МЕТОД СТРУКТУРНОГО СРАВНЕНИЯ ОНТОЛОГИЙ С ИСПОЛЬЗОВАНИЕМ ПРЯМОЙ СЕМАНТИКИ ЯЗЫКА OWL 2

И. А. Заикин

Институт кибернетики Национального исследовательского
Томского политехнического университета, 634050, Томск, Россия

УДК 004.42; 004.82

Предложен алгоритм структурного сравнения онтологий, использующий прямую семантику языка OWL 2 и учитывающий изменения не только в логической составляющей, но и в дополнительных атрибутах, содержащих сведения о связях с другими онтологиями, используемых пространствах имен, идентификаторах онтологии и формате ее представления. Разработана программная реализация алгоритма в виде инструмента командной строки.

Ключевые слова: OWL 2, RDF, прямая семантика, онтология, сущность, утверждение, формат онтологии, префикс пространства имен.

In this paper, we present an algorithm of structural comparison of ontologies which, firstly, uses OWL 2 direct semantics, and, secondly, takes into account changes not only in logical constituent but also in additional attributes containing information about links to other ontologies, used namespaces, ontology identifiers and its serialization format. An implementation of this algorithm has been created in form of a command-line tool.

Key words: OWL 2, RDF, direct semantics, ontology, entity, statement, ontology format, namespace prefix.

Онтология – формальное описание определенной предметной области на стандартизированном языке, понятном как людям, так и вычислительным машинам. В настоящее время среди таких языков наиболее распространенным является язык OWL 2 [1]. Разработка сложных онтологий требует участия специалистов по языку онтологий, специалистов в предметной области, а также специалистов по управлению качеством. Совместная разработка сложных онтологий практически невозможна без использования инструментальных средств поддержки групповой работы и, в частности, систем управления версиями, которые позволяют сохранять журнал изменений, в случае необходимости отменять нежелательные изменения, сравнивать версии, объединять изменения разных членов команды и облегчают разрешение конфликтов. Один из важнейших компонентов таких систем − программа для сравнения онтологий, результатом работы которой является набор изменений между двумя версиями. Обычные программы для сравнения текстов основаны на предположении, что порядок строк в тексте имеет значение, в то время как язык OWL 2 не накладывает ограничений на порядок следования аксиом. К тому же одна и та же онтология может быть сохранена в различных форматах, что не позволяет использовать для сравнения онтологий обычные программы сравнения текстов.

Язык OWL 2 имеет две семантики: 1) исторически сложившуюся, основанную на RDF (resource description framework), основными понятиями которой являются ресурсы и отношения между ними, а сложные конструкции представлены с использованием так называемых пустых узлов; 2) прямую семантику, в которой основными понятиями являются сущность и аксиома. Существуют работы, в которых проводится сравнение RDF-графов, однако в них либо не упоминается о проблеме пустых узлов, либо разработанные алгоритмы очень сложны и ненадежны, что не позволяет применять их на практике. В работе [2] предпринята попытка идентифицировать пустые узлы с помощью связанных с ними ребер. В [3] показано, что в общем случае однозначное сравнение триплетов, содержащих пустые узлы, невозможно, хотя существуют вычислительно сложные алгоритмы проверки изоморфизма графов. В данной работе предложен алгоритм структурного сравнения онтологий с использованием прямой семантики OWL 2. Использование прямой семантики OWL 2 позволяет избежать сложного анализа графов, содержащих пустые узлы, и выполнять сравнение поаксиомно. Также предложенный алгоритм учитывает атрибуты онтологий, не входящие в логическую составляющую, об изменении которых тем не менее важно знать при разработке онтологий.

Постановка задачи. Под онтологией в данной работе понимается совокупность $<E, A, I, N, r_o, r_v, f>$, где E – конечное неупорядоченное множество сущностей (классов, типов данных, индивидов и свойств); A – конечное неупорядоченное множество аксиом (некоторых логических утверждений о сущностях); I – конечное неупорядоченное множество ссылок на другие онтологии (импортов); N – конечное неупорядоченное множество пар вида (p, n); p – префикс пространства имен (короткая строка, используемая для сокращения имен сущностей); n – интернационализированный идентификатор (IRI) пространства имен; r_o – интернационализированный идентификатор онтологии; r_v – интернационализированный идентификатор версии онтологии; f – формат хранения онтологии, принимающий одно из следующих значений: RDF/XML, Turtle, OWL/XML, OWL Functional Syntax, OWL Manchester Syntax.

