351–2362.
13. **Yang, C.-W.** Ontology Driven Approach to Generate Distributed Automation Control from Substation Automation Design / C.-W. Yang, V. Dubinin, V. Vyatkin // IEEE Transactions on Industrial Informatics. – 2017. – Vol. 13, № 2. – P. 668–679.
14. **Gašević, D.** Interoperable Petri Net Models via Ontology / D. Gašević, V. Devedžić // International Journal of Web Engineering and Technology. – 2007. – Vol. 3, № 4. – P. 374–396.


### *References*


1. *Semantic Web. W3C Consortium*. Available at: https://www.w3.org/standards/semanticweb/
2. Allemang D., Hendler J. *Semantic Web for the Working Ontologist. Modeling in RDF, RDFS and OWL*. Morgan Kaufmann Publishers, 2008, 349 p.

3. DuCharme B. *Learning SPARQL: Querying and Updating with SPARQL 1.1*. O'Reilly, 2011, 237 p.
4. Wooldridge M. *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd, 2002, 366 p.
5. Pan J. Z., Staab S., Aßmann U., Ebert J., Zhao Y. *Ontology-Driven Software Development*. Berlin: Springer, 2012, 355 p.
6. Knirsch P. A, Kreowski H.-J. *Lecture Notes in Computer Science*. 2000, vol. 1779, pp. 79–86.
7. Celaya J. R., Desrochers A. A., Graves R. J. *Journal of computers*. 2009, vol. 4, no. 10. pp. 981–996.
8. Taenzer G. *Lecture Notes in Computer Science*. 2000, vol. 1779, pp. 481–490.
9. Vyatkin V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. Third Edition. ISA, o3neida, 2015, 261 p.
10. Dubinin V. N. *Detsentralizovannoe upravlenie raspredelennymi dannymi v lokal'noy vychislitel'noy seti: spetsifikatsiya, modelirovanie, realizatsiya i primenenie v raspredelennykh vychisleniyakh* [Decentralized direction of distributed data in a local area network: specification, modeling, implementation and application in distributed computing]. Penza: Izd-vo Penz. gos. tekhnol. un-ta, 1997, 78 p. Dep. v VINITI 10.06.97, no. 1946-V97. [In Russian]
11. Ipakchi A., Albuyeh F. *IEEE Power and Energy Magazine*. 2009, vol. 7, no. 2, pp. 52–62.
12. Zhabelova G., Vyatkin V. *IEEE Transactions on Industrial Electronics*. 2012, vol. 59, no. 5, pp. 2351–2362.
13. Yang C.-W., Dubinin V., Vyatkin V. *IEEE Transactions on Industrial Informatics*. 2017, vol. 13, no. 2, pp. 668–679.
14. Gašević D., Devedžić V. *International Journal of Web Engineering and Technology*. 2007, vol. 3, no. 4, pp. 374–396.

***Дубинин Виктор Николаевич***
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dubinin.victor@gmail.com

***Dubinin Viktor Nikolaevich***
Doctor of engineering sciences, professor, sub-department of computer engineering, Penza State University (40, Krasnaya street, Penza, Russia)

***Дубинин Алексей Викторович***
студент, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dubinin.aleksey@gmail.com

***Dubinin Aleksey Viktorovich***
Student, Penza State University (40, Krasnaya street, Penza, Russia)

***Чен-Вэй Янг***
постдок, кафедра «Ответственные коммуникации и вычисления», Технический университет Лулео (Швеция, г. Лулео, ул. Регнбогсаллен, корп. А)

E-mail: chen-wei.yang@ltu.se

***Chen-Wei Yang***
Postdoctoral researcher, sub-department of responsible communications and computations, Lulea University of Technology (building A, Regnbogsallen street, Lulea, Sweden)

***Вяткин Валерий Владимирович***
доктор технических наук, профессор, кафедра «Ответственные коммуникации и вычисления», Технический университет Лулео (Швеция, г. Лулео, ул. Регнбогсаллен, корп. А)

E-mail: valeriy.vyatkin@ltu.se

***Vyatkin Valeriy Vladimirovich***
Doctor of engineering sciences, professor, sub-department of responsible communications and computations, Lulea University of Technology (building A, Regnbogsallen street, Lulea, Sweden)

**Образец цитирования:**

Дубинин, В. Н. Использование языка SPARQL в онтологическом моделировании мультиагентных систем в семантическом Web / В. Н. Дубинин, А. В. Дубинин, Ч.-В. Янг, В. В. Вяткин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2020. – № 1 (53). – С. 4–18. – DOI 10.21685/2072-3059-2020-1-1.