УДК 004.04
DOI 10.25205/1818-7900-2018-16-3-113-125

И. А. Корсун [1], **Д. Е. Пальчунов** [1, 2]

[1] *Новосибирский государственный университет*
ул. Пирогова, 1, Новосибирск, 630090, Россия

[2] *Институт математики им. С. Л. Соболева СО РАН*
пр. Академика Коптюга, 4, Новосибирск, 630090, Россия

irina.korsun.nsu@gmail.com, palch@math.nsc.ru

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА ОБРАБОТКИ И ИНТЕГРАЦИИ ЗНАНИЙ НА ОСНОВЕ ТЕХНОЛОГИЙ СЕМАНТИЧЕСКОЙ ПАУТИНЫ *

Статья посвящена разработке методов извлечения из текстов естественного языка определений ключевых понятий предметной области на основе теоретико-модельного подхода. Извлеченная из текстов информация путем преобразования через фрагменты атомарных диаграмм алгебраических систем представляется в виде утверждений в логике описаний (DL). Подобное представление позволяет получать тексты с большей выразительностью по сравнению с алгоритмами, где источником информации являются данные, представленные в виде баз данных или в виде выражений на формализованных языках (например, SQL).

Ключевые слова: онтология, теоретико-модельные методы, фрагменты атомарных диаграмм, определения понятий, извлечение определений, полнота определений понятий, явная определимость, неявная определимость, генерация текстов, порождение фраз естественного языка.

Введение

Современные модели представления информационных ресурсов интенсивно развиваются и внедряются в практику. Важнейшим элементом современных информационных технологий являются онтологии, которые позволяют производить автоматизированную обработку семантики информации с целью ее эффективного использования. С ростом объемов обрабатываемых данных онтологии становятся очень масштабными, что усложняет восприятие конечным пользователем хранящейся в ней информации. Так как в повседневной жизни наиболее распространенной формой представления знаний являются естественно-языковые тексты, проводится все больше исследовательских работ по извлечению описаний объектов OWL-онтологий в виде набора согласованных по смыслу предложений – определений.

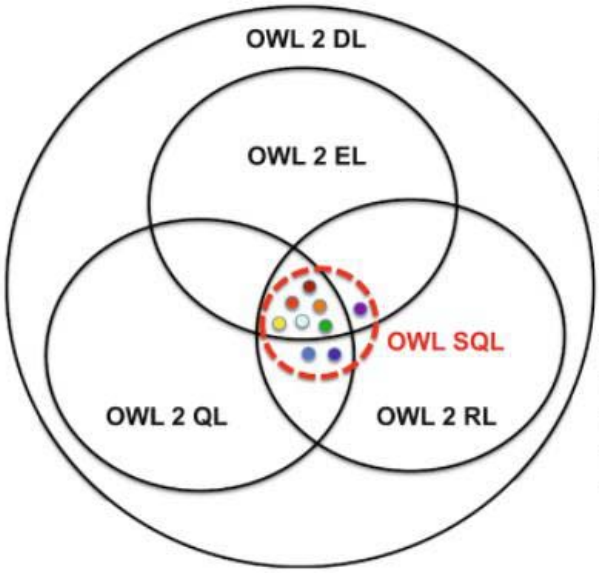
Существует целый каталог систем генераций текстов [1], содержащий краткую информацию обо всех известных (более 340) разработках. Большинство из них относятся к 1963–1980 гг. и имеют алгоритмы шаблонного типа: система хранит уже готовую строку, возможно, с несколькими пропусками, которые заполняются при выдаче сообщения значениями, соответствующими характеру ошибки. Более сложные шаблонные системы дополнительно

* Исследование выполнено при частичной финансовой поддержке Президиума СО РАН (проект «Инженерия интенсиональных онтологий в дедуктивных и информационных системах» Комплексной программы ФНИ СО РАН II.1).

Корсун И. А., *Пальчунов Д. Е.* Интеллектуальная система обработки и интеграции знаний на основе технологий семантической паутины // Вестн. НГУ. Серия: Информационные технологии. 2018. Т. 16, № 3. С. 113–125.

проводят ограниченную лингвистическую обработку генерируемого текста. Примером программ с подобной структурой могут служить следующие: диалоговая система ELIZA [2], система Employee Appraiser (Austin-Haynes) и Performance Now (KnowledgePoint). Шаблонный метод генерации, безусловно, имеет существенный недостаток. Генерируемые тексты сравнимы с естественными до тех пор, пока они имеются в единственном экземпляре. Всё, что написано на базе шаблона, похоже друг на друга за исключением тех частей, куда вставляются параметры.

Также для большинства программ источником информации являются данные, представленные в виде баз данных или в виде выражений на формализованных языках, например SQL. Примеры систем генерации из формального представления: REMIT [3], система Минока [4]. Однако подобный формат представления данных имеет существенные недостатки. Так как SQL полностью отображается на один из профилей OWL, а именно OWL SQL, то можно провести следующий анализ (рис. 1).



	EL	QL	RL
subClass	X	X	X
subProp	X	X	X
domain	X	X	X
range	X	X	X
eqClass	X	X	X
eqProp	X	X	X
inverseProp		X	X
symProp		X	X
transProp	X		X

Рис. 1. Выразительные мощности профилей OWL

Отсюда делаем вывод, что полученные определения из текстов с использованием представления информации на языке SQL достаточно тривиальны и не могут включать в себя более выразительные конструкции.