Пусть v_1 – исходная версия онтологии, v_2 – измененная версия. Функцию, вычисляющую набор изменений между двумя версиями, обозначим через $\delta(v_1, v_2)$. Функцию ε, применяющую набор изменений к онтологии, будем называть функцией применения изменений. Задача сравнения онтологий v_1 и v_2 состоит в нахождении набора изменений $C=\delta(v_1,v_2)$, такого что $\varepsilon(v_1,C)=v_2$.

Метод сравнения онтологий. Для упрощения алгоритма сравнения целесообразно представить онтологию как конечное неупорядоченное множество S утверждений (диаграмма типов утверждений приведена на рис. 1):

– OntologyFormatStatement – утверждение, определяющее формат онтологии (RDF/XML, Turtle, OWL/XML, OWL Functional Syntax, OWL Manchester Syntax);

– NamespacePrefixStatement – утверждение о префиксе пространства имен, содержащее строку префикса и строку пространства имен;

– ImportStatement – утверждение, определяющее ссылку на импортируемую онтологию;

– OntologyIRIStatement – утверждение, определяющее идентификатор онтологии;

– VersionIRIStatement – утверждение, определяющее идентификатор версии онтологии;

– AxiomStatement – утверждение, описывающее аксиому на языке OWL 2.

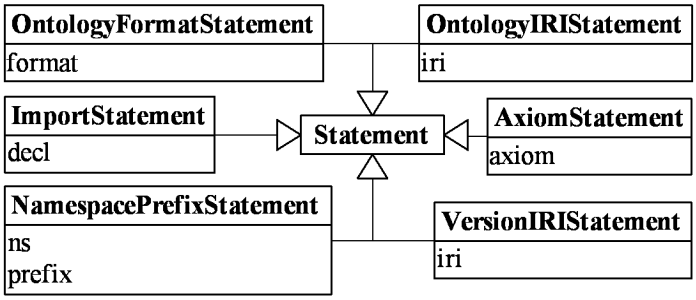

Рис. 1. Типы утверждений

В данном случае сравнение онтологий сводится к сравнению конечных неупорядоченных множеств S_1 и S_2 (S_1 – множество утверждений, полученное из онтологии v_1; S_2 – множество утверждений, полученное из онтологии v_2). В общем виде каждое изменение c можно представить как пару (p, s), где p – операция ("+" или "–"), примененная к утверждению s. Набор изменений C можно получить из пары (S^-, S^+), где $S^- = S_1 \backslash S_2$ – множество утверждений, удаленных из S_1; $S^+ = S_2 \backslash S_1$ – множество утверждений, добавленных в S_1: $C = \{("-", s) \mid s \in S^-\} \cup \{("+", s) \mid s \in S^+\}$.

Алгоритм применения изменений C к онтологии v_1 включает следующие шаги:

1. Преобразовать онтологию v_1 в множество утверждений S_1.
2. Применить изменения $S_2 = (S_1 \setminus \{ s \mid c \in C\ (p="-") \}) \cup \{ s \mid c \in C\ (p="+") \}$.
3. Преобразовать множество утверждений S_2 в онтологию v_2.

Программная реализация. На рис. 2 показана диаграмма основных классов программы. Для работы с онтологиями OWL 2 использовалась библиотека OWL API [4]. Класс OWLOntology из этой библиотеки содержит объектное представление онтологии $<E, A, I, N, r_o, r_v, f>$. Конструктор класса ComparableOntology принимает в качестве аргумента экземпляр класса OWLOntology и строит из него несколько наборов утверждений, разбитых по типам. Метод get-

