УДК 004.8; 004.4'2

МЕТОД ТРЁХСТОРОННЕГО СЛИЯНИЯ ОНТОЛОГИЙ НАЯЗЫКЕ OWL 2

И.А. Заикин

Томский политехнический университет
E-mail: zaikin@tpu.ru

Предложен алгоритм трёхстороннего слияния онтологий при многопользовательском редактировании путём попарного сравнения онтологий, основанный на выявлении совпадающих и конфликтующих изменений. Разработана программная реализация алгоритма.

Ключевые слова:
Онтология, OWL 2, трёхстороннее слияние, версия, изменение, утверждение.

Key words:
Ontology, OWL 2, three-way merge, version, change, statement.

Онтология – это формальное описание определённой предметной области на стандартизированном языке, понятном как людям, так и вычислительным машинам. В настоящее время наиболее распространённым из таких языков является язык OWL 2 [1]. С каждым годом возрастает как количество разработанных онтологий, так и их сложность. Разработка сложных онтологий требует участия специалистов по языку онтологий, специалистов в предметной области, а также специалистов по управлению качеством. Совместная разработка сложных онтологий почти невозможна без инструментальных средств поддержки групповой работы, в частности без систем управления версиями, которые дают возможность сохранять журнал изменений, в случае необходимости отменять нежелательные изменения, сравнивать версии, объединять изменения разных членов команды и облегчают разрешение конфликтов. Одним из важнейших компонентов таких систем является программа для слияния изменений, выполненных различными пользователями. На рис. 1 показаны входные и выходные данные программы трёхстороннего слияния. Символом v_0 обозначена исходная версия онтологии, v_1 – версия, изменённая первым пользователем, v_2 – версия, изменённая вторым пользователем, v_3 – итоговая версия, содержащая изменения обоих пользователей.

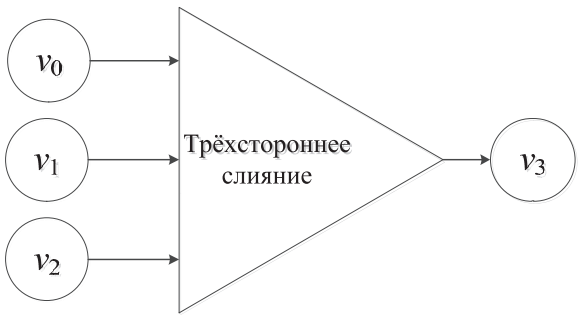

Рис. 1. *Схема трёхстороннего слияния версий*

Для выполнения слияния изменений в файлах традиционно используется попарное их сравнение и последующий поиск совпадающих, несовпадающих и конфликтующих изменений. В данной работе этот подход применяется к онтологиям на языке OWL 2 (рис. 2). В то время как обычные программы для слияния изменений в текстовых файлах опираются на предположение, что порядок строк в тексте имеет значение, 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 слияния изменений в онтологиях обычные программы слияния изменений в текстах.

Язык OWL 2 имеет две семантики: 1) исторически сложившуюся, основанную на RDF, где основными понятиями являются ресурсы и отношения между ними, а сложные конструкции представлены с использованием так называемых «пустых узлов» и 2) прямую семантику, где основными понятиями являются сущность и аксиома. Сравнение RDF-графов – вычислительно сложный процесс [2], а при наличии большого количества пустых узлов невозможно однозначно идентифицировать изменённые триплеты [3]. В данной работе используется метод структурного сравнения онтологий на основе прямой семантики OWL 2. Использование прямой семантики языка OWL 2 даёт возможность не задумываться о пустых узлах, а выполнять сравнение поаксиомно, используя операции над множествами. Предложенный подход также учитывает атрибуты онтологий, не входящие в логическую составляющую, об изменении которых, тем не менее, важно знать при разработке онтологий. Благодаря этому можно, например, выполнять слияние онтологий в различных форматах представления без потери информацию об изменении формата.

