ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 4(8), 2011, с. 85–94

УДК 004.825:004.912

И. В. Трофимов

Эволюция выразительных способностей языка OWL

Аннотация. Рассматриваются языковые конструкции диалектов языка OWL, как средства спецификации предметных онтологий. Затрагивается проблема представления n-арных предметных отношений. Анализируется характер и значимость нововведений в синтаксисе.

Ключевые слова и фразы: OWL, RDF Schema, Semantic Web, выразительные возможности, моделирование понятий, проблема непротиворечивости, семантический анализ.

Введение

Одним из способов представления знаний, получивших в последние годы наибольшее распространение, является язык OWL (web ontology language). OWL разработан консорциумом W3C в рамках инициативы Semantic Web. В 2004 году получил статус рекомендации [1–3]. OWL был создан в ответ на потребность в стандартизации способов представления знаний в Веб. Инициаторами стандартизации выступили авторы разработанных ранее онтологических языков DAML (DARPA Agent Markup Language) и OIL (Ontology Inference Layer).

В основе OWL лежит язык RDFS (RDF Schema) [4]. Выразительных способностей RDFS достаточно для описания классов и свойств RDF-ресурсов, а также иерархий на них. Тем не менее, OWL предоставляет дополнительные возможности: определение класса через ограничения на его свойства, через теоретико-множественные операции над классами, определение характеристик (симметричность, транзитивность и других) и кардинальных чисел свойств.

По выразительным способностям в рамках OWL выделяют три диалекта:

Работа проводилась при финансовой поддержке Министерства образования и науки Российской Федерации.

- OWL Full обладает максимумом выразительных средств, но не гарантирует, что логический вывод в такой онтологии будет вычислимым. В частности в Full-диалекте классы могут одновременно выступать и в роли класса, и в роли экземпляра.
- OWL DL гарантирует вычислительную полноту (логический вывод является вычислимым) и разрешимость (вычисления выполняются за конечное время). OWL DL содержит все языковые конструкции OWL, но их использование ограничено. DL отражает связь этого языка с дескриптивной логикой (разрешимой частью логики предикатов первого порядка).
- OWL Lite обладает наименьшими выразительными способностями, но может быть промежуточным звеном при переходе от простых таксономий к онтологиям.

К сильным сторонам OWL можно отнести ориентированность на независимую распределенную разработку онтологии. Синтаксис языка таков, что любой класс, экземпляр или свойство можно доопределить независимо от того, как они были определены изначально. Процесс доопределения не требует какого-либо согласования с автором исходного определения и может осуществляться без изменения документа, где зафиксировано исходное определение (т.е. в отдельном документе). Таким образом, знания об экземплярах, классах и свойствах могут накапливаться и уточняться постепенно, с участием большого числа людей.

Слабой стороной OWL является то, что он не дает ответа на вопрос, как избежать добавления в онтологию противоречивых утверждений и что делать, если противоречия возникнут. Для решения проблемы непротиворечивости приходится создавать специальные диагностические методы и средства [5–7].

Рассмотрим подробнее базовые языковые конструкции OWL.

0.1. Классы

Классы используются для моделирования понятий (концептов).

Согласно OWL-определению, класс — это абстрагирующий механизм для группировки ресурсов со схожими характеристиками. Каждый OWL-класс ассоциируется с множеством экземпляров (индивидов), называемых экстенсионалом класса. В то же время, класс имеет собственный особый смысл (лежащее в основе понятие), который связан, но не равен экстенсионалу класса. Два класса могут иметь один и тот же экстенсионал, но при этом оставаться разными классами.

В OWL имеется несколько способов определения класса:

- использование идентификатора класса,
- полное перечисление экземпляров, в совокупности составляющих класс,
- посредством ограничения свойства,
- через теоретико-множественные операции над другими классами.

Первый способ декларирует существование некоторого класса, назначая ему имя (идентификатор). Таким образом, в онтологию вводится новое понятие (и стоящий за ним смысл), но его свойства и связь с экстенсионалом остается неопределенной. Синтаксически это конструкция следующего вида:

```
<owl:Class rdf:ID="ИмяКласса"/>,
```

что соответствует RDF-триплету `<ИмяКласса, rdf:type, owl:Class>`. Остальные способы определяют класс через экстенсионал.

0.2. Иерархии классов

Между классами можно устанавливать иерархические отношения типа «общее-частное». Для этого используется конструкция `rdfs:subClassOf`. Например, определение класса «город» и указание на то, что город является частным случаем населенного пункта, выглядит следующим образом:

```
<owl:Class rdf:ID="город">
          <rdfs:subClassOf rdf:resource="#населенный_пункт"/>.
</owl:Class>
```

Согласно семантике OWL, это отношение имеет теоретико-множественную трактовку: экстенсионал подкласса полностью входит в экстенсионал надкласса.