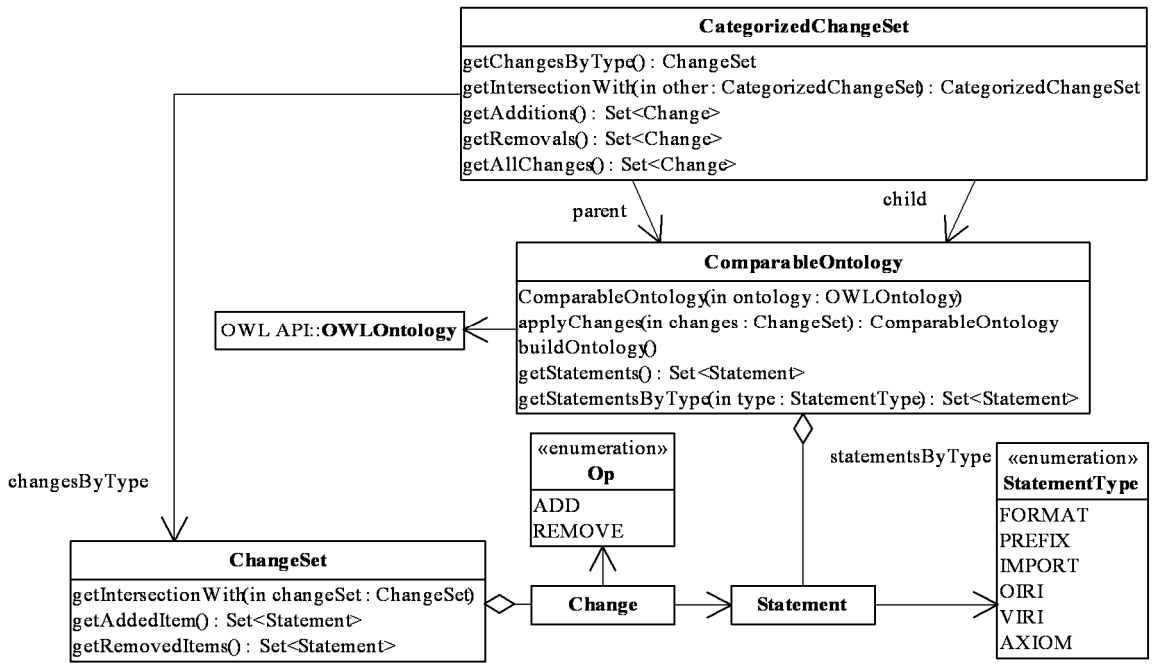

Рис. 2. Основные классы программы

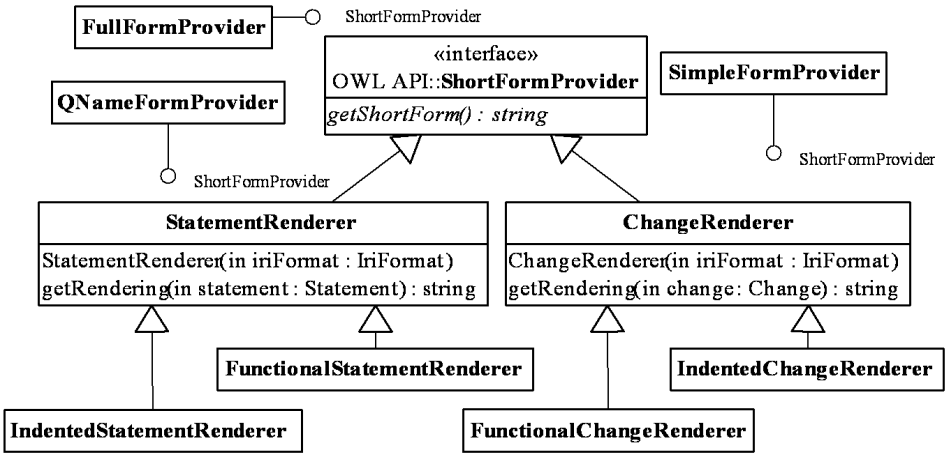

Рис. 3. Классы, отвечающие за текстовое представление изменений

Statements() возвращает множество всех утверждений для выполнения сравнения. Метод buildOntology() восстанавливает экземпляр класса OWLOntology из наборов утверждений statementsByType. Класс CategorizedChangeSet представляет собой набор изменений между двумя онтологиями, сгруппированных по типу утверждений. Класс ChangeSet содержит множество изменений, каждое из которых представлено экземпляром класса Change.

На рис. 3 представлена диаграмма классов, отвечающих за текстовое представление изменений.

Операция getRendering() класса ChangeRenderer возвращает строку, описывающую изменение, переданное в качестве аргумента. Эта операция использует операцию StatementRenderer.getRendering() для преобразования утверждения в текст. Результат выполнения операции зависит от реализации интерфейса ShortFormProvider, которая выбирается автоматически на основе параметра iriFormat. Этот параметр может принимать одно из следующих значений: SIMPLE, QNAME и FULL. В таблице показано соответствие между значением параметра iriFormat, реализацией StatementRenderer и фактическим представлением изменения.