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 множе-

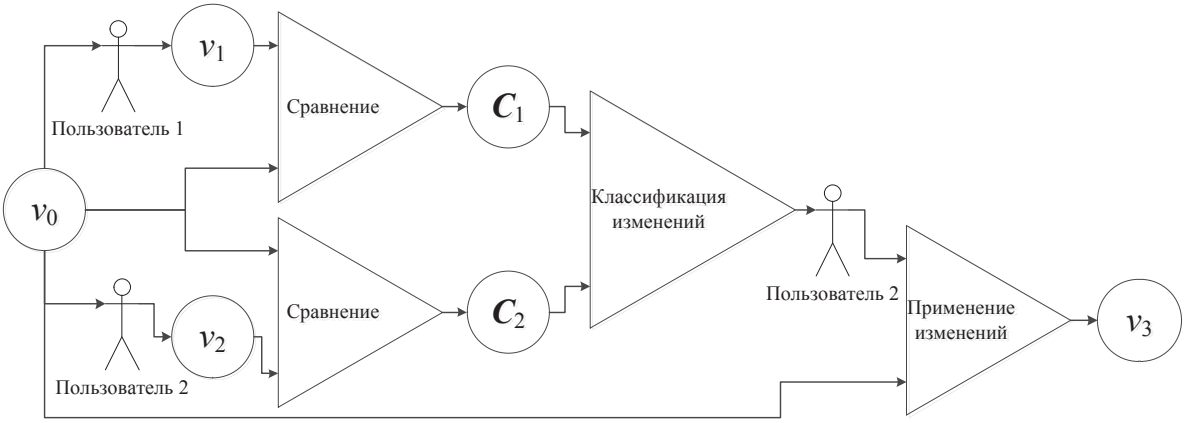

Рис. 2. *Поток изменений при трёхстороннем слиянии*

ство аксиом (некоторых логических утверждений о сущностях); I – конечное неупорядоченное множество ссылок на другие онтологии (импортов); N – конечное неупорядоченное множество пар вида (p,n), где p – префикс пространства имён (короткая строка, используемая для сокращения имён сущностей), а n – интернационализированный идентификатор (IRI) пространства имён; r_o – интернационализированный идентификатор онтологии; r_v – интернационализированный идентификатор версии онтологии; f – формат хранения онтологии, может принимать одно из следующих значений: «RDF/XML», «Turtle», «OWL/XML», «OWL Functional Syntax», «OWL Manchester Syntax».

Пусть v_0 – исходная версия онтологии, v_1 – версия, изменённая первым пользователем, v_2 – версия, изменённая вторым пользователем, C_1 – конечное неупорядоченное множество изменений, внесённых в v_0 первым пользователем, C_2 – конечное неупорядоченное множество изменений, внесённых в v_0 вторым пользователем. Эти изменения можно классифицировать следующим образом:

- Совпадающие изменения $C_m=C_1\cap C_2$;
- Конфликтующие изменения пользователя 1: C_{C1};
- Остальные изменения пользователя 1: $C_{O1}=(C_1\backslash C_m)\backslash C_{C1}$;
- Конфликтующие изменения пользователя 2: C_{C2};
- Остальные изменения пользователя 2: $C_{O2}=(C_1\backslash C_m)\backslash C_{C2}$.

Задача трёхстороннего слияния онтологий v_0, v_1 и v_2 состоит в получении онтологии v_3, основанной на v_0 и содержащей изменения C_m, C_{O1}, C_{O2} и часть изменений C_{C1} и C_{C2}, не приводящих к конфликту.

Метод трёхстороннего слияния онтологий

Для выполнения трёхстороннего слияния онтологий с учётом атрибутов, не входящих в логическую составляющую, целесообразно представить онтологию как конечное неупорядоченное множество утверждений S следующих типов:

- OntologyFormatStatement – утверждение, определяющее формат онтологии («RDF/XML», «Turtle», «OWL/XML», «OWL Functional Syntax», «OWL Manchester Syntax»);
- NamespacePrefixStatement – утверждение о префиксе пространства имён, содержащее строку префикса и строку пространства имён;
- 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.

