УДК 004.75 : 004.89
doi:10.21685/2072-3059-2022-1-6

Моделирование раскрашенных сетей Петри с использованием технологий семантического web

В. Ю. Каев[1], В. Н. Дубинин[2], А. В. Дубинин[3], Л. П. Климкина[4]

[1,2,3]Пензенский государственный университет, Пенза, Россия
[4]Пензенский государственный аграрный университет, Пенза, Россия

[1]veld4n@gmail.com, [2]dubinin.victor@gmail.com,
[3]dubinin.aleksey@gmail.com, [4]ludmila.klimkina@gmail.com

Аннотация. *Актуальность и цели.* В связи с широким распространением концепции семантического web представляют интерес вопросы использования формальных моделей при проектировании параллельных, распределенных и мультиагентных систем на основе семантической паутины. Цель исследования – разработка методов и средств реализации высокоуровневых сетей Петри на основе онтологий, что может служить «мостом» к построению систем данного класса. *Материалы и методы.* Исследования выполнены с использованием онтологического подхода к моделированию сложных систем, технологий семантического web, языка SPARQL, а также положений теории высокоуровневых сетей Петри. *Результаты.* В процессе проведения исследования разработаны: онтология раскрашенных сетей Петри, описывающая как статические, так и динамические элементы сетевой модели; метод интерпретации раскрашенных сетей Петри на основе онтологий и языка SPARQL, позволяющий моделировать динамику сетевой модели как процесс изменения онтологии; комплекс программ OntoNet для распределенной интерпретации раскрашенных сетей Петри в среде семантического web; сетевая модель потоковой обработки событий, служащая демонстрационным примером для тестирования инструментальной системы. *Выводы*. Предложенные методы и средства онтологического моделирования и интерпретации раскрашенных сетей Петри могут быть использованы для быстрой разработки, прототипирования и реализации распределенных, параллельных и многоагентных систем обработки событий, данных и знаний в контексте семантического web.

Ключевые слова: раскрашенные сети Петри, моделирование, онтология, семантический web, SPARQL, инструментальные средства, OntoNet, распределенные системы, обработка данных и знаний

Для цитирования: Каев В. Ю., Дубинин В. Н., Дубинин А. В., Климкина Л. П. Моделирование раскрашенных сетей Петри с использованием технологий семантического web // Известия высших учебных заведений. Поволжский регион. Технические науки. 2022. № 1. С. 62–77. doi:10.21685/2072-3059-2022-1-6

Modeling of colored Petri nets using semantic Web technologies

V.Yu. Kaev[1], V.N. Dubinin[2], A.V. Dubinin[3], L.P. Klimkina[4]

[1,2,3]Penza State University, Penza, Russia
[4]Penza State Agricultural University, Penza, Russia

[1]veld4n@gmail.com, [2]dubinin.victor@gmail.com,
[3]dubinin.aleksey@gmail.com, [4]ludmila.klimkina@gmail.com

Abstract. *Background.* In connection with the widespread use of the Semantic Web concept, the issues of using formal models in the design of parallel, distributed and multi-agent

systems based on the Semantic Web are of interest. The purpose of the work is to develop methods and tools for implementing high-level Petri nets based on ontologies, which can serve as a “bridge” to the building of systems of this class. *Materials and methods.* The research was carried out using an ontological approach to modeling complex systems, semantic Web technologies, the SPARQL language, as well as the provisions of the theory of high-level Petri nets. *Results.* In the course of the research, the following were developed: 1) an ontology of colored Petri nets, which describes both static and dynamic elements of the net model; 2) a method for interpreting colored Petri nets based on ontologies and the SPARQL language, which allows modeling the dynamics of the net model as a process of ontology change; 3) OntoNet software for distributed interpretation of colored Petri nets in the Semantic Web environment; 4) a net model of streaming event processing, which serves as a demonstration example for testing the tool. *Conclusions.* The proposed methods and tools for ontological modeling and interpretation of colored Petri nets can be used for rapid development, prototyping and implementation of distributed, parallel and multi-agent systems for processing events, data and knowledge in the context of the Semantic Web.

Keywords: colored Petri nets, modeling, ontology, semantic Web, SPARQL, software, OntoNet, distributed systems, data and knowledge processing

For citation: Kaev V.Yu., Dubinin V.N., Dubinin A.V., Klimkina L.P. Modeling of colored Petri nets using semantic Web technologies. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2022;(1):62–77. (In Russ.). doi:10.21685/2072-3059-2022-1-6

Введение

Использование формальных моделей является необходимым атрибутом в проектировании большинства систем в области информатики, автоматики и вычислительной техники. Формальные модели используются для спецификации, верификации, оценки производительности, рефакторинга, прототипирования и реализации программно-аппаратных систем, комплексов и сетей. В течение нескольких десятилетий популярным модельным формализмом являются высокоуровневые сети Петри и их подклассы, в числе которых раскрашенные сети Петри (РСП) [1]. С использованием РСП возможно моделирование дискретных систем с параллельными процессами и сложными межпроцессными взаимодействиями. Данный класс сетей Петри отличает наличие свойств, присущих высокоуровневым языкам программирования, при сравнительно компактной форме параметризуемых моделей. К настоящему времени имеется множество реализаций высокоуровневых сетей Петри на различных платформах, в числе которых системы CPN Tools [1] (раскрашенные сети Петри) и Renew [2] (ссылочные сети).

В связи с широким распространением концепции семантического web [3] представляют интерес вопросы использования высокоуровневых сетей Петри при построении параллельных, распределенных и мультиагентных систем на основе семантической паутины, краеугольным камнем которой являются онтологии. Онтология есть явная спецификация концептуализации на уровне знаний [4]. Онтологическое моделирование в настоящее время является перспективным подходом к проектированию сложных систем [5]. Основным языком запросов в семантическом web, используемым для обработки онтологий, является язык SPARQL 1.1, одобренный консорциумом W3C [6].