На рис. 4 показана диаграмма последовательности вызовов при отображении изменения с помощью FunctionalChangeRenderer.

Тест производительности. Пусть $m = |C| / (|S_1| + |S_2|)$ – коэффициент, показывающий, насколько была изменена онтология. На рис. 5 представлена зависимость времени работы t алгоритма от количества утверждений N при $m = 0{,}2$. Общее время включает время загрузки обеих версий онтологий в оперативную память средствами OWL API, время преобразования обеих

Различные способы представления утверждения

iriFormat	FunctionalSyntaxRenderer	IndentedStatementRenderer
SIMPLE	ClassAssertion(Person Ivan)	ClassAssertion: Person Ivan
QNAME	ClassAssertion(foaf:Person sample:Ivan)	ClassAssertion: foaf:Person sample:Ivan
FULL	ClassAssertion(<http://xmlns.com/foaf/0.1/Person> <http://kms.cc.tpu.ru/ontologies/sample#Ivan>)	ClassAssertion: <http://xmlns.com/foaf/0.1/Person> <http://kms.cc.tpu.ru/ontologies/sample/person#Ivan>

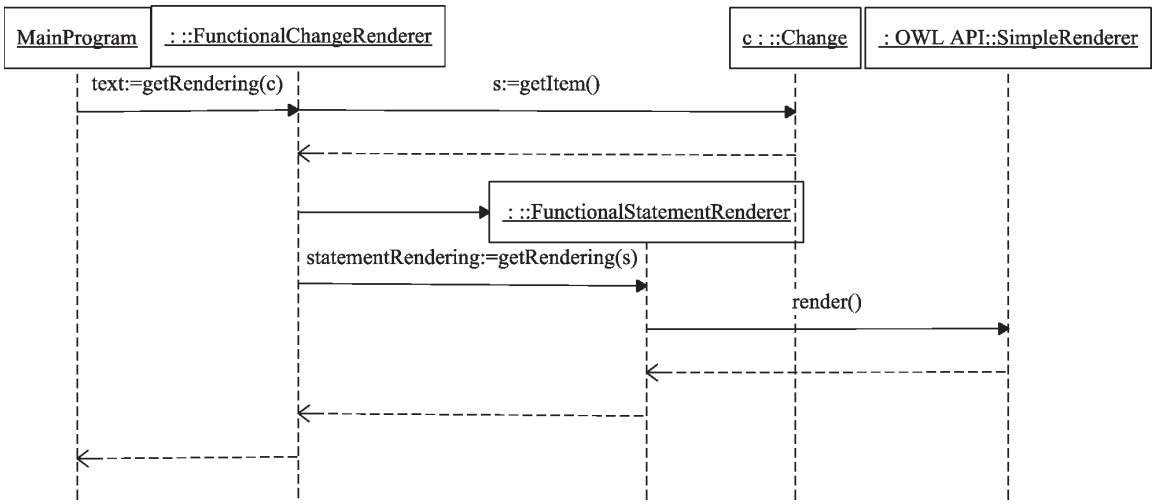

Рис. 4. Диаграмма последовательности для операции FunctionalChangeRenderer.getRendering()

онтологий в множество утверждений и время сравнения. В среднем загрузка в оперативную память занимает в 8,6 раза больше времени, чем преобразование и сравнение. Эксперимент проводился на компьютере с процессором AMD A6-3410MX с тактовой частотой 1,6 ГГц и оперативной памятью типа DDR3 объемом 2 Гб с пропускной способностью 667 МГц.

Заключение. Проведенное исследование позволяет сделать следующие выводы. Предложенный метод сравнения онтологий, в отличие от [2], основан на прямой семантике OWL 2, что обусловливает простоту алгоритмов и надежность их реализации. Предложенный метод, в отличие от методов, приведенных в [5, 6], позволяет учитывать изменения формата онтологии, префиксов пространств имен, импортов и идентификаторов онтологии. Алгоритм эффективно работает с достаточно большими онтологиями: сравнение онтологий, содержащих до 512 000 утверждений, занимает менее 1 мин. Результаты решений, полученные с помощью предложенного метода, предполагается использовать при разработке алгоритма анализа изменений для выявления затронутых сущностей и алгоритма трехстороннего слияния онтологий.