Обозначим такое преобразование как τ:

$$S_i=\tau(v_i),$$

а обратное преобразование – как τ^{-1}:

$$v_i=\tau^{-1}(S_i).$$

В этом случае сравнение онтологий сводится к простому сравнению конечных неупорядоченных множеств утверждений $S_i=\tau(v_i)$. Обозначим функцию, вычисляющую изменения между двумя множествами утверждений S_0 и S_i, как $\delta(S_0,S_i)$. Каждое изменение состоит из утверждения и операции («+» или «–»), применённой к нему. Введём дополнительные обозначения:

$\gamma(o,s)$ – изменение, представляющее операцию o над утверждением s; $\omega(c)$ – операция, соответствующая изменению c: $\omega(\gamma(o,s))=o$; $\varsigma(c)$ – утверждение, соответствующее изменению c: $\varsigma(\gamma(o,s))=s$.

Изменения между S_0 и S_i можно вычислить следующим образом:

$$C_i=\delta(S_0,S_i)=\{\gamma(«-»,s)|s\in S_0\backslash S_i\}\cup\{\gamma(«+»,s)\ |s\in S_i\backslash S_0\}.$$

Определим сигнатуру $\sigma(s)$ утверждения s как конечное неупорядоченное множество сущностей, на которые ссылается утверждение s. Сигнатура множества утверждений рассчитывается как объединение сигнатур отдельных утверждений:

$$\sigma(\{s_1,s_2,...,s_n\})=\sigma(s_1)\cup\sigma(s_2)\cup...\cup\sigma(s_n).$$

Сигнатура изменения равна сигнатуре соответствующего утверждения:

$$\sigma(\gamma(o,s))=\sigma(s).$$

Сигнатура множества изменений рассчитывается как объединение сигнатур отдельных изменений:

$$\sigma(\{c_1,c_2,...,c_n\})=\sigma(c_1)\cup\sigma(s_2)\cup...\cup\sigma(c_n).$$

Конфликтующие изменения пользователя 1 можно найти по следующей формуле:

$$C_{C1}=\{c\in C_1\,|\,\sigma(c)\cap\sigma(C_2\backslash C_m)\neq\varnothing\}.$$

Согласно этой формуле, конфликтующими изменениями пользователя 1 считаются изменения, сигнатура которых пересекается с сигнатурой изменений пользователя 2, т. е. изменения, затрагивающие те же сущности, что и изменения пользователя 2. Аналогично конфликтующие изменения пользователя 2

$$C_{C2}=\{c\in C_2\,|\,\sigma(c)\cap\sigma(C_1\backslash C_m)\neq\varnothing\}.$$

Конфликтующие изменения не могут быть разрешены автоматически, поэтому в большинстве случаев при выполнении трёхстороннего слияния требуется участие пользователя. Обычно выполняет слияние пользователь, вносящий изменения последним. От него требуется выбрать из конфликтующих изменений $C_{C1}\cup C_{C2}$ те, которые необходимо сохранить. Обозначим эти изменения как C_S. Функцию, применяющую множество изменений C к множеству утверждений S, обозначим как $\varepsilon(S,C)$. Тогда итоговая версия онтологии, содержащая изменения обоих пользователей,

$$v_3=\tau^{-1}(\varepsilon(\tau(v_0),\, C_m\cup C_{O1}\cup C_{O2}\cup C_S)).$$

Программная реализация

На рис. 3 показана диаграмма 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 v класса OWLOntology и строит из него множество утверждений $S_i=\tau(v_i)$, которое можно получить, вызвав метод getStatements().

Алгоритм сравнения

Конструктор ChangeSet (s1, s2) выполняет сравнение множеств утверждений по следующему алгоритму.

Входные данные:

- S_1 – исходное множество утверждений;
- S_2 – изменённое множество утверждений.

Выходные данные:

- C – множество изменений.

1. $S^- \leftarrow S_1\backslash S_2$
2. $S^+ \leftarrow S_2\backslash S_1$
3. Для каждого $s\in S^-$
4. $\quad C\leftarrow C\cup\{\gamma(«-»,s)\}$
5. Для каждого $s\in S^+$
6. $\quad C\leftarrow C\cup\{\gamma(«+»,s)\}$

Алгоритм классификации изменений

Классификацию изменений выполняет конструктор класса ConflictFinder.