Таким образом, главная идея настоящей работы заключается в разработке системы извлечения из текстов естественного языка определений ключевых понятий предметной области на основе теоретико-модельного подхода.

Для извлечения из текстов естественного языка определений ключевых понятий предметной области мы используем результаты наших предыдущих исследований. Ранее в [5] был разработан теоретико-модельный подход к извлечению знаний из текстов естественного языка. В основе него лежит представление знаний при помощи конечных фрагментов атомарных диаграмм моделей. Были разработаны и реализованы в виде программной системы методы интерпретации различных частей речи и синтаксических связей с целью автоматического порождения сигнатуры модели [6]. В [7] были предложены алгоритмы отображения бескванторных предложений логик предикатов первого порядка сигнатуры, не содержащей функциональных символов, в логику описаний (DL).

Разрабатываемая на данный момент система сначала извлекает из утверждений на языке DL те, которые имеют отношение к классу или индивидууму, требующему описания в виде текста. Формирует фрагмент OWL онтологии при помощи алгоритмов, также представлен-

ных в [7]. Далее определяет, каким образом выбранная информация будет реализована языковыми средствами в виде предложений на естественном языке. Для этого используются так называемые текстовые планы, представляющие собой последовательность слотов, инструкции, определяющие правила их заполнения, а также лексические словари в виде OWL-онтологий.

Проблемы формального представления определений понятий: явная и неявная определимость

В данной работе мы продолжаем разработку теоретико-модельного подхода к формализации определений ключевых понятий предметной области. В частности, мы продолжаем исследование различных подходов к описанию полноты определений понятий (относительно онтологии, контекста, множества прецедентов предметной области и т. д.), явной и неявной определимости понятий, которое проводилось в [7–10].

В [8] была установлена необходимость неявных определений понятий в формальных глоссариях. В данном параграфе мы изучим взаимосвязь этих результатов с хорошо известной теоремой Бета о неявной определимости.

Необходимые определения можно взять из работ [7; 11].

В [8] было введено понятие формального глоссария.

Определение. Пусть σ – сигнатура. Последовательность предложений $<\varphi_1, \dots, \varphi_n>$ назовем *формальным глоссарием* (*определяющим понятия из* σ), если выполнено:

а) $\sigma(\varphi_1) \subset \sigma(\varphi_1 \,\&\, \varphi_2) \subset \dots \subset \sigma(\varphi_1 \,\&\, \dots \,\&\, \varphi_n) = \sigma$;

б) добавление каждого нового предложения φ_{k+1} консервативно расширяет предыдущий набор предложений $\varphi_1, \dots, \varphi_k$, т. е.

$$\mathrm{Th}(\varphi_1 \,\&\, \dots \,\&\, \varphi_k) = \mathrm{Th}(\varphi_1 \,\&\, \dots \,\&\, \varphi_n) \cap \mathrm{S}(\sigma(\varphi_1 \,\&\, \dots \,\&\, \varphi_k)).$$

Здесь $\mathrm{Th}(\varphi_1 \,\&\, \dots \,\&\, \varphi_n)$ – теория, порожденная (аксиоматизируемая) предложением $(\varphi_1 \,\&\, \dots \,\&\, \varphi_n)$. Мы говорим, что формальный глоссарий $<\varphi_1, \dots, \varphi_n>$ представляет предложение ψ, если $\mathrm{Th}(\psi) = \mathrm{Th}(\varphi_1 \,\&\, \dots \,\&\, \varphi_n)$.

Были доказаны следующие утверждения.

Теорема [8]. Существуют сигнатура σ, состоящая из имен двух понятий, и предложение ψ, определяющее смысл понятий из σ, для которых нет формального глоссария $<\varphi_1, \varphi_2>$, представляющего предложение ψ, такого, чтобы сигнатура $\sigma(\varphi_1)$ состояла из имени одного понятия, а сигнатура $\sigma(\varphi_2)$ – из имен обоих понятий.

Следствие [8]. Не всегда определения понятий могут быть представлены в виде глоссария, определяющего понятия по одному.

Следствие [8]. Не всегда определения понятий могут быть представлены в виде явного глоссария.

На первый взгляд этот результат противоречит известной теореме Бета, которая говорит, что любое неявное определение предиката можно преобразовать в явное.

Определение [11]. Пусть $\Sigma(P)$ – некоторое множество предложений сигнатуры $\sigma \cup \{P\}$, $P \notin \sigma$, т. е. $\Sigma(P) \subseteq S(\sigma)$. Будем говорить, что $\Sigma(P)$ ***неявно определяет*** отношение P, если выполнено

$$\Sigma(P) \cup \Sigma(P') \models (\forall x_1 \dots \forall x_n)\big(P(x_1,\dots,x_n) \leftrightarrow P'(x_1,\dots,x_n)\big).$$

Будем говорить, что $\Sigma(P)$ ***явно определяет*** отношение P, если существует такая формула $\varphi(x_1 \dots x_n) \in S(\sigma)$, что выполнено

$$\Sigma(P) \models (\forall x_1 \dots \forall x_n)\big(P(x_1,\dots,x_n) \leftrightarrow \varphi(x_1,\dots,x_n)\big).$$

Теорема Бета [11]. Множество $\Sigma(P)$ неявно определяет отношение P тогда и только тогда, когда $\Sigma(P)$ определяет отношение P явно.