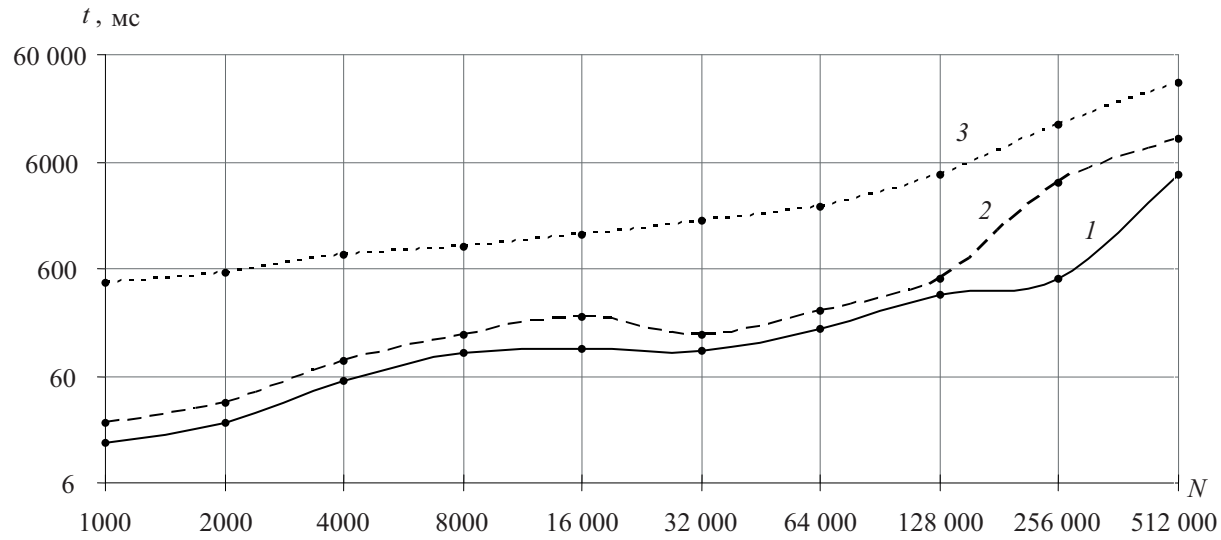

Рис. 5. Зависимость времени работы алгоритма от количества утверждений:
1 – время сравнения; *2* – время преобразования и сравнения; *3* – общее время

Список литературы

1. MOTIK B., PATEL P. F., PARSIA B. OWL 2 Web Ontology Language structural specification and functional style syntax // World Wide Web Consortium. 2009. [Electron. resource]. http://www.w3.org/TR/owl2-syntax/.

2. CARROLL J. Matching RDF Graphs. 2001. 26 ноября. [Electron. resource]. http://www.hpl.hp.com/techreports/2001/HPL-2001-293.pdf.

3. DELTAVIEW // ESW Wiki. 2007. 12 ноября. [Electron. resource]. http://esw.w3.org/DeltaView.

4. HORRIDGE M., BECHHOFER S. The OWL API: A Java API for working with OWL 2 Ontologies // OWL experiences and directions: Proc. of the 5th Intern. workshop, Chantilly, VA (US), Oct. 23–24, 2009. CEUR Workshop Proc. 2009. V. 529. P. 53–62.

5. GONÇALVES R. S., PARSIA B., SATTLER U. Ecco: A hybrid diff tool for OWL 2 ontologies // OWL: Experiences and directions: Proc. of the workshop, Heraklion (Greece), May 27–28, 2012. CEUR. Workshop Proc., 2012. V. 849. [Electron. resource]. http://ceur-ws.org/Vol-849/paper_27.pdf.

6. KREMEN P., SMID M., KOUBA Z. OWLDiff: A practical tool for comparison and merge of OWL ontologies // Database and expert systems applications: Proc. of the 22nd Intern. workshop, Toulouse (France), Aug. 29 – Sept. 2, 2011. Washington: IEEE Computer Soc. Press, 2011. P. 229–233.

Заикин Иван Анатольевич – ассист. Института кибернетики
Томского политехнического университета;
тел.: 8-953-925-33-89; e-mail: i@tpu.ru

Дата поступления – 25.09.12 г.