Входные данные:

- base=$\tau(v_0)$ – множество утверждений исходной версии онтологии;
- remote=$\tau(v_1)$ – множество утверждений онтологии, изменённой другими пользователями;
- local=$\tau(v_2)$ – множество утверждений онтологии, изменённой локальным пользователем.

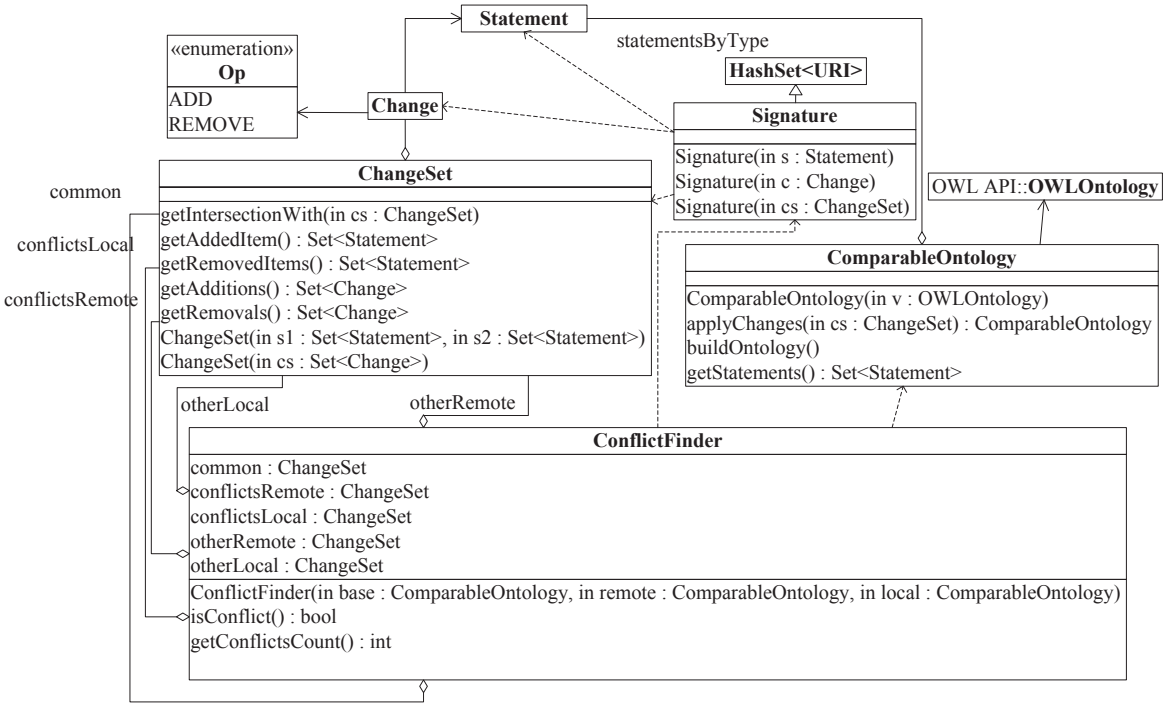

Рис. 3. *Основные классы программы*

Выходные данные:

- common=$\boldsymbol{C}_m$ – совпадающие изменения;
- conflictsRemote=$\boldsymbol{C}_{C1}$ – конфликтующие изменения других пользователей;
- otherRemote=$\boldsymbol{C}_{O1}$ – остальные изменения других пользователей;
- conflictsLocal=$\boldsymbol{C}_{C2}$ – конфликтующие изменения локального пользователя;
- otherLocal=$\boldsymbol{C}_{O2}$ – остальные изменения локального пользователя.

