Технология импорта фрагментов из OWL и KIF-онтологий

В. Ш. Рубашкин, М. В. Фадеева, Б. Ю. Чуприн

Санкт-Петербургский государственный университет
vrubashkin@yandex.ru, fadeevamf@gmail.com, b.chuprin@gmail.com

Аннотация

В работе описан процесс создания дружественной диалоговой среды для выбора и переноса фрагментов онтологии-источника в целевую онтологию. Предполагается, что онтология-источник представлена на языке KIF или OWL. Импорт фрагментов рассматривается как один из этапов создания унифицированной онтологии, интегрирующей частные онтологии через онтологию верхнего уровня.

1. Постановка задачи

Общая задача: создание унифицированной системы онтологий, интегрирующей частные онтологии через онтологию верхнего уровня.

Актуальная повестка дня:

– от онтологии задач и отраслевых онтологий к единой расширяемой онтологии;
– от “выравнивания“ (*mapping, alignment*) и “склеивания“ (*merging*) частных онтологий к импорту фрагментов онтологий в унифицированную среду.

Непосредственная цель работы: создание дружественной диалоговой среды для выбора и переноса фрагментов онтологии-источника в целевую онтологию. "Дружественность" среды понимается как возможность выбирать, переносить и редактировать заимствуемые фрагменты, главным образом, средствами визуального интерфейса.

Общие предпосылки. Целевая среда - онторедактор и онтология I*nTez*. Источники пополнения - онтологии, определенные средствами языков *OWL* или *KIF*. В качестве конкретного примера KIF-онтологии взята онтология *SUMO*, использующая диалект *KIF-SUO*.

2. Положение дел

Методы и средства объединения онтологий, сформированные к началу 2000-х гг., рассмотрены в [3]. Из представленных в этой монографии проектов (*ONIONS* — Conceptual Modeling Group, Rome; *FCA-Merge* — Institute AIFB, University of Karlsruhe; *PROMPT* — Stanford Medical Informatics group, Stanford University) долгожителем, видимо, оказался только *PROMPT* [4] — отчасти благодаря использованию совместно с популярным онторедактором *Protegé*. В любом случае, эти разработки не решили тех практических проблем, которые определились после осознания задачи создания интегрированных политематических онтологий.

В последующих работах представлено несколько идей, определяющих направление дальнейшего развития технологий импорта. В [5] предлагается различать три метода использования существующей онтологии при создании новой онтологии:

1) импорт всей онтологии-источника в новую онтологию с последующим игнорированием ненужных концептов и аксиом (релизуемый в OWL с помощью *owl:imports*.);
2) импорт отдельных концептов и аксиом;
3) разбиение онтологии-источника на *модули* - фрагменты из некоторого количества концептов, объединенные по тематике - с последующим переносом в новую онтологию только необходимых модулей.

Авторы подчеркивают необходимость развития методов, обеспечивающих *частичное* использование существующих баз знаний для формирования новых онтологий и обращают внимание на проблемы, связанные с тем, что онтологии могут быть представлены на разных языках.

В [2] предпринята попытка уточнить само понятие *импорта* концептов и предложено формальное определение “импорта множества терминов S из базы знаний по правилам G”. Под импортом понимается выборочный перенос фрагментов онтологии в другую онтологию на любом этапе создания и использования онтологии. Предполагается, что онтологии представлены на одном языке. Особое внимание уделено не импорту целого файла онтологии, а методам, реализующим более точный и осознанный выбор переносимой части, с тем чтобы переносить как *модули*, так и отдельные концепты.

В [6] описывается плагин для Protégé 2.0 - *SOM* (Simple Ontology Merger), позволяющий импортировать классы, свойства и атрибуты классов из внешней онтологии, а также задавать отношения между классами пополняемой онтологии и импортируемыми классами. Авторы статьи сравнивают *SOM* и *PROMT*, наиболее известный плагин Protégé для объединения онтологий. SOM оценивается как простой и удобный инструмент для работы с онтологиями, описанными на языке OWL.

Труды XV Всероссийской объединенной конференции «Интернет и современное общество» (IMS-2012), Санкт-Петербург, Россия, 2012.

В проекте *SAMBO* [1], по-видимому, впервые предложен эргономичный интерфейс, графически представляющий на одном экране и таксономию целевой онтологии, и таксономию онтологии-источника и предоставляющий возможности выбора переносимых фрагментов. Однако возможности инструмента ограничены взаимодействием между OWL-онтологиями.

Суммируя наметившиеся подходы, следует обратить внимание на следующие три пункта.

1. Задача импорта формулируется как задача формирования единой онтологии (системы онтологий).