0.3. Свойства

Свойства служат для моделирования атрибутивной информации, характеризующей членов классов, а также для моделирования отношений между ними. Например, свойство «возраст» может использоваться в качестве атрибута для экземпляров класса «человек», а свойство «предок» — в качестве отношения между двумя экземплярами

класса «человек». В терминологии OWL свойства, моделирующие атрибутивную информацию, называются свойствами-значениями (datatype properties), а моделирующие отношения — объектными свойствами (object properties).

Простейший способ определить свойство — это объявить о его существовании (задать идентификатор):

```
<owl:ObjectProperty rdf:ID="произведено_в"/>.
```

Свойство является бинарным отношением. OWL обладает средствами для указания ограничений на области определения и значений этого отношения. Это могут быть как безусловные (глобальные), так и условные (зависящие от того, экземпляр какого класса характеризуется свойством) ограничения. Для безусловных ограничений используются конструкции `rdfs:domain` и `rdfs:range`, для условных — `owl:restriction`.

Свойства одновременно используются и для декларации обобщенных утверждений о членах классов, и для спецификации утверждений о конкретных экземплярах. Примером обобщенного утверждения может служить запись

```
<owl:ObjectProperty rdf:ID="находится_в">
          <rdfs:domain rdf:resource="#город"/>
          <rdfs:range  rdf:resource="#государство"/>
</owl:ObjectProperty>,
```

которая постулирует, что города могут находиться в государствах, или более точно, экземпляр класса «город» может быть связан отношением «находится в» с экземпляром класса «государство». В качестве соответствующего конкретного утверждения можно привести

```
<город rdf:ID="Нью-Йорк">
           <находится_в rdf:resource="#США"/>
</город>.
```

Кроме перечисленного, язык OWL позволяет определять:

- иерархию свойств;
- логические характеристики свойств (симметричность, транзитивность);
- связь с другим свойством (обратное свойство);
- безусловные и условные кардинальные числа, определяющие в каких пропорциях могут соотноситься экземпляры, связанные данным свойством.

0.4. Экземпляры (индивиды)

Кроме классов, мы можем рассуждать об их членах — экземплярах. Что моделировать при помощи классов, а что при помощи экземпляров — важный вопрос онтологического моделирования. Считается, что в этом вопросе на первый план выдвигаются утилитарные соображения: каким образом будет использоваться данная спецификация предметной области. В руководстве по OWL [3] об уровнях представления пишут следующее.

«В определенных контекстах, то, что очевидно является классом, может рассматриваться как экземпляр чего-то еще. Например, в онтологии вин у нас есть понятие 'виноград', которое предназначено для обозначения всех *сортов винограда*. 'Виноград каберне-совиньон' является экземпляром этого класса, т.к. он обозначает сорт винограда под названием каберне-совиньон. Однако 'Виноград каберне-совиньон' в свою очередь может рассматриваться, как класс — множество всех ягод винограда каберне-совиньон.»

В диалекте OWL Full эта проблема стоит не так остро, так как этот диалект позволяет сущности являться одновременно и классом, и экземпляром.

0.5. Свойства свойств

Одним из существенных недостатков OWL является отсутствие возможности естественным образом определять свойства у свойств. Это не позволяет моделировать атрибуты у предметных отношений, n-арные отношения и атрибуты у атрибутов. Пусть, например, в нашей онтологии зафиксирован факт встречи двух людей.

```
<человек rdf:ID=" Иванов">
            <встретился_с rdf:resource="#Смирнов"/>
</человек>.
```

Если мы теперь захотим зафиксировать дату встречи, то синтаксис OWL не позволит нам это сделать.

Существует два пути обхода этой проблемы. Один из них — выйти за рамки OWL и прибегнуть к механизму реификации (от англ. reify —материализовывать). Реификация является средством RDF [?8]. Ее суть состоит в том, чтобы рассматривать RDF-утверждения как отдельные объекты. В RDF утверждением считается триплет вида `<объект, предикат, субъект>`. Данный триплет может быть определен как

экземпляр класса «Утверждение» (`rdf:Statement`) и сам использоваться как объект в других утверждениях. Это позволяет делать утверждения об утверждениях. Реификация позволяет, например, специфицировать, кто является автором утверждения, когда было сделано утверждение и т.п.

Вернемся к примеру о встрече. Факт `<Иванов, встретился_с, Смирнов>` можно объявить как экземпляр класса `rdf:Statement` и далее использовать его как отдельный объект в других утверждениях, и в частности, ассоциировать с ним свойство *дата_встречи*.