1. remoteChanges←δ(base, remote)
2. localChanges←δ(base, local)
3. common←remoteChanges∩localChanges
4. remoteChanges←remoteChanges\common
5. localChanges←localChanges\common
6. remoteChangesSig←σ(remoteChanges)
7. localChangesSig←σ(localChanges)
8. conflictsRemote←∅
9. conflictsLocal←∅
10. otherRemote←remoteChanges
11. otherLocal←localChanges
12. Для каждого $c \in$remoteChanges
13. Если localChangesSig∩$\sigma(c)\neq\varnothing$
14. conflictsRemote←conflictsRemote∪{c}
15. otherRemote←otherRemote\{c}
16. Для каждого $c\in$localChanges
17. Если remoteChangesSig∩$\sigma(c)\varnothing$
18. conflictsLocal←conflictsLocal∪{c}
19. otherLocal←otherLocal\{c}

Алгоритм применения изменений

Алгоритм применения изменений прост благодаря тому, что онтология, включая её атрибуты, представлена в виде множества утверждений. Алгоритм реализован в методе applyChanges() класса ComparableOntology.

Входные данные:

- statements – множество утверждений;
- changes – множество изменений.

Выходные данные:

- statements – множество утверждений.

1. Для каждого $c\in$changes
2. Если $\omega(c)$=«+»
3. statements←statements∪{$\varsigma(c)$}
4. Иначе
5. statements←statements\{$\varsigma(c)$}

Тест производительности

Обозначим $m(v_i,v_j)=|\delta(\tau(v_i),\tau(v_j))|/(|\tau(v_i)|+|\tau(v_j)|)$.

Для проверки производительности метода были сгенерированы тройки онтологий (v_0,v_1,v_2) со следующими характеристиками:

- $\boldsymbol{C}_m=\boldsymbol{C}_{C1}=\boldsymbol{C}_{C2}=\varnothing$;
- $\boldsymbol{C}_{O1}=\boldsymbol{C}_1$;
- $\boldsymbol{C}_{O2}=\boldsymbol{C}_2$;
- $|\tau(v_0)|=|\tau(v_1)|=|\tau(v_2)|$;
- $m(v_0,v_1)=m(v_0,v_2)=m(v_1,v_2)=0{,}02$.

Первые три характеристики обусловлены тем фактом, что выполнить слияние в автоматическом режиме возможно лишь при отсутствии конфликтов, которые потребовалось бы разрешать вручную при их наличии.

На рис. 4 показан график зависимости времени t слияния от количества утверждений N в онтологиях на вычислительной машине с процессором AMD A6–3410MX с тактовой частотой 1,6 ГГц и оперативной памятью типа DDR3 объёмом 8 Гб с пропускной способностью 667 МГц. Видно, что при больших значениях $|\tau(v)|$ время слияния растёт экспоненциально. Однако при очень больших значениях $|\tau(v)|$ объём оперативной памяти, требуемый для размещения четырёх версий онтологии, превышает доступный объём физической оперативной памяти, что приводит к использованию дискового пространства, вследствие чего время слияния значительно увеличивается.

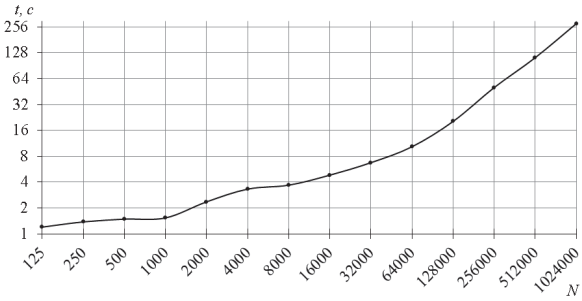

Рис. 4. *График зависимости времени слияния от количества утверждений*

Графический пользовательский интерфейс

Для разрешения конфликтов необходимо предоставить пользователю возможность выбирать изменения, которые он желает внести в итоговую версию онтологии. На рис. 5 показан графический пользовательский интерфейс программы слияния.

