УДК 004.822:004.832.32

А.С. Поспешный, С.Г. Стиренко

# ЭФФЕКТИВНЫЙ АЛГОРИТМ РАССУЖДЕНИЙ ДЛЯ OWL 2 EL ОНТОЛОГИЙ С ТИПИЗИРОВАННЫМИ ВЫРАЖЕНИЯМИ НА БАЗЕ ЛОГИЧЕСКОГО ПРОЦЕССОРА ELK

*Аннотация:* Многократно и регулярно высказывалось мнение, что типизированные выражения с использованием конкретных типов данных будут иметь первостепенную роль в Семантической паутине и необходимы для составления практически-применимых баз знаний. В этой статье мы представляем модификацию высокопроизводительного логического процессора ELK с поддержкой типизированных свойств в рамках профиля OWL 2 EL. Проведённые испытания показали исключительную скорость классификации больших онтологий со значительным количеством экземпляров и типизированных свойств, что открывает новые перспективы для применения логических процессоров на практике.

*Ключевые слова:* онтология, конкретные типы данных, логический анализ, EL++, ELK

## OWL 2 EL Онтологии

OWL (Web Ontology Language) — это формальный язык описания онтологий, основанный на дескриптивной логике с несколькими вариантами синтаксиса (OWL/XML, RDF/XML, N3 и др.). Принятие Консорциумом Всемирной паутины (W3C) этого стандарта наряду с пересмотренной версией стандарта RDF завершило формирование фундамента для Семантической паутины - новой информационной архитектуры которая призвана сделать размещённую в Интернет информацию более понятной для компьютеров.

Язык OWL предназначен для приложений, задача которых состоит не в представлении информации в удобном для человека виде, а в обработке содержимого информационных ресурсов. При создании этого языка был использован и обобщен опыт предшествующих разработок в этой сфере.

При стандартизации второй, текущей на данный момент, версии языка OWL были сформулированы отдельные его профили, более известные как фрагменты или подмножества языка в вычислительной логике. Применение профилей, как правило, происходит тогда, когда возможно принести в жертву выразительность языка в обмен на значительный рост производительности логической обработки написанных на нём онтологий.

OWL 2 EL, один из таких профилей, базируется на языке дескриптивной логики *EL++*, имеющий хорошие вычислительные свойства и при этом обладает достаточной практической экспрессивностью. EL



 ISSN 1560-8956

профиль особенно хорошо подходит для приложений оперирующих очень большим количеством классов и свойств, так как основные задачи логического анализа могут быть выполнены за полиномиальное время по отношению к размеру обрабатываемой онтологии.

Доступность алгоритма рассуждения, его производительность и масштабируемость стали залогом стремительного развития онтологий на базе профиля OWL 2 EL. Особенно большую популярность такие онтологии получили в области медицины и генетики.

Вслед за онтологиями появилось множество логических анализаторов способные за приемлемое время производить их классификацию. Среди них особо стоит упомянуть CB, CEL, jCEL, Snoroket и ELK. Исчерпывающий обзор логических анализаторов для OWL EL онтологий представлен в [1].

Упомянутый выше логический анализатор ELK благодаря своей продуманной масштабируемой архитектуре стал в основу нашей работы.

## Логический процессор ELK

ELK является свободно распространяемым высокопроизводительным логическим анализатором с открытым исходным кодом для онтологий OWL 2 EL профиля. Исключительная высокая производительность логического анализа достигается за счёт применения оптимизированного параллельного накопительного алгоритма для рассуждений, работа которого, в общих чертах, заключается в итеративном применении к множеству исходных аксиом трансформирующих правил, порождающих новые аксиомы-следствия.

EKL обладает гибкой модульной архитектурой и может быть использован в разнообразных конфигурациях (рис. 1), в частности возможно использование ELK как утилиты, вызываемой из командной строки операционной системы, как библиотеки, интегрируемой в приложение посредством интерфейса OWL API или как подключаемого расширения к редактору онтологий и баз знаний Protege.

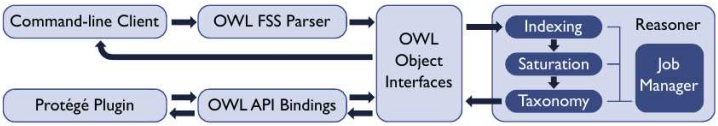


Рис. 1 – Базовые модули ELK и информационные потоки при классификации онтологии

Исчерпывающее описание алгоритмов используемых в ELK, особенности их реализации и доказательства их корректности и полноты можно найти в работах [2–4].