Альтернативный путь — декларировать отношения как классы, а не как свойства [?9]. Авторы этого решения критикуют использование реификации для определения свойств у свойств, приводя следующие доводы. «В n-местных отношениях дополнительные аргументы, обычно, характеризуют не само утверждение (statement), а экземпляр отношения». Поэтому было бы более естественным оперировать непосредственно экземплярами отношений, нежели утверждениями об этих экземплярах.

Суть второго подхода в следующем. Отношение определяется как класс, а экземпляры этого класса будут выступать в роли экземпляров отношения. В свою очередь, аргументы этого отношения связываются с ним при помощи механизма свойств и их количество уже не ограничено.

Пусть, например, у нас имеются следующие предложения, которые мы хотим формализовать в рамках онтологии:

- С большой долей вероятности у Ивана ангина.
- Андрей приобрел книгу «О грибах» в магазине mush-book.com за 200 руб. в качестве подарка на день рождения.

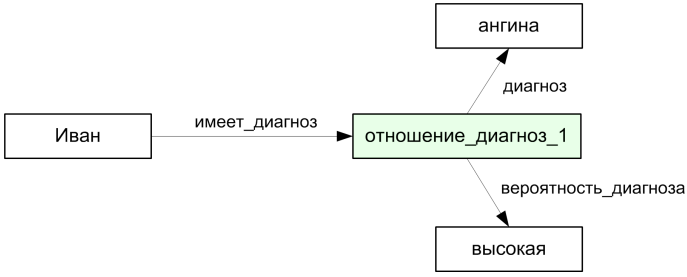

Рис. 1. Свойства и отношения классов выделены

Определяя отношения как классы, мы могли бы формализовать эти предложения следующим образом. Вводится класс *Отношение-диагноз* и строится его экземпляр *отношение_ диагноз_ 1*. Свойствами этого экземпляра будут *диагноз* и *вероятность_ диагноза*. Иван будет обладать свойством *имеет_ диагноз*, которое связывает Ивана с экземпляром *отношение_ диагноз_ 1* (см. рис. 1).

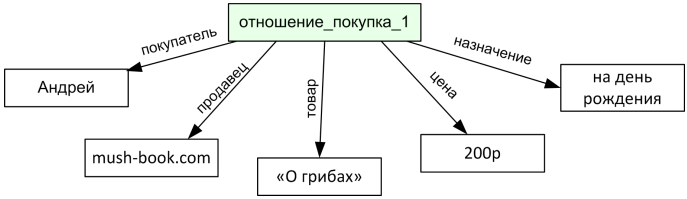

Рис. 2. Свойства и отношения классов выделяются неоднозначно

Если затруднительно выделить какой-либо аргумент отношения как объект или субъект, можно всех участников отношения связать с ним через свойства (см. рис. 2).

Здесь *покупатель, продавец, товар, цена* и *назначение* — свойства экземпляра отношения *отношение_ покупка_ 1*.

Отметим, что прием с определением отношения при помощи класса не является стандартным подходом (на него не распространяется рекомендация W3C). Это всего лишь один из способов преодолеть недостаточную выразительную способность OWL.

0.6. Эквивалентность и несовместимость

Кроме перечисленных средств представления знаний в OWL имеется набор конструкций, позволяющих устанавливать эквивалентности и несовместимости между классами, свойствами и экземплярами. Они в какой-то мере позволяют преодолеть сложности, порожденные принципом независимой распределенной разработки онтологии.

- owl:equivalentClass определяет эквивалентность пары классов. Если C1 и C2 — эквивалентные классы, то из «X является экземпляром C1» следует, что «X является экземпляром C2» и наоборот.
- owl:equivalentProperty определяет эквивалентность пары свойств. Если P1 и P2 — эквивалентные свойства, то из «X связан с Y отношением P1» следует, что «X связан с Y отношением P2» и наоборот.

- owl:disjointWith определяет несовместимость классов. Если C1 и C2 — несовместимые классы, то из «X является экземпляром C1» следует, что «X не является экземпляром C2». Примером несовместимых классов может служить пара «мужчина» и «женщина».
- owl:sameAs определяет, что два имени экземпляра в онтологии ссылаются на один и тот же экземпляр из предметной области.
- owl:differentFrom определяет, что два имени экземпляра в онтологии ссылаются на разные экземпляры из предметной области. Для одновременного определения группы взаимно различных экземпляров можно использовать конструкцию owl:AllDifferent.

1. OWL 2

Получив статус рекомендации W3C, язык OWL стал активно использоваться в основанных на знания программных продуктах и исследовательских проектах. Системы автоматического логического вывода [?10, ?11] и редакторы онтологий [8] стали ориентироваться на OWL. Практика позволила выявить ограниченность выразительных способностей OWL и недостатки технического характера (сложность синтаксического разбора, невозможность обнаружить опечатки в именах) [9]. Это привело к созданию новой версии языка — OWL 2 [10].