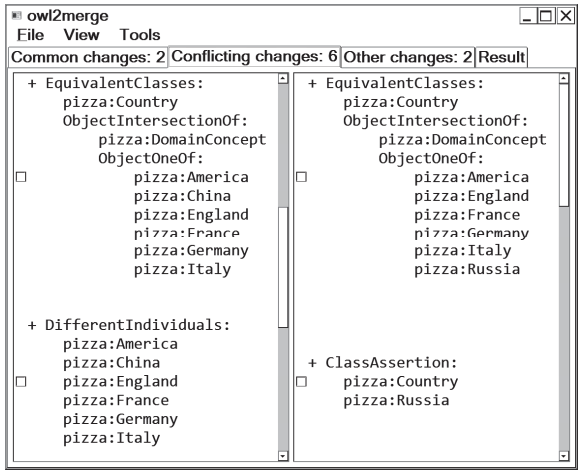

Рис. 5. *Вкладка «Конфликтующие изменения» главного окна программы трёхстороннего слияния онтологий*

Главное окно программы состоит из четырёх вкладок, отображающих совпадающие изменения, конфликтующие изменения, остальные изменения и результат – изменения, выбранные для внесения в итоговую версию онтологии. По умолчанию выбраны все совпадающие и другие неконфликтующие изменения, а конфликты предлагается разрешить пользователю.

Заключение

На основе проведённых исследований можно сделать следующие выводы:

1. Предложенный метод позволяет выполнять трёхстороннее слияние онтологий с учётом изменений формата онтологии, префиксов пространств имён, импортов и идентификаторов онтологии за приемлемое время.
2. Недостатком программной реализации является отсутствие возможности группировки изменений по сущностям, а также добавления новых изменений (не внесённых ни одним из пользователей). В дальнейшем предполагается усовершенствование программной реализации.

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. URL: http://www.w3.org/TR/owl2-syntax/ (дата обращения: 23.10.2012).
2. Carroll J. Matching RDF Graphs. 2001. URL: http://www.hpl.hp.com/techreports/2001/HPL-2001–293.pdf (дата обращения: 23.10.2012).
3. DeltaView // ESW Wiki. 2007. URL: http://esw.w3.org/DeltaView (дата обращения: 23.10.2012).
4. Horridge M., Bechhofer S. The OWL API: A Java API for Working with OWL 2 Ontologies // OWL Experiences and Directions: Procs. of the VI Intern. Workshop. – Chantilly, 2009. – V. 529. – P. 53–62.

Поступила 31.10.2012 г.

УДК 004.02:004.82

СЕМАНТИЧЕСКОЕ АННОТИРОВАНИЕ ДОКУМЕНТОВ В ЭЛЕКТРОННЫХ БИБЛИОТЕКАХ

Ле Хоай, А.Ф. Тузовский

Томский политехнический университет
E-mail: lehotomsk@yahoo.com

Рассматривается задача выполнения аннотирования документов в электронных библиотеках, использующих семантические технологии. Описываются причины и преимущества использования таких аннотаций. Предложен новый метод выполнения полуавтоматического аннотирования, и показаны результаты тестирования его программной реализации.

Ключевые слова:
Аннотирование документов, семантические технологии, семантическая аннотация, электронная библиотека организации, методы аннотирования.

Key words:
Document annotation, semantic technology, semantic annotation, enterprise digital library, annotation method.

Введение

В настоящее время большинство электронных ресурсов, в особенности веб-страницы, создается только для их использования людьми. Форматы их описания практически не включают формальных описаний знаний, содержащихся в этих документах. Формальное описание основного смысла документа в удобном для программной обработки формате является целью использования активно разрабатываемых в последнее десятилетие семантических технологий, таких как RDF, RDFS, OWL и SPARQL.

Электронные библиотеки представляют собой специализированные информационные системы, которые выполняют управление коллекциями электронных ресурсов (например, таких как текстовые документы, изображения, мультимедиа файлы) с целью повышения эффективности использования содержащихся в них знаний некоторыми сообществами пользователей. Под семантическими электронными библиотеками (СЭБ) понимаются электронные библиотеки, использующие семантические технологии для организации всех процессов своей работы, таких как описание ресурсов, ведение каталогов, описание профилей пользователей, поиск и рекомендация ресурсов пользователям и т. п. Одной из важных функций семантических электронных библиотек является предоставление возможности аннотирования публикуемых ресурсов. В данной статье рассматривается проблема реализации такого подхода за счет перехода от человеко-читаемых к программно-обрабатываемым описаниям электронных ресурсов.