Начало использованию онтологического представления сетей Петри (СП) было положено в работе [7]. В статье [8] предложена онтология сетей

Петри на основе языка OWL DL. Однако данная онтология представляет только статику сетевой модели. В работе [9] предложена онтология высокоуровневых СП на основе F-Logic, которая отражает в декларативном стиле как статическую структуру сетевой модели, так и динамику функционирования. Онтология реализована в системе FLORA-2, которая комбинирует логическое программирование и дедуктивные базы данных с парадигмой объектно-ориентированного программирования. На основе данной онтологии в статье [10] разработан «движок» OPENET для высокоуровневых сетей Петри. Инструментальная система имеет архитектуру «клиент-сервер», причем клиент использует RMI-интерфейс для доступа к системе OPENET. Данная система была успешно использована для реализации модулей обучения в методологии IMS LD [11]. Определенным недостатком системы OPENET является ее ориентация на онтологическое представление, которое не является мейнстримом в современном семантическом web, что может затруднить ее интеграцию с другими системами современного семантического web.

В данной работе предлагаются:

1) OWL DL-онтология раскрашенных сетей Петри, описывающая как статические, так и динамические элементы сетевой модели;

2) метод интерпретации раскрашенных сетей Петри на основе онтологий и языка SPARQL, позволяющий моделировать динамику сетевой модели как процесс изменения онтологии;

3) архитектура программных средств OntoNet для распределенной интерпретации раскрашенных сетей Петри в среде семантического web;

4) сетевая модель потоковой обработки событий, служащая демонстрационным примером для тестирования инструментальной системы.

Особенностями РСП, используемых в данной работе, являются: наличие позиций, представляющих RDF-хранилища данных, возможность использования нескольких однонаправленных дуг между двумя элементами сетевой модели; разметка переходов и дуг с помощью выражений; приоритетные переходы, а также использование концепции модулей для иерархической структуризации и распределенной интерпретации сетевой модели.

1. Онтология раскрашенных сетей Петри

Онтологическое представление РСП имеет следующие преимущества:

1) возможность «обогащения» синтаксического описания РСП семантической информацией;

2) возможность проведения семантического анализа описаний РСП;

3) упрощение процесса интеграции с другими онтологическими моделями и приложениями семантической сети;

4) возможность использования онтологии РСП как основы для прототипирования и реализации распределенных систем обработки и управления в семантическом web, включая многоагентные системы. Представление онтологии с использованием языков OWL/RDF позволяет использовать язык запросов SPARQL для выборки и модификации данных, что является необходимым условием для моделирования динамики функционирования РСП.

Онтология РСП включает следующие базовые классы: *CPN* (РСП), *Node* (Узел сети), *Arc* (Дуга), *Place* (Позиция), *Transition* (Переход), *Term*

(Выражение), *ColorSet* (Набор цветов), *Variable* (Переменная), *Function* (Функция), *Constant* (Константа), *Multiset* (Мультимножество), *BasisSet* (Базовый набор), *Data* (Значение), *Marking* (Маркировка), *MarkingOfPlace* (Маркировка позиции), *Evaluation* (Вычисление), *Firing* (Срабатывание), *TransitionMode* (Режим перехода), *Binding* (Сопоставление), *Port* (Порт).

Графическое представление TBox-онтологии РСП приведено на рис. 1.