2. Осознана важность предварительной структуризации онтологии-источника и наличия средств выделения и импорта отдельных фрагментов. Соответственно, может идти речь об ограничении различаемых и используемых в процедурах импорта отношений между концептами.

3. Оформилась идея «визуального импорта» как эргономичного и практически эффективного метода реализации технологии импорта фрагментов онтологий.

3. Что импортировать, как представлять?

Лексические единицы самой онтологии будем далее именовать *концептами*. Элементы языка представления знаний (ЯПЗ), используемые для определения и описания концептов будем называть *конструкторами* онтологии. Конструкторы по отношению к концептам — единицы метаязыка онтологии. Концепты (и данные) используются в высказываниях, описывающих экземпляры; конструкторы — нет. С этой точки зрения сама онтология есть не что иное как *язык для представдления фактографических знаний.*

Первый, возможно, самый трудный вопрос при разработке технологии импорта состоит в том, что чему соответствует. Возможны две ситуации: (а) объединение онтологий, использующих один и тот же ЯПЗ (например, OWL); (б) объединение онтологий, использующих разные ЯПЗ. И в том, и в другом случае придется учитывать различия в разделении на концепты-примитивы и определяемые (производные) концепты, а также различия в способах таксономической организации примитивов и способах определения производных концептов. В случае объединения онтологий, использующих разные ЯПЗ, возникает еще одна проблема, которая состоит в разной категоризации концептов и, соотвественно, разном наборе их словарных характеристик. Для разноязычных онтологий первая и нетривиальная часть алгоритмизации в задаче онтологического импорта сводится к установлению соответствия между категориями языков представления, либо, как, например, при сопоставлении OWL и KIF-представлений, к установлению соответствия между категориями языка представления (OWL) и классами концептов (KIF) — категориальная унификация. Технологические проблемы организации переноса фрагментов по сравнению с этим оказываются более простыми.

Применительно к OWL имеем следующие категории концептов:

– классы (*Class*);
– атрибуты (*DataTypeProperty*);
– бинарные отношения (*ObjectProperty*).

Собственно язык OWL представляет собой набор конструкторов, относящихся к одному из следующих трех типов.

1. Конструкторы примитивов. Они вводят новые концепты путем указания категории и имени. В функциональной нотации это, например:

Declaration(Class(:Person))
Declaration(DataProperty(:hasAge))
Declaration(ObjectProperty(:hasChild)).

2. Конструкторы, определяющие отношения между концептами и их свойства (*SubClassOf, EquivalentClasses, DisjointClasses, DisjointObjectProperties, ObjectPropertyDomain, ObjectPropertyRange, FunctionalDataProperty* и др.).

3. Конструкторы производных концептов – определяют новые концепты в терминах уже имеющихся.(*ObjectIntersectionOf, ObjectUnionOf, ObjectSomeValuesFrom, ObjectHasValue* и др.).

В языке онтологии InTez реализована более детальная содержательная категоризация:

– наименования признаков («атрибуты»);
– имена объектов («классы»);
– имена процессов, действий, событий;
– статические отношения (не обязательно бинарные).

Наиболее существенные отличия от OWL сводятся к следующему.

Классы-примитивы вводятся только через операцию разбиения исходного класса, определяемую через указание некоторого классификационного признака. Таким образом, таксономия примитивов представляется в формате **дерева признаков** с двумя чередующимися типами узлов: узлы - классы и узлы — наименования классификационных признаков [7].

Процессы / действия / события выделены в отдельную категорию, тогда как в OWL они будут представлены в категории *Class* либо *ObjectProperty*.

В языке *KIF* содержательная категоризация, погружена внутрь онтологии. Основные конструкторы концептов-примитивов – *Subclass* и *Partition.* Одновременно они определяют два отношения между концептами — общее-частное (оба) и несовместимость (*Partition*). Последний вводит подклассы, получаемые исчерпывающим разбиением исходного класса: (*partition Entity Physical Abstract).* Однако, в отличие от *InTez*, использование *Partition* для добавления концептов-примитивов в SUMO не является обязательным. Поэтому используется также конструктор *Disjoint.* Поскольку *KIF* - язык логического типа, основные характеристики концептов задаются в форме аксиом. Существенное отличие языка *KIF* от специализированных языков пред-

ставления онтологий состоит в том, что он позволяет формулировать как связи между конкретными концептами (онтологические высказывания), так и *правила*. К последним следует отнести все высказывания, имеющие вид формальной импликации с более чем одним универсальным квантором. Пример - правила типа *uncle rule:*