Таким образом, теорема Бета утверждает, что понятие P, смысл которого неявно задается множеством предложений $\Sigma(P)$, может быть явно определено формулой φ.

В чем разница между понятиями неявной определимости соответственно в [8] и в [11], и почему в данном случае не возникает противоречия между указанными теоремой [8] и теоремой Бета [11]?

Дело в том, что понятие неявной определимости в смысле Бета по существу означает не определение, а задание отношения: происходит полное задание объема определяемого предиката. Формула $\varphi(x_1, ..., x_n)$ выделяет на модели множество кортежей (n), на которых предикат является истинным. Иначе говоря, это явное задание в первую очередь объема предиката (экстента в терминах анализа формальных понятий [12]), а не его содержания (интента). В то же время формальный глоссарий [8], определяющий некоторый набор понятий, ***не задает полностью*** объемы предикатов, соответствующих этим понятиям. Это, в частности, означает, что возможны обогащения одной и той же модели данным новым набором понятий (т. е. новым набором сигнатурных символов) таким, что у соответствующих предикатов будут разные объемы при разных обогащениях модели.

С другой стороны, явная определимость в смысле Бета в точности означает формульную определимость предиката P на моделях, на которых истинно множество предложений $\Sigma(P)$, а именно: в любой формуле $\psi \in F(\sigma \cup \{P\})$ мы можем заменить все вхождения предиката $P(x_1, ..., x_n)$ на формулу $\varphi(x_1, ..., x_n)$, получив в результате формулу $[\psi]^P_\varphi \in S(\sigma)$. При этом для любой модели $\mathfrak{A} \in K(\sigma)$, если $\mathfrak{A} \models \Sigma(P)$, то на модели $\mathfrak{A}$ истинность формул ψ и $[\psi]^P_\varphi$ равносильна:

$$\mathfrak{A} \models (\forall x_1 ... \forall x_k)\left(\psi(x_1,...,x_k) \leftrightarrow [\psi]^P_\varphi(x_1,...,x_k)\right).$$

Таким образом, новое понятие, описываемое предикатом P, добавленным к сигнатуре σ, и определяемое множеством предложений $\Sigma(P)$, на самом деле не дает ничего нового (с точки зрения содержания), а является своего рода «синтаксическим сахаром» – просто сокращением для формулы φ, выразимой в исходном наборе понятий σ.

Именно формульная определимость предиката P обуславливает то, что формула $\varphi(x_1, ..., x_n)$ из теоремы Бета полностью и однозначно задает объем предиката P на моделях, на которых истинно множество предложений $\Sigma(P)$.

Рассмотрим вопрос: а задает ли полностью формула φ смысл, содержание предиката P? А именно, из определения мы имеем

$$\Sigma(P) \models (\forall x_1 ... \forall x_n)\left(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\right).$$

Будет ли верно обратное, т. е. истинно ли предложение

$$(\forall x_1 ... \forall x_n)\left(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\right) \models \Sigma(P)?$$

Отрицательный ответ на этот вопрос дает следующее утверждение.

Предложение. Существуют сигнатура σ и множество предложений $\Sigma(P)$ такие, что

$$\Sigma(P) \cup \Sigma(P') \models (\forall x_1 ... \forall x_n)\left(P(x_1,...,x_n) \leftrightarrow P'(x_1,...,x_n)\right),$$

$$\Sigma(P) \models (\forall x_1 ... \forall x_n)\left(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\right),$$

но неверно

$$(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big) \models \Sigma(P).$$

Таким образом, предложение $(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big)$ всегда полностью задает объем предиката P, но не всегда полностью задает его смысл, содержание предиката P (если считать множество предложений $\Sigma(P)$ неявным описанием смысла понятия, обозначаемого предикатом P).

Здесь можно было бы возразить, что мы можем добавить к $\Sigma(P)$ что угодно, и при этом сохранится истинность утверждения

$$\Sigma(P) \models (\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big),$$

а утверждение

$$(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big) \models \Sigma(P)$$

перестанет быть верным, даже если оно было верно до того. В связи с этим мы можем сформулировать вопрос: а следует ли $\Sigma(P)$ из $(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big)$ вместе с множеством всех следствий множества предложений $\Sigma(P)$ в сигнатуре σ, т. е. вместе с множеством предложений $\mathrm{Th}\big(\Sigma(P)\big) \cap S(\sigma)$? Положительный ответ на этот вопрос дает следующее утверждение.

Предложение. Пусть $\Sigma(P) \models (\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big)$, $\varphi \in S(\sigma)$ и $\Sigma(P) \subseteq$ $\subseteq S\big(\sigma \cup \{P\}\big)$. Тогда выполнено

$$\mathrm{Th}\big(\Sigma(P)\big) \cap S(\sigma) \cup \big\{(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big)\big\} \models \Sigma(P).$$

Тем не менее сделанное нами выше утверждение, что формула

$$(\forall x_1 ... \forall x_n)\big(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)\big)$$

полностью задает объем предиката P, но не всегда полностью задает его смысл, остается верным. Это иллюстрирует следующий пример.