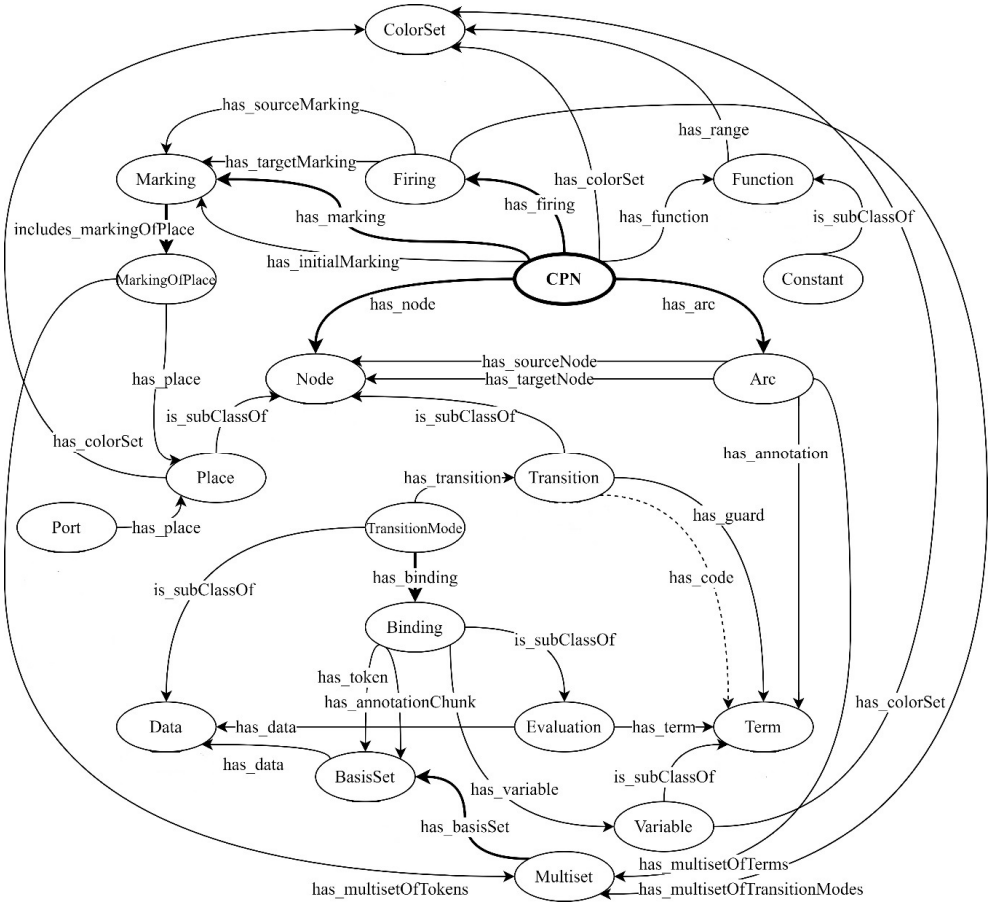

Рис. 1. Онтология РСП на уровне TBox

На рис. 1 классы представлены в виде овалов, а объектные свойства – в виде дуг. Для классов онтологии с использованием дескриптивной логики [12] определены аксиомы классов. Предложенные аксиомы накладывают семантические ограничения на мощность и диапазон отношений, что позволяет проводить семантический анализ описаний.

Класс *CPN* описывает модуль раскрашенной сети Петри. Он определяется следующей аксиомой:

$$CPN \equiv \leq 1.has\_initialMarking.Marking \sqcap$$

$$\sqcap \exists has\_arc.Arc \sqcap \exists has\_colorSet.ColorSet \sqcap$$

$$\sqcap \exists has\_firing.Firing \sqcap \exists has\_function.Function \sqcap$$

$$\Pi\ \exists has\_marking.Marking \Pi\ \exists has\_node.Node\ .$$

Класс *Place* описывает позиции РСП. Он является подклассом *Node* и определяется аксиомой:

$$Place \equiv Node \Pi = 1.has\_colorSet.ColorSet \Pi = 1.has\_initialTokens.Term\ .$$

Класс *Transition* описывает переходы РСП. Он является подклассом *Node* и определяется аксиомой:

$$Transition \equiv Node \Pi \le 1.has\_code.Term \Pi = 1.has\_guard.Term\ .$$

Класс *Arc* представляется следующей аксиомой:

$$Arc \equiv = 1.has\_annotation.Term \Pi\ \le 1.has\_multisetOfTerms.Multiset \Pi$$

$$\Pi = 1.has\_sourceNode.Node \Pi = 1.has\_targetNode.Node\ .$$

Класс *Firing* описывает срабатывание перехода в сети. Представление данного класса в виде аксиомы:

$$Firing \equiv = 1.has\_multisetOfTransitionModes.Multiset \Pi$$

$$\Pi = 1.has\_sourceMarking.Marking \Pi\ = 1.has\_targetMarking.Marking\ .$$

Класс *TransitionMode* описывает режим перехода. Он является подклассом *Data*, поскольку также является значением для базового набора и используется при формировании мультимножеств режимов перехода. Определяющая аксиома для данного класса:

$$TransitionMode \equiv Data \Pi = 1.has\_transition.Transition \Pi$$

$$\Pi\ \exists has\_binding.Binding\ .$$

Класс *Binding* описывает сопоставление переменных и их значений. Он является частным случаем и, как следствие, подклассом *Evaluation*. Его отличие заключается в наличии дополнительных отношений выборки базовых наборов. Данный класс определяется следующей аксиомой:

$$Binding \equiv Evaluation \Pi = 1.has\_annotationChunk.BasisSet \Pi$$

$$\Pi = 1.has\_token.BasisSet \Pi = 1.has\_variable.Variable\ .$$

Класс *Port* описывает порт для межмодульного взаимодействия. Он определяется аксиомой:

$$Port \equiv = 1.has\_place.Place \Pi = 1.has\_type.\{«In» , »Out»\} \Pi$$

$$\Pi \le 1.reserved\_with.xsd : string \Pi\ \exists connected\_to.xsd : string\ .$$

Создание онтологии (в формате *OWL*) осуществлялось в редакторе онтологий *Protégé* [13].

2. Моделирование динамики РСП с использованием SPARQL

В данной работе используется подход к представлению динамики функционирования системы как последовательности изменений онтологиче-

ской модели [14]. Процесс моделирования РСП представляет собой симуляцию срабатываний переходов посредством выполнения цепочек *SPARQL*-запросов (транзакций), в результате чего онтология ABox сетевой модели претерпевает изменения. Фрагменты и описание некоторых запросов, представляющих интерес, приводятся ниже. Стоит отметить, что повторяющиеся и незначительные части запросов заменены поясняющими комментариями в угоду краткости.

Первый запрос используется для инициализации системы. Он позволяет получить все статичные составляющие сети. По результатам данного запроса может выполняться конструирование начальной маркировки. Для этой цели используется конструирующий запрос:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX t: <http://www.onto.net/core/>
PREFIX a: <http://www.onto.net/abox/heads-and-tails/>
WITH <http://localhost:3030/ontonet/data/abox>
INSERT {
 # добавление начальной маркировки, маркировок позиций и мультимножеств токенов}
WHERE {
 GRAPH <http://localhost:3030/ontonet/data/tbox> {
  ?cpn rdf:type t:CPN.}
 BIND(IRI(CONCAT(STR(a:), "marking_", STRUUID())) as ?initial_marking) {
  # привязка UUID позиции
  # формирование UUID маркировок позиций и мультимножеств токенов {
   # формирование базового набора токенов, количества вхождений и значений токенов}
  UNION { # аналогично для другого базового набора токенов... }
  BIND(IRI(CONCAT(STR(a:), "data_", STRUUID())) as ?data)
  BIND("{\"prop\": \"JSON-string\"}" as ?data_value)}
 UNION { # аналогично для другой позиции... }}
```

В дальнейших фрагментах запросов префиксы для краткости будут опущены. Запрос для добавления в сеть всех возможных сопоставлений переменных и значений используется с целью последующей фильтрации этих значений. Это увеличивает гибкость системы, поскольку основной упор остается на формировании немногословных *SPARQL*-запросов.

Запрос для формирования потенциальных режимов перехода используется для агрегации сопоставлений с целью последующей фильтрации.

Запрос для исключения режимов перехода, для которых переход не разрешен, осуществляет фильтрацию режимов перехода по двум признакам:

а) значения переменных должны быть непротиворечивы в рамках одного режима перехода;

б) аннотации входных дуг должны быть способны извлечь зарезервированные токены из позиций, не допуская конфликтов:

```
DELETE {
 # удаление режима перехода}
WHERE {
 { SELECT DISTINCT ?transition_mode
  WHERE {
   { ?transition_mode rdf:type t:TransitionMode.
```

```
      FILTER EXISTS {
       ?transition_mode rdf:type t:TransitionMode;
                        t:has_binding ?binding1;
                        t:has_binding ?binding2.
       FILTER (?binding1 != ?binding2)
       # проверка несовпадения значений одной переменной в режиме перехода
       FILTER (?value1 != ?value2)}}
    UNION
    { # недостаточное количество токенов в позициях
     { SELECT ?transition_mode ((?n + SUM(?diff)) as ?res)
      WHERE {
       { SELECT DISTINCT ?transition_mode ?anno_chunk_bs ?token_bs
        WHERE { # выборка... }}
       ?anno_chunk_bs t:has_multiplicity ?anno_chunk_multiplicity.
       ?token_bs t:has_multiplicity ?token_multiplitity.
       BIND (?anno_chunk_multiplicity as ?k)
       BIND (?token_multiplitity as ?n)
       { SELECT DISTINCT ?anno_chunk_bs (SUM(?n) as ?sum_n)
        WHERE {
         { SELECT DISTINCT ?anno_chunk_bs ?token_bs
          WHERE { # выборка... }}
         ?token_bs t:has_multiplicity ?token_multiplitity.
         BIND (?token_multiplitity as ?n)}
        GROUP BY ?anno_chunk_bs}
       BIND ((?sum_n - ?n - ?k) as ?diff)  # степень выборки токена базовыми наборами аннотации}
      GROUP BY ?transition_mode ?token_bs ?n}
     FILTER (?res < 0) # разница между количеством доступных и выбираемых токенов должна быть >= 0}}}
 { ?transition_mode t:has_binding ?binding. }
 UNION
 { ?transition_mode t:has_transition ?transition. }}
```

Дальнейшие действия движка начинаются с отправки запроса для получения данных о срабатывании:

```
SELECT ?type ?id ?variable_name ?value ?anno_chunk_bs ?token_bs
WHERE {
 # привязка UUID срабатывания
 ?firing t:has_multisetOfTransitionModes ?multiset.
 ?multiset t:has_basisSet ?basis_set.
 ?basis_set t:has_data ?transition_mode.
 {BIND ("multiplicity" as ?type)
  # выборка количества вхождений токенов и шаблонов}
   UNION
  {BIND ("relation" as ?type)
  # получение выборок токенов частями аннотации}
   UNION
  {BIND ("transition" as ?type)
  # получение UUID перехода}
   UNION
  {BIND ("binding" as ?type)
  # получение переменных и значений сопоставлений для режима перехода}}
ORDER BY ?type
```

Обработка срабатывания осуществляется движком по факту встраивания экземпляра срабатывания с некоторым режимом перехода. За извлечение данных о срабатывании аналогично отвечает отдельный запрос.

Запрос для получения комбинаций выборки токенов возвращает все возможные варианты извлечения токенов из входных позиций в данном режиме перехода. Его фрагмент:

```
SELECT <?переменные токенов...>
WHERE {
 BIND (<кол-во вхождений для частей аннотации> as <?переменная части аннотации>)
 # аналогично для других частей аннотации...
 { SELECT  <?переменные токенов...> <?переменные частей аннотации...>
  WHERE {
   { VALUES <?переменная части аннотации для токена> { <значения от 0 до количества вхождений аннотации>}
    BIND ((<?переменные части аннотации для токена+...>) as <?переменная токена>)
    FILTER (<?переменная токена> <= <количество вхождений для токена>)}
   { # аналогично для других токенов... }
   BIND ((<?переменные части аннотации для токенов+...>) as <?переменная части аннотации>)
   # аналогично для других частей аннотации...}}}
```

Запрос для выполнения перехода состоит из трех частей. Две из них не столь значительны и осуществляют очистку онтологии от излишних триплетов. Третья часть определяет стабильные и новые маркировки позиций и базовые наборы токенов и осуществляет добавление новой маркировки, завершая таким образом срабатывание перехода. Запрос позволяет сохранять в новой маркировке сети отношения с нетронутыми маркировками позиций, а в новых маркировках позиций – с нетронутыми базовыми наборами токенов. Это позволяет упростить сравнение маркировок при обходе цепочки срабатываний:

```
INSERT {
 # добавление новой маркировки со стабильными и новыми маркировками позиций
 # добавление новых маркировок позиций и мультимножеств токенов}
WHERE {
 # формирование UUID новой маркировки
 # привязка UUID текущей маркировки
 { GRAPH <http://localhost:3030/ontonet/data/tbox> {?cpn rdf:type t:CPN.}}
 UNION
  {?firing t:has_sourceMarking ?source_marking.
  {?source_marking t:includes_markingOfPlace ?marking_of_place.
   MINUS { # маркировки позиций с извлекаемыми из них токенами }
   MINUS { # маркировки позиций с добавляемыми в них токенами }
   BIND(?marking_of_place as ?stable_marking_of_place)}
    UNION
   {?source_marking t:includes_markingOfPlace ?marking_of_place.
   FILTER( EXISTS { # удаляемые токены }
    || EXISTS { # добавляемые токены } )
   # формирование UUID новой маркировки позиции и нового мультимножества токенов
```