В таблице 1 приведены все OWL выражения, поддерживаемые ELK версии 0.3 — последней на момент написания этой статьи.

ELK поддерживает основные задачи логической обработки онтологий, в том числе проверку на противоречия, TBox-классификацию, и вычисление конкретного типа именованных экземпляров (ABox-реализация). ELK на прямую не поддерживает комплексные (анонимные) выражения, однако достаточно легко добавить подобную поддержку путём добавления к исходной онтологии нового именованного класса, эквивалентного анонимному выражению, для каждого такого выражения.

По результатам тестирования [4] ELK способен классифицировать онтологию SNOMED CT, содержащую более 300 000 концептов, за несколько секунд, лидируя в своём классе.

Однако применив ELK к нашей задаче мы столкнулись с одним из его главных недостатков — отсутствие поддержки типизированных свойств. Недавний анализ использования OWL онтологий в сети Интернет [5] показал, что типизированные свойства остаются чрезвычайно важным элементом используемых в Сети онтологий, несмотря на их ограниченную поддержку распространёнными логическими анализаторами. Многократно высказывалось мнение, что типизированные выражения зачастую являются основной составляющей баз знаний, а их доля в планируемой Семантической Паутине может составить более половины всех выражений.

Таблица 1.
Поддерживаемые OWL выражения в ELK 0.3.0

| **Тип OWL выражений** | **Выражение** |
|---|---|
| Аксиомы | SubClassOf |
| | EquivalentClasses |
| | DisjointClasses |
| | SubObjectPropertyOf |
| | EquivalentObjectProperties |
| | TransitiveObjectProperty |
| | ReflexiveObjectProperty |
| | ObjectPropertyDomain |
| | ClassAssertion |
| | ObjectPropertyAssertion |
| Классы | owl:Thing |
| | owl:Nothing |
| | ObjectIntersectionOf |
| | ObjectSomeValuesFrom |
| | ObjectHasValue |
| | DataHasValue |
| Объектные свойства | ObjectPropertyChain |
| Экземпляры | NamedIndividual |

 ISSN 1560-8956

## Типы данных, используемые в OWL 2 EL онтологиях

Согласно спецификации языка OWL 2, профиль OWL 2 EL предусматривает использование 19-и типов данных, большинство из которых определённо в рамках спецификации XSD - языка описания структуры XML - документов. Рисунок 2 резюмирует все поддерживаемые в рамках профиля типы данных, устанавливает их наследственность и разрешённые ограничительные аспекты.

Аспектом (фасетом) называется часть определения, служащая для задания набора значений простого типа и определяющая набор его допустимых значений.

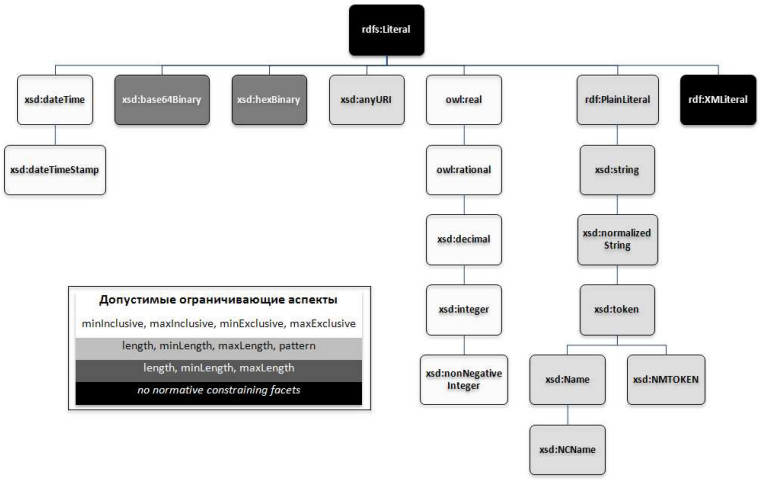


Рис. 2 – Допустимые типы данных в онтологиях OWL 2 EL профиля

Важно отметить, что профиль OWL 2 EL не предусматривает использование следующих типов данных: *xsd:double, xsd:float, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:language и xsd:boolean*. Причина этого заключается в том, что список разрешённых профилем типов был составлен таким образом, что бы пересечение пространств значений каждого из них с любым другим из типов было либо пустым, либо бесконечным. Подобное ограничение помогает обеспечить необходимые вычислительные характеристики логического анализа EL онтологий.