Пример. Пусть $\sigma = \varnothing$, а $\Sigma(P) = \big\{(\forall x)P(x),\ (\forall x \forall y)\big(\big(P(x) \,\&\, P(y)\big) \to (x = y)\big)\big\}$. Тогда $\Sigma(P) \models (\forall x)\big(P(x) \leftrightarrow \varphi(x)\big)$, где $\varphi(x) = (x = x)$. С другой стороны, формула $\varphi(x)$ не является полным определением смысла предиката $P(x)$, задаваемого множеством предложений $\Sigma(P)$. А именно, из $\varphi(x)$ не следует, что предикат $P(x)$ выделяет ровно один элемент (т. е. предикат $P(x)$ истинен ровно на одном элементе); однако это свойство предиката P следует из его неявного определения $\Sigma(P)$. Таким образом, «явное определение» $(\forall x)\big(P(x) \leftrightarrow \varphi(x)\big)$ полностью задает объем предиката P, но не определяет его смысл.

Стало быть, из теоремы Бета не следует, что ***любое неявное определение смысла предиката можно свести к его явному определению***.

Можно было бы сказать, что ***любое неявное определение объема предиката можно свести к его явному определению***, но это также является не совсем верным. Дело в том, что у неявного определения $\Sigma(P)$ и у явного определения $(\forall x)\big(P(x) \leftrightarrow \varphi(x)\big)$ классы моделей разные. Действительно, любую модель сигнатуры $\sigma(\varphi)$ можно доопределить до модели сигнатуры $\sigma(\varphi) \cup \{P\}$ так, чтобы выполнялось утверждение $(\forall x)\big(P(x) \leftrightarrow \varphi(x)\big)$. С другой стороны, множество предложений $\Sigma(P)$ из приведенного выше примера может быть истинно только на одноэлементных моделях.

«Неявное» определение $\Sigma(P)$ объема предиката P в теореме Бета на самом деле сводится не к предложению $(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n))$, а к множеству предложений $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma)) \cup \{(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n))\}$. Множества предложений $\Sigma(P)$ и $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma)) \cup \{(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n))\}$, как было показано выше, семантически эквивалентны, т. е. на них истинны одни и те же модели. Таким образом, «неявное» определение $\Sigma(P)$ ***объема*** предиката P сводится к «явному» определению его объема $(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n))$, но по модулю множества предложений $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma))$ сигнатуры σ. Это множество предложений задает класс моделей сигнатуры σ, на которых далее задается предикат P через его объем.

С другой стороны, является неверным утверждение, что «неявное» определение $\Sigma(P)$ ***смысла*** предиката P сводится к «явному» определению его смысла

$$(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n)),$$

по модулю множества предложений $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma))$. В рассмотренном выше примере, существенное свойство предиката P – то, что он истинен только на одном элементе (т. е. в определенном смысле задает константу – имя этого элемента), – не содержится в «явном» определении $(\forall x)(P(x) \leftrightarrow \varphi(x))$. В этом примере множество предложений $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma))$ может быть истинно только на одноэлементных моделях. Поэтому из $(\mathrm{Th}(\Sigma(P)) \cap S(\sigma))$ и $(\forall x)(P(x) \leftrightarrow \varphi(x))$ дедуктивно следует предложение

$$(\forall x \forall y)((P(x) \,\&\, P(y)) \rightarrow (x = y)).$$

При этом очевидно, что в множестве предложений

$$(\mathrm{Th}(\Sigma(P)) \cap S(\sigma)) \cup \{(\forall x_1 ... \forall x_n)(P(x_1,...,x_n) \leftrightarrow \varphi(x_1,...,x_n))\}$$

свойство $(\forall x \forall y)((P(x) \,\&\, P(y)) \rightarrow (x = y))$ предиката P содержится неявно (!), в то время как в «неявном» определении $\Sigma(P) = \{(\forall x)P(x),\ (\forall x \forall y)((P(x) \,\&\, P(y)) \rightarrow (x = y))\}$ это свойство содержится явно. Получается, что в данном примере теорема Бета сводит не неявное определение смысла предиката к явному определению, а наоборот, явное определение к неявному.

Таким образом, с точки зрения смысла предиката P, а не его объема, понятия явного и неявного определения, используемые в теореме Бета, не являются полностью адекватными и требуют пересмотра.

Алгоритм формирования описания объекта из фрагмента OWL-онтологии

Общая и полная схема генерации без детализации происходящих процессов состоит из трех основных блоков:

1) планирование содержания текста – построение структуры текста;
2) микропланирование – построение планов предложений;
3) языковое оформление – реализация построенных планов предложений соответствующими грамматическими структурами.

В прикладных системах генерации к этим трем этапам часто добавляется четвертый этап – физическое представление, на котором производится форматирование текста согласно выбранному формату (PDF, HTML и др.).

Планирование состоит из следующих частей: выбор контента (информационного содержания), в котором система выбирает информацию для передачи в следующий этап конвейера, и текстовое планирование, где она планирует структуру текста, который будет сгенерирован.

Когда систему просят описать целевой объект, она прежде всего извлекает из онтологии все OWL-утверждения, связанные с объектом вниз по иерархии вплоть до уровня, указанного в настройках пользователя. Затем алгоритмы преобразуют извлеченный набор OWL операторов в триплеты (см. таблицу).

