УДК 510.62:004.82

ДОКАЗАТЕЛЬСТВО ЭФФЕКТИВНОСТИ SIR АЛГОРИТМА АВТО-ПОСТРОЕНИЯ ИНТЕРФЕЙСОВ ВЗАИМОДЕЙСТВИЯ ВЕБ-СЕРВИСОВ

Каширин И.Ю., Курдюков Н.С.

Рязанский государственный радиотехнический университет, Рязань, email: mort.snowflake@gmail.com

Представлено подтверждение эффективности SIR алгоритма авто-построения интерфейсов взаимодействия веб-сервисов. Показаны основы вычислительной сложности для анализа алгоритмов логического вывода из онтологий. Продемонстрирована дескриптивная логика DL-Lite$_R$ и сложность для основных задач логического анализа. Приведено доказательство теоремы о невысокой вычислительной сложности алгоритма. Обнародованы результаты тестирования программной реализации SIR в экспериментальных условиях. Результаты тестирования показали, что выполнение SIR алгоритма требует сравнительно небольшого количества процессорного времени и оперативной памяти. Более того, для онтологии, разработанной специально для системы на базе SIR, алгоритм выполняется быстрее, чем для произвольных онтологий. Результаты статьи были использованы для выбора решения при проектировании SOA-системы перед разработкой технической реализации. SOA-система была апробирована на данных веб-портала образовательных учреждений г. Рязани.

Ключевые слова: онтология, веб-сервис, SOA, дескриптивная логика, вычислительная сложность

EFFICIENCY PROOF OF SIR ALGORITHM OF AUTOMATIC WEB-SERVICE INTERACTION INTERFACES RENDERING

Kashirin I.Y., Kurdyukov N.S.

Ryazan State Radio Engineering University, Ryazan, email: mort.snowflake@gmail.com

The author talks about an efficiency of SIR algorithm of web-service interaction interfaces rendering. The author shows bases of computing complexity for the analysis of reasoners from ontologies. The DL-Lite$_R$ descriptive logic and complexity for the most popular reasoning aims is shown. The theorem of low computing complexity of algorithm is proved. Results of testing of program realization of SIR in experimental conditions are published. Results of testing showed that performance of SIR algorithm demands rather small amount of processor time and random access memory. Moreover, the algorithm is carried out quicker for the ontology developed specially for SIR based systems, than for common ontologies. Results of article were used for a decision choice at SOA system design before development of technical realization. The SOA system was approved on data of the web portal of educational institutions of Ryazan city.

Keywords: ontology, web-service, SOA, description logic, computation complexity

На данный момент существует множество систем для создания веб-приложений с SOA-архитектурой: Microsoft.NET, IBM Lotus Notes, Oracle SOA Suite. В то же время не существует эффективных систем, способных автоматически строить интерфейсы и превращать обычные сайты старого поколения в веб-сервисы. Необходимость таких систем обусловливается проблемами реализации EAI (Enterprise Application Integration) [9]. Решить рассматриваемую задачу можно с помощью применения парадигмы основанного на онтологиях доступа к данным (OBDA)[2]. Этот подход позволяет использовать преимущества онтологий и технологий дескриптивных логик. При этом открывается возможность построения интерфейсов на базе семантической сервис-ориентированной архитектуры (SSOA), чем обеспечивается дальнейшее эффективное развитие технологии Semantic web. Для решения поставленной задачи нами разработан алгоритм SIR. В то же время разработка программы логического анализа на основе этого алгоритма для онтологий в среде Semantic Web является ресурсоемким процессом [6]. Во избежание плохих результатов после создания программы необходимо использовать явный критерий для анализа применимости алгоритма SIR до этапа разработки программы. Таким критерием может послужить вычислительная сложность алгоритмов логического вывода в онтологиях.

Цель работы: выявить эффективность алгоритма SIR посредством доказательства его невысокой вычислительной сложности и тестирование программной реализации в экспериментальных условиях.

Теоретические исследования

Алгоритм SIR является алгоритмом логического анализа онтологии. Основным критерием эффективности алгоритма логического анализа помимо выполнения задачи является невысокий показатель вычислительной сложности алгоритма [6]. Вычислительная сложность алгоритма определяет зависимость необходимого объёма памяти и времени для выполнения этого алгоритма от размера входных данных. [5] Основ-

ным понятием вычислительной сложности алгоритмов является класс вычислительной сложности задачи, которую алгоритм должен решать [5]. Далее приведены классы сложности, распространенные в области разработки алгоритмов для дескриптивных логик [3]:

$$\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{ExpTime} \subseteq \text{NExpTime}$$

Класс LogSpace содержит задачи, для решения которых детерминированной машине Тьюринга потребуется объем памяти, находящийся в логарифмической зависимости от входных значений.