$\forall x\, y\, z$ ((*ИМЕТЬ_МАТЕРЬЮ (x, y) &*

ИМЕТЬ_БРАТОМ (y, z)) $\rightarrow$ *ИМЕТЬ_ДЯДЕЙ (x, z)).*

Другой, возможно, самый востребованный тип правил – это мета-аксиомы, определяющие логические зависимости не между конкретными концептами, а между определенными типами концептов. Актуальный для онтологии *InTez* пример - утверждение об объемной несовместимости классов *X1* **и** *X2*, выделенных по основанию *Y*. В терминах, определяемых языком словарного описания онтологии *InTez*, это правило может быть представлено в виде:

$\forall$*X1 X2 Y (SC (Y) = scAttr & ST (Y) = stQual &*
SC (X1) = scObj & ST (X1) = stFeat &
ProperAttr (X1) = Y &
SC (X2) = scObj & ST (X2) = stFeat &
ProperAttr (X2) = Y) $\rightarrow$
RExtension (X1 , X2) = tzIncompatible.

Правила в текущей версии онтологии InTez не используются.

4. Технология переноса фрагментов

Технология переноса реализуется посредством двух автономных модулей, взаимодействующих через данные. Первый распознает концепты и таксономические отношения между ними в онтологии-источнике и конвертирует их в формат целевой онтологии. Второй обеспечивает выбор фрагментов в источнике и их перемещение в целевую онтологию.

Результатом конвертирования данных онтологии-источника в структуру онтологии *InTez* является дерево признаков, представляющее распознанную часть таксономии онтологии-источника. В качестве среды хранения используется БД Access с той же, что и в *InTez* логической схемой (таблица концептов и таблица связей). В текущей версии технологии импорта распознаются, извлекаются из онтологии-источника и конвертируются в формат целевой онтологии концепты, вводимые конструкторами *Subclass* и *Partition.* Дополнительно фиксируются отношения *Disjoint* между уже извлеченными концептами. При дальнейшем развитии системы конвертирования предполагается извлекать также концепты, вводимые конструкторами *OWL ObjectIntersectionOf, ObjectSomeValuesFrom, ObjectAllValuesFrom, ObjectHasValue.*

Конвертирование связей типа *Subclass* (с учетом возможности множественного наследования) производится следующим способом. При обнаружении в онтологии-источнике для данного концепта *D* первого отношения *Subclass,* указывающего на подчиняющий концепт *D1,* связь $D \rightarrow D1$ добавляется в дерево признаков, при этом она всегда опосредуется формально добавляемым классификационным признаком, так что в результате создаются и добавляются два концепта. При обнаружении для данного концепта *D* всех последующих отношений *Subclass* (каждый раз проверяется присутствие концепта *D* в дереве признаков) между *D* и подчиняющим концептом *D2* устанавливается отношение *импликативной связи*, как это предусмотрено в языке представления онтологии *InTez.*

Собственно технология переноса реализуется как взаимодействие между двумя оконными интерфейсами онторедактора *InTez* [8], [9], отображающими, соответственно, целевую онтологию и приведенную к формату целевой онтологии онтологию-источник. В окне онтологии-источника администратор указывает в дереве признаков концепт — вершину поддерева, выбранного для переноса; она запоминается в поле *Выбрано* таблицы концептов. После этого в окне целевой онтологии указывается концепт, которому должен быть подчинен переносимый фрагмент онтологии, и затем кнопкой «Забрать» инициируется процедура переноса. (При этом проверяется соответствие семантической категории и семантического типа подчиняющего и подчиняемого концепта).

Модуль распознавания реализован на языке Java с использованием регулярных выражений в качестве основного инструмента распознавания. Модуль переноса реализован как дополнительная опция онторедактора *InTez.* Модуль переноса последовательно выбирает очередной куст переносимого фрагмента в БД онтологии-источника и добавляет его в дерево признаков целевой онтологии. Для добавления используются стандартные процедуры ввода онторедактора *InTez.*

В дальнейшем предполагается проверять наличие производных концептов, которые также будут добавляться в целевую онтологию с использованием стандартных процедур ввода онторедактора *InTez*:

1) в формате *И-толкования* (конструктор *OWL ObjectIntersectionOf*);

2) в формате *Определяемый значением признака* и *R-толкование* (конструкторы *OWL ObjectHasValue, ObjectSomeValuesFrom, ObjectAllValuesFrom*).

Добавляемые концепты удаляются из онтологии — источника.

Вид интерфейса пользователя в момент выбора добавляемого фрагмента показан на рис.1.