OWL-утверждения и соответствующие им формы триплетов

OWL-утверждения	Триплеты
Object – экземпляр	
ClassAssertion(Class, object)	<object, instanceOf, Class>
ClassAssertion(ObjectComplementOf(Class), object)	<object, not(instanceOf), Class>
ClassAssertion(ObjectOneOf(indiv1,indiv2) object)	<object, one of, or(indiv1, indiv2, …)>
ClassAssertion(ObjectHasValue(ojbProp, indiv) object)	<object, objProp, indiv>
ClassAssertion(ObjectHasValue(dataProp,dataValue) object)	<object, dataProp, indiv>
ClassAssertion(ObjectHasSelf(objProp) object)	<object, objProp, object>
ClassAssertion(ObjectMaxCardinality(number,prop,[Class]) object)	<object,maxCardinality(prop),number[:Class]>
ClassAssertion(ObjectMinCardinality(number,prop,[Class]) object)	<object,minCardinality(prop),number[:Class]>
ClassAssertion(ObjectExactCardinality(number,prop,[Clas s]) object)	<object,exactCardinality(prop),number[:Class]>
ClassAssertion(ObjectSomeValuesVrom(objProp,Class) object)	<object, someValuesFrom(objProp),Class>
ClassAssertion(ObjectAllValuesVrom(objProp,Class) object)	<object, allValuesFrom(objProp),Class>
ClassAssertion(ObjectIntersectionOf(C1,C2,…) object)	convert(ClassAssertion(C1, object)) convert(ClassAssertion(C2, object))…
ClassAssertion(ObjectUnionOf (C1,C2,…)object)	or(convert(ClassAssertion(C1,object)),convert (ClassAssertion(C2, object)),…)
ObjectPropertyAssertion(objProp,object,indiv)	<object, objProp, indiv>
DataPropertyAssertion(dataProp,object,dataValue)	<object, dataProp, dataValue>
NegativeObjectPropertyAssertion(objProp,object,indiv)	<object, not(objProp), indiv>
NegativeDataPropertyAssertion(dataProp,object,dataValue)	<object, not(dataProp), dataValue>
DifferentIndividuals(object,indiv)	<object, differentIndividuals, indiv>

Окончание таблицы

OWL-утверждения	Триплеты
DifferentIndividuals(indiv,object)	<object, differentIndividuals, indiv>
SameIndividual(object,indiv)	<object, sameIndividuals, indiv>
SameIndividualindiv object)	<object, sameIndividuals, indiv>
Object – класс	
EquivalentClasses(Object, Class/expr)	convert (SubClassOf (Object, Class/expr)
EquivalentClasses(Class/expr, Object)	convert (SubClassOf (Object, Class/expr)
SubClassOf(Object, Class)	<Object, isA, Class>
SubClassOf(Object, ObjectComplementOf(Class))	<Object not(isA), Class>
SubClassOf(Object,ObjectOneOf(indiv1, indiv2 …)	<Object, one of, or(indiv1, indiv2 …)>
SubClassOf(Object,ObjectHasValue(objProp, indiv))	<Object, objProp, indiv>
SubClassOf(Object,ObjectHasValue(dataProp, dataValue)	<Object, dataProp, dataValue>
SubClassOf(Object, ObjectHasSelf(objProp))	<Object, objProp, Object>
SubClassOf(Object, ObjectMaxCardinality(number,prop, [Class]))	<Object,maxCardinality(prop),number[:Class]>
SubClassOf(Object,ObjectMinCardinality(number,prop,[Class]))	<Object,minCardinality(prop),number[:Class]>
SubClassOf(Object, ObjectExactCardinality(number,prop,[Class]))	<Object,exactCardinality(prop),number[:Class]>
SubClassOf(Object, ObjectSomeValuesFrom(objProp, Class))	<Object, someValuesFrom(objProp), Class>
SubClassOf(Object, ObjectAllValuesFrom(objProp, Class))	<Object, allValuesFrom(objProp), Class>
SubClassOf(Object,ObjectIntersectionOf (C1/expr, C2/expr …)	convert(SubClassOf(C1/expr, Object)) convert(SubClassOf(C2/expr, Object)) …
SubClassOf(Object,ObjectUnionOf(C1/expr, C2/expr …)	or(convert(SubClassOf(C1/expr,Object)) convert(SubClassOf(C2/expr, Object))…)
DisjointClasses(Object, Class)	<Object, not(isA), Class>
DisjointClasses(Class, Object)	<Object, not(isA), Class>

Текстовое планирование представляет собой упорядочивание предложений по тематике с использованием дополнительных ресурсов системы.

Для работы метода группировки предложений по темам автор онтологии может определить разделы и назначить каждому из них свойства. Если разделы будут определены, то на этапе текстового планирования система распределит триплеты по соответствующим группам. Также эксперт может задать порядок самих разделов, что приводит к дополнительной итерации упорядочивания. Все секции, принадлежность свойств секциям и порядок разделов и свойств определяются в зависимых от онтологии ресурсах генерации. В нашем случае ресурсы также имеют представление OWL-онтологий.