```
    ?marking_of_place t:has_place ?place;
                      t:has_multisetOfTokens ?multiset.
     {?multiset t:has_basisSet ?token_bs.
     MINUS { # удаляемые токены }
     MINUS { # добавляемые токены }
     BIND(?token_bs as ?stable_token_bs)}
      UNION
     {?multiset t:has_basisSet ?token_bs.
     FILTER(EXISTS { # удаляемые токены } || EXISTS { # добавляемые токены })
     # формирование UUID и значений для нового базового набора токенов
     OPTIONAL { # удаляемые токены }
     OPTIONAL { # добавляемые токены }
     BIND ((?token_multiplicity - ?del_multiplicity + ?add_multiplicity) as ?multiplici-
ty)
     FILTER (?multiplicity > 0)}
      UNION
     {# для новых уникальных токенов
     # формирование UUID и значений для нового базового набора токенов
     MINUS { # проверка наличия аналогичных токенов }}}}}
DELETE { # удаление неиспользованных режимов перехода }
WHERE { # выборка свободных режимов перехода };
DELETE { # удаление неиспользованных сопоставлений }
WHERE { # выборка свободных сопоставлений }
```

3. Распределенная интерпретации РСП

Для организации распределенной интерпретации РСП могут использоваться следующие средства:

1) распределенное хранение данных (ресурсов);

2) распределенная обработка федеративных *SPARQL*-запросов;

3) распределение модулей РСП по узлам вычислительной сети.

Третий вариант выглядит наиболее предпочтительным, так как позволяет реализовывать микросервисы и дает возможность вести параллельную обработку модулей.

Существует два подхода к организации взаимосвязи модулей:

1) взаимодействие путем обмена сообщениями через интерфейсы модулей;

2) разделение общего ресурса (например, позиций сетевой модели).

Второй подход сложен в плане синхронизации, поэтому был использован только первый подход. При этом допускается связь «один ко многим» для модулей.

При срабатывании переходов и изменении позиций, являющихся выходными портами, отрабатывается следующий алгоритм синхронизации. На узле генерируется уникальная последовательность (хеш), с помощью которой резервируются локальные порты, используемые при переходе. Выполняется попытка занятия используемых портов на удаленных узлах. Хеш при этом используется для определения приоритета узла в случае одновременного выполнения переходов, затрагивающих контактные позиции. Если заняты удаленные порты, занимаются и локальные порты, после чего выполняется переход и порты освобождаются. Такой алгоритм особенно подходит для иерархических сетей, позволяя определить и выполнить приоритетный переход.

На рис. 2 предлагается структура интерпретатора модуля распределенной РСП названного OntoNet.

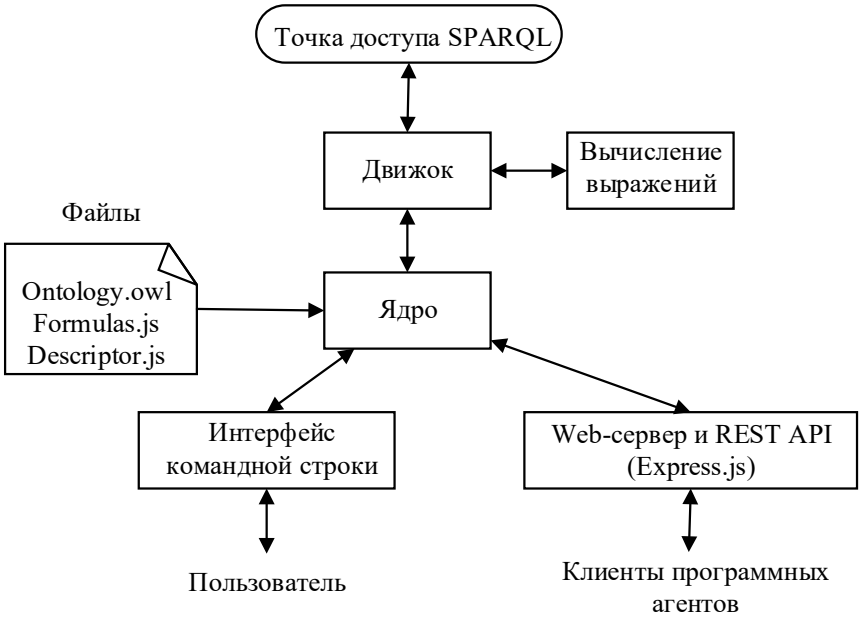

Рис. 2. Структура интерпретатора OntoNet

Данный интерпретатор включает следующие программные компоненты: движок, модуль расчета выражений, интерфейс командной строки, вебсервер и связующее ядро. Система написана на *TypeScript* и предлагает возможность обработки разметки сети с помощью выражений на *JavaScript*.

Взаимодействие осуществляется через интерфейс командной строки, который позволяет конфигурировать систему путем подгрузки файлов *formulas.js* и *descriptor.js*, установить соединение со *SPARQL*-сервером по *URL*, загрузить ABox-онтологию сетевого модуля и наблюдать за процессом его работы.

В онтологии ABox модуля РСП описываются входные и выходные порты для организации межузлового взаимодействия. Порты инициализируются URL связанных узлов и типом (*input/output*). Онтология TBox при этом может выноситься на отдельный сервер. Однако допускается и хранение собственных экземпляров TBox-онтологии. В таком случае функциональные возможности благодаря обратной совместимости будут ограничены лишь используемой версией.