Приставка N в названиях классов NlogSpace, NP и NexpTime означает, что эти классы содержат задачи, которые могут разрешиться за количество ресурсов, соответствующее их классу сложности с названием без приставки на недетерминированной машине Тьюринга. Так как недетерминированная машина Тьюринга – всего лишь абстрактная модель, то алгоритм этого класса вряд ли сможет разрешиться на реальном компьютере, соответствующем детерминированной машине, за то же время.

Класс P вмещает все те проблемы, решение которых детерминированной машиной Тьюринга полиномиально зависит от размера входа.

ExpTime – это класс, содержащий задачи, которые детерминированная машина Тьюринга решает за количество времени, находящееся в экспоненциальной зависимости от входных значений.

Задачи класса P считаются интуитивно простыми задачами. Задачи класса ExpTime считаются сложными. Так как не доказано, что NP = P [1], то задачи класса NP считаются сложными.

Для задач логического вывода различают три способа вычисления сложности, перечисленные далее.

1. Сложность относительно данных: сложность оценивается как функция от размера *Abox*.

2. Сложность относительно терминологии: сложность оценивается как функция от размера *Tbox*.

3. Комбинированная сложность: сложность оценивается как функция от размера *Tbox* и *Abox*.

Дискриптивная логика DL-Lite$_R$

Для определения сложности алгоритма необходимо учесть вычислительную сложность дескриптивной логики, для которой разработан алгоритм.

Алгоритм SIR построен для работы с онтологиями, написанными на языке веб-онтологий OWL 2 QL. Этот профиль языка OWL 2 реализует логику DL-Lite$_R$ семейства логик DL-lite[1].

Синтаксис логики DL-Lite$_R$ поддерживает:

– положительные включения $C \sqsubseteq D$,

– отрицательные включения $C \sqsubseteq \neg D$,

где C и D – базовые концепты, представляющие собой либо атомарный концепт, либо роль R, ограниченную квантором существования и принадлежащую верхнему концепту ($\exists R.\top$). Кроме того, поддерживается инверсия ролей (R^-). Несмотря на скудную выразительную силу приведенного синтаксиса, этого достаточно, чтобы описать основные элементы ER-модели.

Алгоритмы логического анализа служат для определения совместности базы и для переписывания запросов. Так как алгоритм работает с логикой DL-Lite$_R$ и с алгоритмами переписывания запросов, для определения его эффективности необходимо знать вычислительную сложность основных задач логического вывода для DL-Lite$_R$.

Данные о вычислительной сложности DL-Lite$_R$ [7] представлены в табл. 1.

Таблица 1

	Сложность относительно терминологии	Сложность относительно данных	Комбинированная сложность
Разрешимость базы знаний в DL-liteR	-	LogSpace	PTime
Ответ на конъюнктивные запросы в DL-LiteR	Ptime	LogSpace	NP-полная

Вычислительная сложность алгоритма SIR

Входными параметрами алгоритма запросов к онтологии являются триплет запросов к базе знаний [7] $< C,Q,S >$.И Tbox онтологии. Выходными данными алгоритма является объединение триплетов запросов к базам данных. Далее запросы автоматически оборачиваются в стандартный код интерфейса веб-сервисов. Полученные интерфейсы размещаются на веб-сервисах исходя из содержания выходных данных алгоритма SIR.

Запрос адреса сервера и запросы адреса клиента могут содержать одинаковые элементы условий между собой. Такие параметры называются параметрами с зависимыми условиями. Для обработки параметров

с зависимыми элементами используется функция DTRew.

Теорема: Пусть T – это DL-lite$_R$ TBox и C, Q, S конъюнктивные запросы к T. Время выполнения алгоритма SIR(C, Q, S, T) является полиномиально зависимым от размера T.

Доказательство:

Для доказательства теоремы выявим существование доказательства лемм, приведенных ниже.

Лемма 1. Вычислительная сложность алгоритма SIR (C, Q, S, T) для параметров без зависимых элементов не превысит полиномиальную сложность от размера Tbox.