В общем случае алгоритм сортировки имеет следующий вид:

Процедура **orderMessageTriples**

ВХОД:

t[0]: целевой объект

t[1], ..., t[n]: объекты второго уровня
L[0]: неотсортированный список триплетов, описывающих t[0]
...
L[n]: неотсортированный список триплетов, описывающих t[n]
SMap: Карта соответствий между отношениями (предикатами) и наименованием секции
SOrder: Отсортированные наименования секций
POrder: Частично отсортированный массив отношений:

ВЫХОД:
Отсортированный список триплетов

ШАГИ:
```
от  i := 0 до  n {
        orderMessageTriplesAux(L[i], SMap, SOrder, POrder)
}
от  i := 1 до  n {
        insertAfterFirst(, L[0], L[i])
}
Вернуть:  L[0]
```

Процедура **orderMessageTriplesAux**

ВХОД:
L: Неотсортированный список триплетов об одном объекте
SMap: Карта соответствий между отношениями (предикатами) и наименованием секции
SOrder: Отсортированные наименования секций
POrder: Частично отсортированный массив отношений

ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ:
S[1], ..., S[k]: списки с триплетами по каждой секции

ВЫХОД:
Отсортированный список триплетов об одном объекте

ШАГИ:
```
<S[1], ..., S[k]> := splitInSections(L, SMap)
от  i := 1 до  k {
        S[i] := orderTriplesInSection(S[i], POrder)
}
<S[1], ..., S[k]> := reorderSections(S[1], ..., S[k], SOrder)
Вернуть: concatenate(S[1], ..., S[k])
```

Микропланирование – это блок, который позволяет от предметных знаний перейти к языковым. В нем решается, каким образом выбранная информация будет реализована языковыми средствами в виде предложений на естественном языке. В нашем случае микропланирование состоит из трех подэтапов.

1. Лексикализация концептов сообщения, т. е. выбор подходящих слов для выражения выбранного в них содержания.

Для каждого глагола, существительного или прилагательного, которые эксперт хочет использовать в планах предложений, должны быть представлены соответствующие «синтаксические формы». Большинство синтаксических форм слов русского языка можно автоматически создавать из базовых форм, используя простые правила морфологии. Расширение нашей системы подобными компонентами будет рассмотрено в дальнейших работах.

2. Агрегирование сообщений до структур, соответствующих отдельным предложениям создаваемого текста.

Системы генерации текстов часто объединяют предложения, полученные после обработки триплетов, в более длинные, чтобы улучшить читаемость. Агрегация предложения выполняется при помощи набора определенных правил, которые применяются к структурам, полученным в ходе работы текстового планировщика. Другими словами, они применяются к предложениям, уже отсортированным и сгруппированным по семантике. Действительно,

объединение не связных по смыслу предложений будет звучать неестественно. Так как агрегация происходит по четко заданным шаблонам, ее возможности полностью зависят от результата работы предыдущего этапа. Рассмотрим теперь каждое из используемых правил.

Исключение повторов одного имени существительного с несколькими прилагательными. Последовательность триплетов сообщений в форме <S, P, O1>, ..., <S, P, On> будет объединена в один триплет <S, P, и (O1, ... On)>.

Объединение последовательности триплетов, **выраженных в формах <S; M(P); O> и <S; P; O>**, **для одинаковых S и P**, причем M – это один из minCardinality, maxCardinality, exactCardinality. Применение данного правила снимает ограничение на maxMessagesPer Sentence.

Модель продается максимум в трех странах. Модель продается минимум в трех странах. Модель продается в Испании, Италии и Греции => Модель продается в трех странах: Испании, Италии и Греции.

Объединение предложений вида <S, **instanceOf**, **C> или <S**, **isA**, **C> с предложением <S**, **P**, **O>**, для того же S, где P является свойством онтологии.

Ромашка – это цветок. Ромашка растет в поле => Ромашка – это цветок, который растет в поле.

Объединение нескольких предложений, **каждое из которых является выражением триплета <S; Pi; Oi>**, **для одного и того же S**, где Pi – это свойство онтологии. В каждом из последующих или предшествующих предложений должен быть сам объект, за которым сразу следует только прилагательное. Прилагательные вписываются в результирующее предложение, сохраняя их порядок, определенный планировщиком.

Это мотоцикл. Он красный. Он дорогой. => Это красный, дорогой мотоцикл.

Объединение последовательности предложений, **связанных с одним и тем же глаголом**, **каждый из которых выражает триплет <S**, **Pi**, **Oi>**, где S одинаковый во всех предложениях, а Pi – это свойства онтологии. Результирующее предложение может быть сформировано путем однократного использования существительного и глагола. Соединительный союз «и» вставляется перед последним существительным.

Напиток имеет средний вкус. Напиток имеет умеренный аромат => Напиток имеет средний вкус и умеренный аромат.

Объединение последовательности предложений, **не связанных с одним и тем же глаголом**, **каждое из которых является представлением триплета <S**, **Pi**, **Oi>**, где S одинаковое во всех тройках, а Pi – свойства онтологии.

Вино сухое. Вино имеет средний аромат. Оно произведено в Италии. => Вино сухое, имеет средний аромат и произведено в Италии.

Архитектура системы

Для программного продукта были выработаны следующие требования:

1) форма десктопного приложения;
2) работа с уже существующей системой логического вывода;
3) работа с уже существующей системой получения фрагментов атомарных диаграмм и алгоритмами получения DL утверждений;
4) удобство и простота использования.

Под эти требования подходило большое число языков программирования, однако основной выбор пал на объектно-ориентированные. Также были сформулированы не функциональные требования:

1) быстрота и удобство разработки;
2) возможность устройства модульной архитектуры;
3) знания и умения разработки на выбранном языке программирования.

Поскольку быстрота работы программы в данном случае не является существенным требованием, был выбран кроссплатформенный язык программирования Java, так как он удовлетворил всем поставленным требованиям.

Структура программы представлена следующими директориями.