Срабатывания в распределенных РСП хранятся на каждом узле в отдельности и отражают действия в рамках их внутренних процессов, а также внешних воздействий связанных узлов. При этом общие выражения и константы могут выноситься в TBox-онтологию в виде исключения, если это предпочтительно. Каждый модуль отвечает за срабатывание только своих переходов. При формировании меток в позиции, являющейся еще и выходным портом, модуль пересылает связанному узлу SPARQL-запрос на формирование, который аналогичен выполненному внутри самого модуля. Межмо-

дульное взаимодействие сводится к передаче меток через порты. Решение о выполнении перехода принимается клиентской программой.

С помощью веб-сервера организуется взаимодействие с клиентскими приложениями и другими модулями в сети. Модуль расчета выражений используется для вычисления значений аннотаций дуг, сторожевых условий и формируемых токенов. Движок взаимодействует непосредственно с оконечной точкой доступа *SPARQL*. В нем реализована вся логика по инициализации модели и срабатыванию переходов, однако вычисление выражений делегируется модулю расчета.

Приложение спроектировано с использованием паттерна *Observer*, что позволяет уменьшить связность компонентов и в дальнейшем с легкостью расширять и улучшать систему, например, заменить интерфейс командной строки на полноценное приложение с графическим интерфейсом.

4. Пример. Сетевая модель потоковой обработки событий

В качестве иллюстративного примера предлагается модель потоковой обработки событий, когда каждое событие связано с данными. В системе имеется два независимых канала поступления информации: событийный канал, по которому поступают события, и канал данных, по которому поступают данные. Независимость этих двух видов каналов обусловливается логикой работы системы. Аналогом является разделение событий и данных в функциональных блоках (ФБ) стандарта IEC 61499 [15]. Взаимосвязь модулей сетевой модели показана на рис. 3.

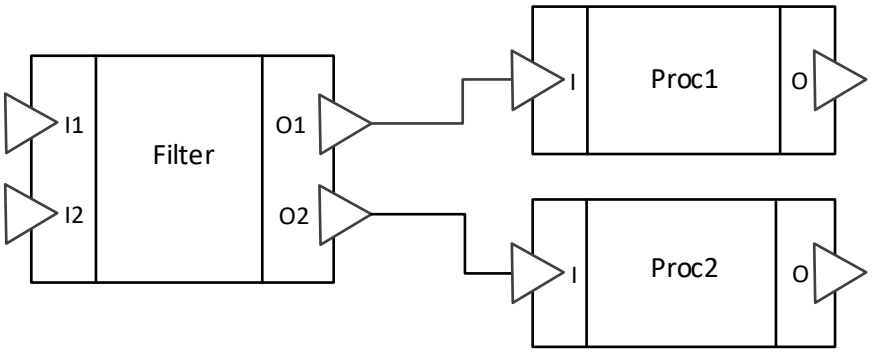

Рис. 3. Взаимосвязь модулей сетевой модели

Модуль Filter осуществляет фильтрацию поступающих из канала I1 событий по связанным с ними данным, поступающим из канала I2. События разного класса направляются на выходы O1 и O2 соответственно. Модули Proc1 и Proc2 осуществляют обработку данных, связанных с входными событиями, определяют имя выходного события и значения результирующих данных, связанных с этим событием. Полученные данные подаются на выходы O.

Модуль Filter приведен на рис. 4. Следует отметить, что в иллюстративных целях на данном рисунке представлена маркировка позиций. Событийные токены через входной порт I1 поступают в позицию EVENT. Формат событийных токенов следующий:

$$\left[\left\langle Id\ событийного\ входа\right\rangle,\left\langle Id\ информационного\ входа\right\rangle\right].$$

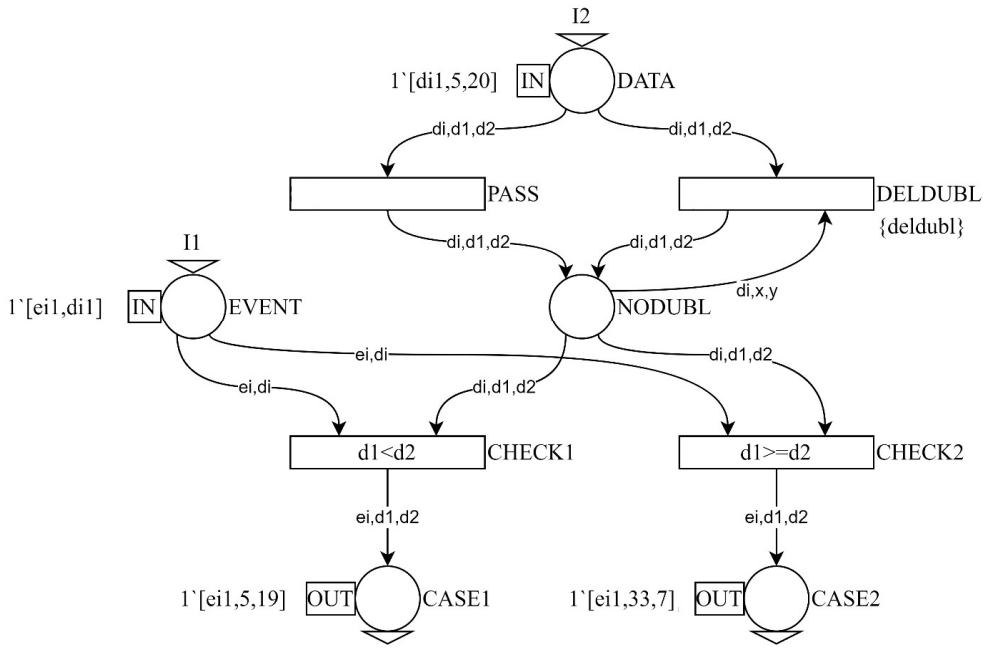

Рис. 4. Модуль *Filter*

Таким образом осуществляется динамическая привязка информационного входа к событийному.

Токены, представляющие данные, через входной порт I2 поступают в позицию DATA. Их формат следующий:

$$\left[\left\langle Id\ информационного\ входа\right\rangle,\left\langle Данные1\right\rangle,\left\langle Данные2\right\rangle\right].$$

С помощью переходов DELDUBL и PASS организуется обновление данных, при котором в позицию NODUBL попадают только новые токены, перекрывая старые. Следует учесть, что приоритет перехода DELDUBL должен быть выше приоритета перехода PASS.

За сортировку событий отвечают переходы CHECK1 и CHECK2, которым назначены соответствующие сторожевые условия. В них также осуществляется привязка данных к выходным событиям. В результате в выходную позицию CASE1 попадают токены, для которых оказалось справедливым отношение:

$$Данные1 > Данные2.$$

В противном случае токены помещаются в позицию CASE2.

Модуль Proc1 приведен на рис. 5. Токен, содержащий идентификатор события и данные, поступает сначала в позицию INPUTS. В переходе PROCESSING производится обработка данных, которая заключается в определении суммы первого и второго значений данных. В позиции RESULT формируется новый токен, содержащий все тот же идентификатор события и результирующие данные.