В таблице 2 приведено краткое описание каждого разрешённого типа данных и указано источник, подробно их определяющий.

Согласно спецификации, OWL 2 EL онтологии могут использовать упомянутые выше типы данных в экзистенциальных квантифика-

торах *DataSomeValuesFrom* и *DataHasValue,* при утверждении экземпляров (DataPropertyAssertion, NegativeDataPropertyAssertion), пересечении диапазонов данных (*DataIntersectionOf*), перечислении единственного
литерала (*DataOneOf*) и в некоторых других выражениях.

## Поддержка типизированных свойств в ELK

Для реализации поддержки вышеупомянутых аксиом с типизированными выражениями мы добавили новые правила логического вывода, описанные на Рисунке 3,

$$\frac{}{A \sqsubseteq \bot} \quad A \sqsubseteq \exists r_+ \in O, \quad r_+ \rightarrow_D \bot$$

$$\frac{A \sqsubseteq \exists r^+}{A \sqsubseteq B} \quad \exists r_- \sqsubseteq B \in O, \quad r_+ \rightarrow_D r_-$$

$$\frac{A \sqsubseteq B}{A \sqsubseteq \exists r^+} \quad B \sqsubseteq \exists r_+ \in O$$

Рис. 3 – Правила обработки типизированы выражений

где O – онтология, $A, B \in N_C$, $r \in N_R$, $N_R$ – исчислимо бесконечное множество атомарных ролей, $N_C$ – множество атомарных концептов, выражение $r_+ \rightarrow_D \bot$ означает, что из ограничения следует пустое множество, в то время как выражение $r_+ \rightarrow_D r_-$ означает, что ограничение справа от удовлетворяет ограничение слева от него. Знаки $+$ и $-$ указывают на то, что выражение встречается справа и слева от $\sqsubseteq$ соответственно.

Следующей задачей было проиндексировать типизированные выражения исходной онтологии таким образом, что бы максимально быстро производить поиск подходящих выражений $\exists r \sqsubseteq B \in$ O и разрешение операции $r_+ \rightarrow_D r_-$.

Было решено хранить типизированные выражения следующим образом (рис. 4). Для каждого типизированного свойства в онтологии хранится контейнер корневых типов.