- Основные классы программы:

1) пакет, отвечающий за возможность графической работы;
2) пакет представления данных;
3) пакет классов, реализующих основную логику приложения.

- В программе используются дополнительные библиотеки:

1) Apache Commons;
2) OWLAPI – библиотека для работы с OWL-онтологиями;
3) Jena – библиотека для работы с OWL-онтологиями;
4) Hermit – машина логического вывода.

Основной модуль разработан с использованием архитектурного шаблона Pipeline. На рис. 2 представлена его use-case диаграмма, отображающая основные возможные действия пользователя.

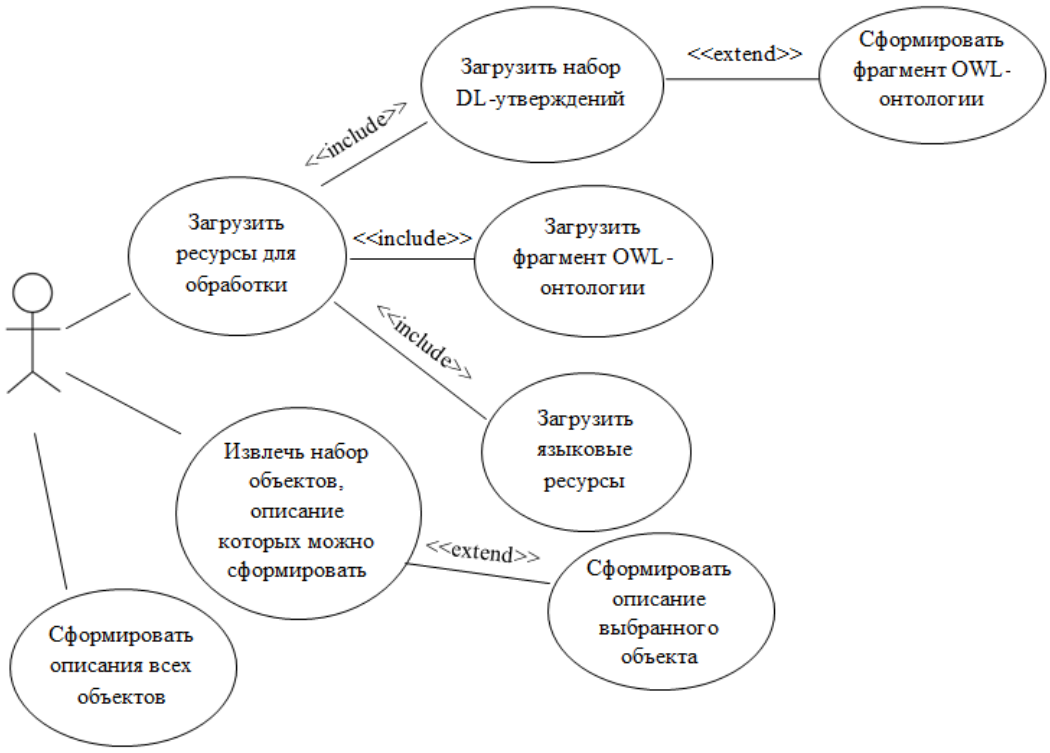


Рис. 2. Use-case диаграмма основного модуля системы

В рамках дополнительных возможностей все загружаемые элементы можно проверить машиной логического вывода Hermit.

Заключение

В работе изучены вопросы полноты определений ключевых понятий предметной области, явной и неявной определимости ключевых понятий. Исследована полнота определений понятий относительно объема и содержания соответствующих сигнатурных предикатов.

Разработаны методы интеграции фрагментов определения ключевого понятия, извлеченных из текстов естественного языка. Методы интеграции частей определения понятия основаны на представлении знаний, извлеченных из текстов естественного языка, в виде

фрагментов атомарной диаграммы алгебраической системы и на преобразовании фрагментов атомарной диаграммы в логику описаний (DL) и в OWL. Полученные OWL-спецификации погружаются в OWL-онтологию, соответствующую данной предметной области.

Разработаны методы интеграции OWL-спецификаций, представленных в OWL-онтологии, и порождения по ним фраз естественного языка.

Список литературы

1. *Bateman J., Zock M.* List of Natural Language Generation Systems. URL: http://www.fb10.-uni-bremen.de/anglistik/langpro/NLG-table/NLG-table-root.html.

2. *Shah H., Warwick K., Vallverdú J., Wu D.* Can machines talk? Comparison of Eliza with modern dialogue systems // Computers in Human Behavior. 2006. No. 58. P. 278–95.

3. *Lowden B. G. T., Roeck A. N. de.* The REMIT system for paraphrasing relational query expressions into natural language // Proc. Int'l. Conf. on Very Large Data Bases. Kyoto, Japan, 1986.

4. *Minock M.* STEP A Natural Language Interface to Database. URL: http://www.cs.umu.se/~mjm/

5. *Махасоева О. Г., Пальчунов Д. Е.* Автоматизированные методы построения атомарной диаграммы модели по тексту естественного языка // Вестн. НГУ. Серия: Информационные технологии. 2014. Т. 12, № 2. С. 64–73.

6. *Махасоева О. Г., Пальчунов Д. Е.* Программная система построения атомарной диаграммы модели по тексту естественного языка. Св-во о государственной регистрации программы для ЭВМ № 2014619198, зарегистрировано 10.09.2014.