Рассмотрим нововведения OWL 2 [11], связанные с расширением выразительных способностей.

OWL 1 позволял определять свойства как симметричные и транзитивные. В OWL 2 спектр логических характеристик свойств расширен рефлексивностью, антирефлексивностью и антисимметричностью. Также появилась возможность декларировать «локальную рефлексивность», когда свойство рефлексивностью не характеризуется, но для некоторых классов объектов рефлексивность присутствует.

В OWL 2 были добавлены уточненные ограничения кардинальности (qualified cardinality restrictions). OWL 1 предоставлял лишь возможность ограничивать количество экземпляров, связанных свойством. Например, определить класс людей, у которых, по меньшей мере, трое детей. Уточненные ограничения кардинальности позволяют также ограничить класс (или диапазон данных для атрибутивных свойств) экземпляров, которые ограничиваются количественно. Например, определить класс людей, у которых, по меньшей мере, трое детей и все они являются мальчиками.

OWL 1 позволял объявлять классы как несовместимые. В OWL 2 появилась возможность делать то же самое для свойств. Объявление некоторого множества свойств несовместимыми означает, что два экземпляра не могут быть соединены более чем одним свойством из этого множества. В качестве примера можно привести пару свойств «находиться над» и «находиться внутри».

Еще одна новая возможность OWL 2 — определение свойств через композицию других свойств (property chain inclusion). Это позволяет определять, например, такие аксиомы: «X находится в Z, если X находится в Y и Y является частью Z».

В OWL 2 вводится конструкция для определения ключа — множества свойств, позволяющих уникально идентифицировать экземпляры заданного класса. Также расширен перечень типов данных для атрибутивных свойств и предоставлена возможность задавать некоторые собственные ограничения на диапазоны значений.

Остальные нововведения носят инструментальный характер и служат для удобства разработки приложений, работающих с онтологиями. Среди них хочется выделить возможность использования IRI (Internationalized Resource Identifiers) вместо URI (Uniform Resource Identifiers). Это позволяет присваивать классам, свойствам и другим элементам онтологии имена на произвольных языках, а не только на английском.

Список литературы

[1] OWL Web Ontology Language.Overview. W3C Recommendation : W3C, 27 October 2009, URL: http://www.w3.org/TR/owl-features/. ↑[]

[2] OWL Web Ontology Language Guide. W3C Recommendation : W3C, 10 February 2004, URL: http://www.w3.org/TR/owl-guide/. ↑

[3] OWL Web Ontology Language. Reference. W3C Recommendation : W3C, 10 February 2004, URL: http://www.w3.org/TR/owl-ref/. ↑[], 0.4

[4] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation : W3C, 10 February 2004, URL: http://www.w3.org/TR/rdf-schema/. ↑[]

[5] Kalyanpur A. *Debugging and Repair of OWL Ontologies*, PhD thesis, The Graduate School of the University of Maryland, 2006 ↑[]

[6] Lam S.C. J. *Methods for Resolving Inconsistencies In Ontologies*, PhD thesis, Department of Computer Science, Aberdeen, 2007 ↑

[7] Deng X., Haarslev V., Shiri N. *Measuring Inconsistencies in Ontologies* // Proceeding ESWC '07, 2007 ↑[]

[8] Knublauch H., Fergerson R. W., Noy N. F., Musen M. A. *The Protege OWL plugin: An open development environment for semantic web applications* // Proc. ISWC-04, 2004, p. 229–243 ↑1

[9] Grau B.C., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U. *OWL 2: The Next Step for OWL* // Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 2008. **6**, no. 4, p. 309–322 ↑1

[10] OWL 2 Web Ontology Language. Document Overview. W3C Recommendation : W3C, 27 October 2009, URL: http://www.w3.org/TR/owl2-overview/. ↑1

[11] OWL 2 Web Ontology Language. New Features and Rationale. W3C Recommendation : W3C, 27 October 2009, URL: http://www.w3.org/TR/owl2-new-features/. ↑1

I. V. Trofimov. *The evolution of the OWL language power.*

Abstract. Language constructs of OWL dialects are considered as a tools for subject specification ontology. The issue of representation n-ary object relationships is addressed. The nature and significance of the syntax innovations is under consideration.

Key Words and Phrases: OWL, RDF Schema, Semantic Web, expressive possibilities, modeling concepts, the consistency problem, semantic analysis.

Образец ссылки на статью:

И. В. Трофимов. *Эволюция выразительных способностей языка OWL* // Программные системы: теория и приложения : электрон. научн. журн. 2011. № 4(8), с. 85–94. URL: http://psta.psiras.ru/read/psta2011_4_85-94.pdf