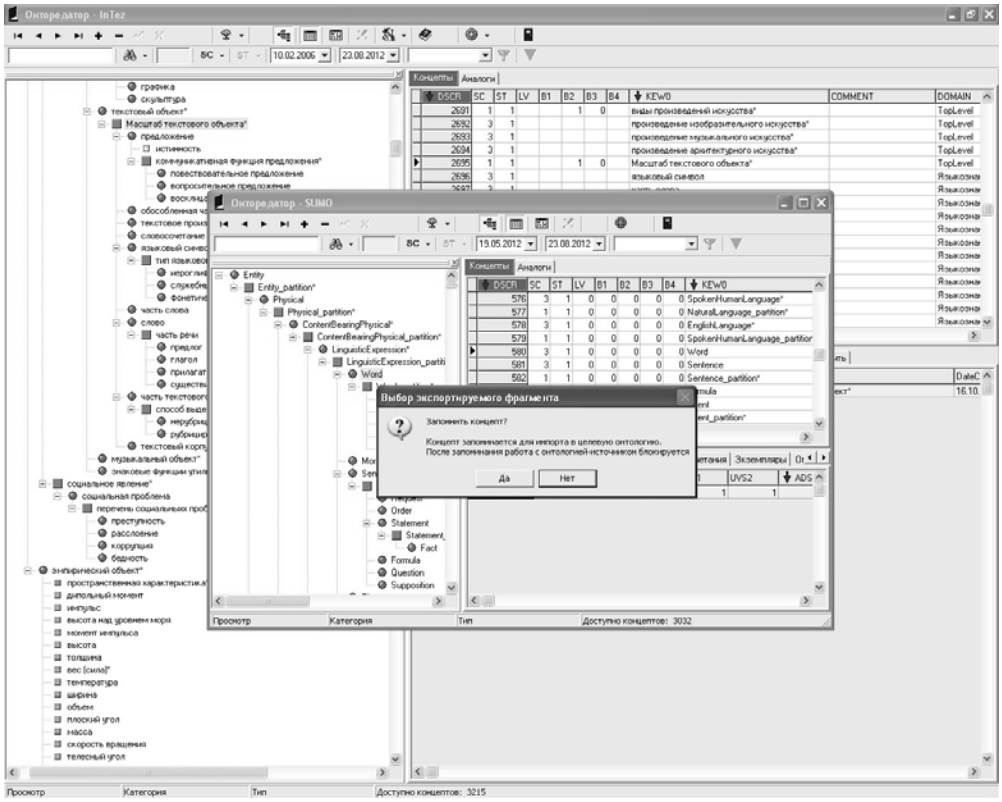

Рис.1. Интерфейс пользователя InTez

Литература

[1] Abdulahad B., Lounis G. A user interface for the ontology merging tool SAMBO. (Technical Report), 2004. URL: http://liu.diva-portal.org/smash/record.jsf?pid=diva2:19999 (дата обращения: 01.08.2012).

[2] Borgida A. On Importing Knowledge from DL Ontologies: some intuitions and problems // Proc. 20th Intern. Workshop on Description Logics, Bressanone, Italy, 2007. URL: http://dydan.rutgers.edu/Research/SemanticGraphs/Papers/DL07.final.pdf (дата обращения: 16.09.2012).

[3] Gomez-Perez A., Fernando-Lopez M., Corcho O. Ontological Engineering. Springer – Ferlag, 2004.

[4] Malik S.K., Prakash N., Rizvi S.A.M. Ontology Merging Using Prompt Plug-In of Protégé in Semantic Web // Proceedings of the 2010 International Conference on Computational Intelligence and Communication Networks, 2010. P. 476-481.

[5] Pan J. Z., Serafini L., Zhao Y. Semantic Import: An Approach for Partial Ontology Reuse // Proc. of the ISWC 1st Workshop on Modular Ontologies (WoMO'06), 2006. URL: http://ceur-ws.org/Vol-232/paper6.pdf (дата обращения: 16.09.2012).

[6] Zheng H.T., Kim H.G. Developing an Easy Tool for Ontology Integration // 9th Asia-Pacific Decision Sciences Institute Conference (APDSI-KOPOMS KOREA 2004) / Global Electronic Business Research Center. URL: http://iceb.nccu.edu.tw/proceedings/APDSI/2004/pdf/018.pdf (дата обращения: 01.07.2012).

[7] Рубашкин В. Ш. Представление и анализ смысла в интеллектуальных информационных системах. М.: Наука, 1989.

[8] Рубашкин В. Ш., Пивоварова Л. М. Онторедактор как комплексный инструмент онтологической инженерии // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции "Диалог 2008". М.: Наука, 2008. С. 453-459.

[9] Сайт онтологии InTez. URL: http://inttez.ru (дата обращения: 02.06.2012).

OWL and KIF ontologies fragments import technology

V. Rubashkin, M. Fadeeva, B. Chuprin

The paper describes the development of the user-friendly environment for interactive selection and transfer of the source ontology fragments in OWL and KIF to the target ontology. This is one of the steps to a unified ontological system creation, which will integrate domain ontologies with a top-level ontology.