Позиция RDF имеет особое значение. Она представляет собой *RDF*-хранилище данных о связях между событийными входами и выходами.

Структура данных представляет собой триплет следующего вида: $[ei, "conn", eo]$, где *ei* и *eo* – идентификаторы событийных входов и выходов; *conn* – строковая константа. В сетевой модели может быть несколько подобных *RDF*-хранилищ. После нахождения соответствия «вход-выход» формируется токен, содержащий идентификатор событийного выхода и значение связанных с ним данных. Модуль Proc2 в данной работе не рассматривается.

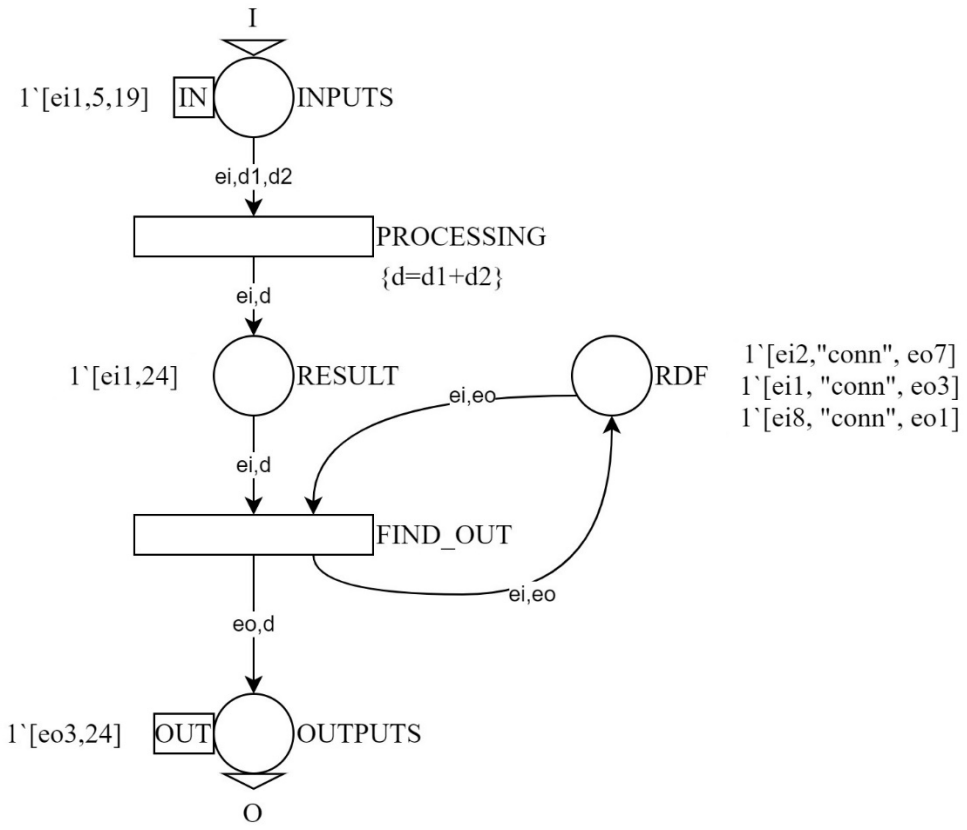

Рис. 5. Модуль *Proc1*

Структура системы распределенной обработки на основе вышеприведенной сетевой модели состоит из трех сетевых узлов, каждый из которых включает систему OntoNet, SPQRQL-сервер и онтологическое описание соответствующего модуля РСП. Следует отметить, что модули могут оперироваться не только в сетевом окружении, но и одним мультизадачным клиентским приложением.

Заключение

Предложенные методы и средства онтологического моделирования и интерпретации раскрашенных сетей Петри могут быть использованы для быстрой разработки, прототипирования и реализации распределенных, параллельных и многоагентных систем обработки событий, данных и знаний в контексте семантического web.

Направлением дальнейших исследований является более глубокая специализация раскрашенных сетей Петри под особенности семантического web, задание начальной маркировки сети в виде SPARQL-запроса, расширение моделирующих возможностей сетевой модели за счет структурной самомодификации и использования онтологий и SPARQL-запросов в качестве значений (цветов) токенов, а также адаптация разработанных средств под конкретные приложения обработки событий, данных и знаний из практической сферы.

Список литературы

1. Jensen K., Kristensen L. M. Coloured Petri Nets. Modelling and Validation of Concurrent System. Springer-Verlag, 2009. 395 p.
2. Renew. The Reference Net Workshop. URL: http://www.renew.de/ (дата обращения: 05.02.2022)
3. Szeredi P., Lukácsy G., Benkő T. The Semantic Web Explained: The Technology and Mathematics behind Web 3.0. Cambridge : Cambridge University Press, 2014. 478 p.
4. Gruber T. R. A Translation Approach to Portable Ontology Specifications // Knowledge Acquisition. 1993. № 5(2). P. 199–220. doi:10.1006/knac.1993.1008.
5. Keet C. M. An Introduction to Ontology Engineering. College Publications, 2018. 344 p.
6. DuCharme B. Learning SPARQL: Querying and Updating with SPARQL 1.1. O'Reilly, 2013. 386 p.
7. Gašević D., Devedžić V. Interoperable Petri Net Models via Ontology // International Journal of Web Engineering and Technology. 2007. Vol. 3, iss. 4. P. 374–396. doi:10.1504/IJWET.2007.014439.
8. Zhang F., Ma Z. M., Ribarić S. Representation of Petri Net with OWL DL Ontology // Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE Publisher, 2011. P. 1452–1456.
9. Vidal J. C., Lama M., Bugarín A. A High-level Petri Net Ontology Compatible with PNML // Petri Net Newsletter. 2006. Vol. 71. P. 11–23.
10. Vidal J. C., Lama M., Bugarín A. OPENET: Ontology-based engine for high-level Petri nets // Expert Systems with Applications. 2010. Vol. 37, iss. 9. P. 6493–6509. doi:10.1016/j.eswa.2010.02.136.
11. Vidal J. C., Lama M., Sanchez E., Bugarín A., Novegil A. OPENET LD: An Ontology-based Petri Net Engine to Execute IMS LD Units of Learning // 9th IEEE International Conference on Advanced Learning Technologies (ICALT). IEEE Publisher, 2009. P. 499–503.
12. McGuinness D. L., Nardi D., Patel-Schneider P. F. The description logic handbook: Theory, implementation, and applications. Cambridge : Cambridge University Press, 2010. 578 p.
13. Protégé. A free, open-source ontology editor and framework for building intelligent systems. URL: https://protege.stanford.edu/ (дата обращения: 05.02.2022).