7. *Корсун И. А., Пальчунов Д. Е.* Теоретико-модельные методы извлечения знаний о смысле понятий из текстов естественного языка // Вестн. НГУ. Серия: Информационные технологии. 2016. Т. 14, № 3. С. 34–48.

8. *Пальчунов Д. Е.* Моделирование мышления и формализация рефлексии. Ч. 2. Онтологии и формализация понятий // Философия науки. 2008. № 2 (37). С. 62–99.

9. *Корсун И. А., Пальчунов Д. Е.* Методы извлечения и формального представления онтологических знаний из текстов естественного языка // Знания – Онтологии – Теории (ЗОНТ-2017): Материалы Всерос. конф. с междунар. участием. 2017. Т. 2. С. 21–25.

10. *Капустина А. И., Пальчунов Д. Е.* Разработка онтологической модели тарифов и услуг сотовой связи, основанной на логически полных определениях понятий // Вестн. НГУ. Серия: Информационные технологии. 2017. Т. 15, № 2. С. 34–46.

11. *Кейслер Г., Чэн Ч. Ч.* Теория моделей. М.: Мир, 1977.

12. *Ganter B., Wille R.* Formal Concept Analysis. Mathematical foundations. Berlin, Heidelberg: Springer-Verlag, 1999.

Материал поступил в редколлегию 27.06.2018

I. A. Korsun [1], **D. E. Palchunov** [1, 2]

[1] *Novosibirsk State University*
1 Pirogov Str., Novosibirsk, 630090, Russian Federation

[2] *S. L. Sobolev Institute of Mathematics SB RAS*
4 Academician Koptyug Ave., Novosibirsk, 630090, Russian Federation

irina.korsun.nsu@gmail.com, palch@math.nsc.ru

AN INTELLECTUAL SYSTEM FOR PROCESSING AND INTEGRATING KNOWLEDGE BASED ON SEMANTIC WEB TECHNOLOGIES

The paper is devoted to the development of model-theoretic methods of concepts definitions extraction from the natural language texts. The information extracted from texts is represented in the form of statements on the Description Logic language (DL) by transformation through fragments of atomic diagrams of algebraic systems. Such a representation allows you to get more expressive

texts, in contrast to algorithms, where information is represented in a database or by expressions in a formal language (for example, SQL).

Keywords: ontology, model-theoretic methods, fragments of atomic diagrams, concept definitions, definition extraction, completeness of definitions of concepts, explicit definability, implicit definability, text generation, natural language generation.

References

1. Bateman J., Zock M. List of Natural Language Generation Systems. URL: http://www.fb10.-uni-bremen.de/anglistik/langpro/NLG-table/NLG-table-root.html.

2. Shah H., Warwick K., Vallverdú J., Wu D. Can machines talk? Comparison of Eliza with modern dialogue systems. *Computers in Human Behavior*, 2006, no. 58, p. 278–95.

3. Lowden B. G. T., Roeck A. N. de. The REMIT system for paraphrasing relational query expressions into natural language. *Proc. Int'l. Conf. on Very Large Data Bases*. Kyoto, Japan, 1986.

4. Minock M. STEP A Natural Language Interface to Database. URL: http://www.cs.umu.se/~mjm/.

5. Makhasoeva O. G., Palchunov D. E. Semi-automatic methods of a construction of the atomic diagrams from natural language texts. Vestnik NSU. Series: Information Technologies, 2014, vol. 12, no. 2, p. 64–73. ISSN 1818-7900 (in Russ.)

6. Makhasoeva O. G., Palchunov D. E. Program system for the construction of the atomic diagram of a model from natural language texts. Certificate of the State Registration of the computer program. No. 2014619198, registered 10.09.2014. (in Russ.)

7. Korsun I. A., Palchunov D. E. Model-Theoretic Methods of Extraction of Knowledge on the Meaning of Concepts from the Natural Language Texts. *Vestnik NSU. Series: Information Technologies*, 2016. vol. 14, no. 3, p. 34–48. ISSN 1818-7900 (in Russ.)

8. Palchunov D. E. Modeling of reasoning and formalization of reflection II: Ontologies and formalization of concepts. *Filosofiya nauki*, 2008, no. 2 (37), p. 62–99. (in Russ.)

9. Korsun I. A., Palchunov D. E. Methods of extraction and formal representation of ontological knowledge from natural language texts. *Proc. of the All-Russian Conference with International Participation «Knowledge-Ontology-Theory» (KONT-17)*. Novosibirsk, 2017, vol. 2, p. 21–25.

10. Kapustina A. I., Palchunov D. E. The development of ontological model of tariffs and services of mobile operator, based on logically complete definitions of concepts. *Vestnik NSU. Information Technologies*, 2017, vol. 15, no. 2, p. 34–46. ISSN 1818-7900 (in Russ.)

11. Chang C. C., Keisler H. J. Model Theory. *Studies in Logic and the Foundations of Mathematics*, 1977.

12. Ganter B., Wille R. Formal Concept Analysis. Mathematical foundations. Berlin, Heidelberg, Springer-Verlag, 1999.

For citation:

Korsun I. A., Palchunov D. E. An Intellectual System for Processing and Integrating Knowledge Based on Semantic Web Technologies. *Vestnik NSU. Series*: *Information Technologies*, 2018, vol. 16, no. 3, p. 113–125. (in Russ.)

DOI 10.25205/1818-7900-2018-16-3-113-125