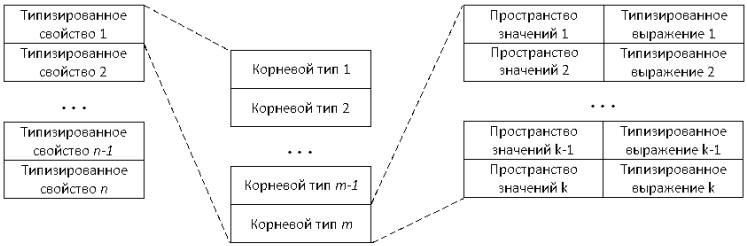


Рис. 4 – Реестр типизированных выражений

 ISSN 1560-8956

Таблица 2

Описание типов данных, используемых в OWL 2 EL онтологиях

| Категория | Тип данных | Описание | Пример | Источник |
|---|---|---|---|---|
| Числа | owl:real | Произвольное вещественное число. Данный тип не имеет лексической формы. | - | OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax[6] |
| | owl:rational | Рациональное число, заданное в форме обыкновенной дроби: ’числитель’ / ’знаменатель’ | 1 / 3, -34 / 2 | |
| | xsd:decimal | Десятичное число произвольной точности заданное в виде последовательности десятичных цифр и дробной части, отделённой точкой. Лексема может содержать знак + или - в начальной позиции | -1.23, 126787.543233, +100000.00, 210 | XML Schema Part 2: Datatypes Second Edition[7] |
| | xsd:integer | Целое число, заданное в виде последовательности десятичных цифр. Лексема может содержать знак + или - в начальной позиции | -126789, -1, 0, +1, 126789 | |
| | xsd: nonNegativeInteger | Целое число, большее или равное нулю | 0, 1, 126789 | |
| Дата и время | xsd:dateTime | Время в виде комбинации календарной даты и времени, как определено в § 5.4 ISO 8601. | 2012-03-19T21:32:52 | |
| | xsd:dateTimeStamp | Время в виде комбинации календарной даты и времени, как определено в § 5.4 ISO 8601 с обязательным указанием часового пояса | 2001-10-26T21:32:52+02:00 | |
| Массивы символов | rdf:PlainLiteral | Строка вида “abc@lng”, где “abc” - производный набор символов или пустая строка, а “lng” - код языка согласно *RFC 4647* или пустая строка. | “foo@en”, “foo@”, | rdf:PlainLiteral: A Datatype for RDF Plain Literals [8] |
| | xsd:string | Строка символов в виде последовательности символов Unicode и ISO/IEC 10646, включая символы пробела, табуляции, возврата каретки и перевода строки | “ Lorem ipsum dolor sit amet ” | XML Schema Part 2: Datatypes Second Edition [7] |
| | xsd:normalizedString | Строки с нормализованными пробелами. Этот тип данных является производным от типа xsd:string | “ Lorem ipsum” | |
| | xsd:token | Строки, размеченные на лексемы. Этот тип данных является производным от типа xsd:normalizedString | “Lorem ipsum dolor sit amet” | |
| | xsd:Name | Лексема, начинающаяся с буквы, символа подчеркивания или двоеточия, за которым следуют символы имени (буквы, цифры и прочие символы). Этот тип данных является производным от типа xsd:token | ”:foo” | |
| | xsd:NCName | Этот тип данных совпадает с типом xsd:Name, за исключением того, что лексемы этого типа не могут начинаться с двоеточия | “foo” | |
| | xsd:NMToken | Тип атрибута NMTOKEN представляет собой набор символов имени (букв, цифр и прочих) в любых сочетаниях. В отличие от Name и NCName, не накладывает ограничений на начальный символ. Этот тип данных является производным от типа xsd:token | “038foo” | |
| Двоичные данные | xsd:base64Binary | Двоичные данные, преобразованные в позиционную систему счисления с основанием 64 (base64) | T1dMIDlgRU wgT250b2xvZ3k= | |
| | xsd:hexBinary | Двоичные данные представленные в виде последовательности двоичных октетов в шестнадцатеричной системе. | 0F7A | |
| Прочие | xsd:anyURI | Универсальный индикатор ресурсов (URI) согласно RFC 2396 и RFC 2732 | http://www.example.com/ | |
| | rdf:XMLLiteral | Строка которая является сбалансированным и автономным XML контентом. Обеспечивает использование XML контента как возможного значения литерала. | <foo>bar</foo> | Resource Description Framework (RDF): Concepts and Abstract Syntax [9] |
| | rdfs:Literal | Произвольный литерал. Литералы могут быть как простыми, так и типизированными. Этот тип является базовым. | - | RDF Vocabulary Description Language 1.0: RDF Schema [10] |

Корневым будем называть такой тип данных, который является наиболее общим типом своей категории. Корневой тип стоит на вершине иерархии родственных ему типов данных и разделяет с ними общее пространство значений.

Для каждого корневого типа создается ещё один контейнер, содержащий пары значений <*Пространство значений, типизированное выражение*>. В нашем случае типизированное выражение это $r \sqsubseteq B$.

После индексации, когда ELK входит в фазу насыщения, для каждого выражения $r_+$ производится поиск выражения $r_-$ которое его поглощает. В Таблице 3 приведена матрица всех возможных таких поглощений, где A представляет ограничение $r_-$, а $B$ представляет $r_+$.