14. Дубинин В. Н., Дубинин А. В., Янг Ч.-В., Вяткин В. В. Использование языка SPARQL в онтологическом моделировании мультиагентных систем в семантическом Web // Известия высших учебных заведений. Поволжский регион. Технические науки. 2020. № 1. С. 4–18.
15. Vyatkin V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design. ISA, 2015. 261 p.

References

1. Jensen K., Kristensen L.M. *Coloured Petri Nets. Modelling and Validation of Concurrent System*. Springer-Verlag, 2009:395.

2. *Renew. The Reference Net Workshop*. Available at: http://www.renew.de/ (accessed 05.02.2022)
3. Szeredi P., Lukácsy G., Benkő T. *The Semantic Web Explained: The Technology and Mathematics behind Web 3.0*. Cambridge: Cambridge University Press, 2014:478.
4. Gruber T.R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*. 1993;(5):199–220. doi:10.1006/knac.1993.1008.
5. Keet C.M. *An Introduction to Ontology Engineering*. College Publications, 2018:344.
6. DuCharme B. *Learning SPARQL: Querying and Updating with SPARQL 1.1*. O'Reilly, 2013:386.
7. Gašević D., Devedžić V. Interoperable Petri Net Models via Ontology. *International Journal of Web Engineering and Technology*. 2007;3(4):374–396. doi:10.1504/IJWET.2007.014439
8. Zhang F., Ma Z.M., Ribarić S. Representation of Petri Net with OWL DL Ontology. *Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE Publisher, 2011:1452–1456.
9. Vidal J.C., Lama M., Bugarín A. A High-level Petri Net Ontology Compatible with PNML. *Petri Net Newsletter*. 2006;71:11–23.
10. Vidal J.C., Lama M., Bugarín A. OPENET: Ontology-based engine for high-level Petri nets. *Expert Systems with Applications*. 2010;37(9):6493–6509. doi:10.1016/j.eswa.2010.02.136.
11. Vidal J.C., Lama M., Sanchez E., Bugarín A., Novegil A. OPENET LD: An Ontology-based Petri Net Engine to Execute IMS LD Units of Learning. *9th IEEE International Conference on Advanced Learning Technologies (ICALT)*. IEEE Publisher, 2009:499–503.
12. McGuinness D.L., Nardi D., Patel-Schneider P.F. *The description logic handbook: Theory, implementation, and applications*. Cambridge: Cambridge University Press, 2010:578.
13. Protégé. *A free, open-source ontology editor and framework for building intelligent systems*. Available at: https://protege.stanford.edu/ (accessed 05.02.2022).
14. Dubinin V.N., Dubinin A.V., Yang Ch.-V., Vyatkin V.V. Using the SPARQL language in ontological modeling of multi-agent systems in the Semantic Web. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2020;(1):4–18. (In Russ.)
15. Vyatkin V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*. ISA, 2015:261.

Информация об авторах / Information about the authors

Владислав Юрьевич Каев
студент, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: veld4n@gmail.com

Vladislav Yu. Kaev
Student, Penza State University (40 Krasnaya street, Penza, Russia)

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

E-mail: dubinin.victor@gmail.com

Viktor N. Dubinin
Doctor of engineering sciences, associate professor, professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

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

E-mail: dubinin.aleksey@gmail.com

Aleksey V. Dubinin
Student, Penza State University (40 Krasnaya street, Penza, Russia)

Людмила Петровна Климкина
старший преподаватель кафедры финансов и информатизации бизнеса, Пензенский государственный аграрный университет (Россия, г. Пенза, ул. Ботаническая, 30)

E-mail: ludmila.klimkina@gmail.com

Lyudmila P. Klimkina
Senior lecturer of the sub-department of finance and business informatization, Penza State Agricultural University (30 Botanicheskaya street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию / Received 17.01.2022

Поступила после рецензирования и доработки / Revised 06.02.2022

Принята к публикации / Accepted 28.02.2022