Таблица 3
Матрица поглощений

| B / A | *Empty ValueSp.* | *Entire ValueSp.* | *Binary Value* | *DateTime Value* | *Literal Value* | *Numeric Value* | *DateTime Interval* | *Length Restrict.* | *Numeric Interval* | *Pattern* |
|---|---|---|---|---|---|---|---|---|---|---|
| *Empty ValueSp* | ø | ø | ø | ø | ø | ø | ø | ø | ø | ø |
| *Entire ValueSp* | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ | $B_D \subseteq A_D$ |
| *Binary Value* | ø | ø | $B_D = A_D$<br>$A = B$ | ø | ø | ø | ø | ø | ø | ø |
| *DateTime Value* | ø | ø | ø | $A =^1 B$ | ø | ø | ø | ø | ø | ø |
| *Literal Value* | ø | ø | ø | ø | $A =^2 B$ | ø | ø | ø | ø | ø |
| *Numeric Value* | ø | ø | ø | ø | ø | $A =^3 B$ | ø | ø | ø | ø |
| *DateTime Interval* | ø | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le B$<br>$A_{high} \ge B$ | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le B_{low}$<br>$A_{high} \ge B_{high}$ | ø | ø | ø |
| *Length Restrict.* | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le B_{len}$<br>$A_{high} \ge B_{len}$ | ø | $B_D \subseteq A_D$<br>$A_{low} \le B_{len}$<br>$A_{high} \ge B_{len}$ | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le B_{low}$<br>$A_{high} \ge B_{high}$ | ø | $B_D \subseteq A_D$<br>$B \sqsubseteq^4 A$ |
| *Numeric Interval* | ø | ø | ø | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le^3 B$<br>$A_{high} \ge^3 B$ | ø | ø | $B_D \subseteq A_D$<br>$A_{low} \le^3 B_{low}$<br>$A_{high} \ge^3 B_{high}$ | ø |
| *Pattern* | ø | ø | ø | ø | $B_D \subseteq A_D$<br>$B \overset{4}{\to} \notin ø$ | ø | ø | $B_D \subseteq A_D$<br>$B \sqsubseteq^4 A$ | ø | $B_D \subseteq A_D$<br>$B \sqsubseteq^4 A$ |

[1] Проверка на равенство учитывает положение A и B на шкале времени с учётом указанных часовых поясов или при его отсутствии. Если часовой пояс указан только у одного параметра, используется смещение $\pm$14:00. См. [7]

[2] Литералы считаются равными когда все символы их лексических форм равны между собой, у обоих отсутствует тег языка и/или типа данных или он одинаковый у обоих литералов, если таковы указаны. См. [8]

[3] При сравнении, числа приводятся к общему, наиболее конкретному, типу данных.

[4] Проверка того, что любая интерпретация регулярного выражения (ограничение) $B$ будет удовлетворять ограничению (интерпретации регулярного выражения) $A$. Все ограничения приводятся к виду регулярных выражений. См. [11]

В нашей реализации любое типизированное выражение $r$приводится к внутреннему представлению, отображающее фор-

мируемое им пространство значений. Перечисленные в Таблице 3 ограничения условно делятся на 3 группы:

- Значение: двоичные данные (Binary Value), дата и время (DateTime Value), строка символов (Literal Value) и число (Numeric Value).
- Ограниченное множество: промежуток числовой прямой (Numeric Interval), промежуток времени (DateTime Interval), ограничение длинны (Length Restriction)
- Прочие: пустое множество (Empty Value Space), всё пространство значений определённого типа данных (Entire Value Space) и регулярное выражение (Pattern)

Для корректной проверки совместимости типов данных $B_D \subseteq A_D$ между типизированными ограничениями для всех значений уточняется их фактический тип данных. Так, например, рациональное число “10/2” будет зафиксировано как целое число больше нуля.

## Тестирование

Для проверки работоспособности наших модификаций мы выбрали разрабатываемую нами семантическую информационную систему Грид [12], и модифицировали её таким образом, что бы она соответствовала профилю OWL 2 EL.

Воспользовавшись генератором баз знаний Грид ресурсов [13], мы создали две тестовые онтологии. Первая, полная онтология, содержала 1 087 124 аксиом, 65 классов, 33 объектных свойства, 109 типизированных свойств и 131 637 экземпляра. Второй, усечённый вариант онтологии отличался тем, что содержал 230 670 аксиом и 36 191 экземпляр.

Наша тестовая база знаний содержит подробное описание всех элементов Грид-системы, которая обслуживает выполнение экспериментов на Большом Адронном коллайдере. Онтология представляет собой множество экземпляров, утверждения их свойств и отношений между ними. В качестве “полезной нагрузки” мы добавили несколько классов к онтологии:

UK_Site ≡ Site and hasLocation some (Location and hasName some string[pattern ".*, UK"])

Idle_CE ≡ ComputingElement and hasState some (CEState and hasRunningJobs value 0 and hasWaitingJobs value 0 and hasFreeJobSlots some integer[>0]) and hasState some (CEState and hasStatus value Production)

x64_Cluster ≡ SubCluster and (describedBy some (hasPlatformType value "x86_64"^^string) and (describedBy some (hasRAMSize some integer[>= 4096, <= 8192])))

После классификации экземпляры вышеуказанных классов будут соответствовать искомым Грид-ресурсам.

В таблице 4 приведены результаты тестирования. Для сравнения мы взяли два наиболее популярных логических анализатора, которые поддерживают рассуждения с типизированными свойствами — Pellet

и HermiT. Тестовая конфигурация: Intel Core 2 Duo T9300 @ 2.50GHz, 4 Gb RAM, OpenSUSE 12.1 (Linux 3.1.10), Java 1.7.0_05 (-Xmx3200M -Xms3200M), Pellet 2.2.0, HermiT 1.3.6. Каждый тест проводился 3 раза.

Таблица 4
Результаты тестирования

| Логический анализатор | Время классификации усечённой онтологии, мс | Время классификации полной онтологии, мс |
|---|---|---|
| ELK | 7366 | 54912 |
| ELK[1] | 5598 | 12567 |
| Pellet | 165419[2] | out of mem |
| HermiT | timeout | timeout |

[1] Оптимизированная сборка

[2] Так как Pellet не поддерживает регулярные выражения, определение UK_Site для него было заменено на Site and hasLocation some (Location and hasName value “London, UK”)

К сожалению время необходимое HermiT для классификации вышло за установленный лимит в 1 час, а Pellet не хватило оперативной памяти для успешного логического анализа полной онтологии. Мы также проанализировали производительность ELK и заметили, что возможно значительно ускорить его работу за счёт некоторой потери полноты логического анализа строковых литералов. Так как наша онтология содержала большое количество типизированных свойств с типом xsd:string, мы воспользовались другим, более простым алгоритмом их обработки который не уточняет фактический тип строковых литералов. Конкретно для нашей тестовой онтологии это не повлияло на корректность полученных результатов, но сократило время классификации на 25% и 77% для усечённой и полной онтологии соответственно.

## Заключение

Представленные нами изменения к логическому анализатору ELK открывают дорогу к его широкому использованию в тех случаях, когда необходимо быстро обрабатывать большие массивы знаний представленные в виде OWL онтологий. Представленная здесь поддержка типизированных свойств и конкретных типов данных позволяет применять логический анализ над фактическими и конкретными данными такими, например, как записи баз данных, внешние информационные системы, службы каталогов LDAP, документы RDF и многие другие подобные источники.

Несмотря на некоторые ограничения, наложенные профилем OWL 2 EL, основанные на нём онтологии имеют достаточную практическую экспрессивность и благодаря высокопроизводительным логическим анализаторам подобных ELK, могут быть эффективно использованы в различных прикладных областях.

Представленные изменения в скором времени будут доступны в новой версии ELK. Исходный код ELK с поддержкой типизированных свойств доступен по ссылке:

https://elk-reasoner.googlecode.com/svn/branches/elk-parent-datatypes

**Литература**


1. *K. Dentler, R. Cornet, A. Ten Teije, N. De Keizer.* Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semantic Web, vol. 2(2), pp. 71–87, 2011.
2. *Yevgeny Kazakov, Markus Krötzsch, František Simanuík.* Concurrent Classification of EL Ontologies**.** Proceedings of the 10th International Semantic Web Conference (ISWC-11). LNCS 7032, Springer, 2011.
3. *Yevgeny Kazakov, Markus Krötzsch, František Simanuík.* ELK: A Reasoner for OWL EL Ontologies**.** System Description 2012. http://korrekt.org/papers/Kazakov-Kroetzsch-Simancik_ELK-system-description_TR.pdf
4. *Yevgeny Kazakov, Markus Krötzsch, František Simanuík.* ELK Reasoner: Architecture and Evaluation**.** Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012). CEUR Workshop Proceedings, 2012.
5. *B. Glimm, A. Hogan, M. Krötzsch, A. Polleres.* OWL: Yet to arrive on the Web of Data?. Arxiv preprint, 2012.
6. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/owl2-syntax/.
7. XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2
8. rdf:PlainLiteral: A Datatype for RDF Plain Literals. W3C Recommendation 27 October 2009. http://www.w3.org/TR/rdf-plain-literal
9. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-concepts
10. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema
11. *Anders Moller.* dk.brics.automaton - Finite-State Automata and Regular Expressions for Java. 2010. http://www.brics.dk/automaton
12. *Поспешный А.С., Стиренко С.Г.* GRID-DL – семантический информационный сервис ГРИД // Компьютинг, 2011. - Том 10, Выпуск 3. - С. 285 - 294.
13. *Поспешный А.С., Стиренко С.Г.* Онтология ресурсов для семантического информационного сервиса Грид, Электронное моделирование. 33 (4) (2011).


Отримано 22.05.2012 р.