Доказательство. Основными действиями в алгоритме SIR при входных параметрах без зависимых условий является переписывание трех запросов. С помощью методики анализа основных алгоритмических конструкций [4] получим функцию вычисления трудоемкости алгоритма:

$$f_1(C,Q,S,T) = r(C,T) + r(Q,T) + r(S,T), \quad (1)$$

где r – это функция переписывания запросов. Переписывание запросов (ответ на конъюнктивные запросы), согласно табл. 1, относится к классу сложности Ptime. Операция сложения относится к задачам с линейной зависимостью, входящих в класс P. Переписывание запросов в DL-Lite полиноминально зависит от размера T.

При вводе параметров с зависимыми переменными структура алгоритма будет состоять из переписывания запросов и функции обработки связных запросов. Ключевым значением в алгоритме при параметрах с зависимыми элементами обладает функция обработки связных запросов DTRew.

Лемма 2. Вычислительная сложность функции DTRew(C, uC, S, T) является полиномиально зависимой от размера Tbox.

Доказательство. Ниже представлен анализ алгоритмических конструкций [4] худшего случая для функции DTRew.

Функция DTRew (C, uC, S, T)

Входные параметры: конъюнктивные запросы: C, S; объединение переписанных запросов uC; TBox T;

Выходные данные: Ассоциативный массив C’SDict;

```
foreach C’ in C do    (3 операции на 1 проход цикла)
        foreach S’ in S do    (3 операции на 1 проход цикла)
                foreach varqC in C’ do    (3 операции на 1 проход цикла)
                        foreach varqS in S’ do    (3 операции на 1 проход цикла)
                                if (varqC = varqS) and (not(C’ in DTSet)) (4 операции на каждый
                                        элемент коллекции)
                                then DTSet : = DTSet + C’;  (2 операции)
if DTSet < > 0    (1 операция)
then
        foreach uC’ in uC do    (3 операции на 1 проход цикла)
                C’S : = S    (1 операция)
                foreach C’ in uC’ do (3 операции на 1 проход цикла)
                        foreach varqC in C’ do    (3 операции на 1 проход цикла)
                                if varqC in DTSet    (2 операции на каждый элемент коллекции)
        then C’S : = C’S ^ C’;    (2 операции)
 C’SDict[uC’] : = C’S;    (2 операции)
return C’SDict;    (1 операция)
```

Худшим случаем для функции DTRew является тот случай, когда во все проверки условия «if» будут истинны.

Конструкция «следование» в процедурных языках программирования занимает 1 операцию [4].

Функция вычисления трудоемкости DTRew:

$$f_2\wedge(C,S,vqC,vqS,DTSet,uC,uC') = C\big(3+1+S(3+1+vqS(3+1+4DTSet+1+2))\big)+$$
$$+1+1+uC(3+1+1+1+uC'(3+1+C'(3+1+2DTSet+1+2))+1+2)+1+1= \quad (2)$$
$$= C*S(7vqS+4vqS*DTSet+4)+uC*uC'(2DtSet*7C'+4)+9uC+4C+4;$$

$$f_2\wedge(C,S,vqC,vqS,DTSet,uC,uC')\subseteq P, \quad (3)$$

где C,S – количество элементов запросов; vqC, vqS – количество элементов переменных запросов (varqC, varqS); uC – количество переписанных запросов к клиенту; uC' – количество элементов переписанного запроса к клиенту; DTSet – количество элементов коллекции зависимых переменных.

Линейная зависимость входит в класс P. Следовательно, вычислительная сложность функции DTRew не превышает в худшем случае класс Ptime от входных параметров функции.

Лемма 3. Вычислительная сложность алгоритма SIR(C, Q, S, T) для параметров с зависимыми элементами является полиномиально зависимой от размера Tbox.

Доказательство. Функция вычисления трудоемкости алгоритма SIR(C, Q, S, T) будет рассчитана следующим образом:

$$f\wedge = f_1 + r(f_2\wedge) + x, \quad (4)$$

где x – количество переписанных запросов в $r(f_2\wedge)$. Согласно лемме 1, вычислительная сложность f_1 относится к классу Ptime. Согласно лемме 2, f_2 относится к классу Ptime. x, $r(f_2\wedge)$ имеют линейную зависимость. Следовательно:

$$f\wedge \subseteq Ptime. \quad (5)$$

На основании леммы 1, леммы 2 и леммы 3 можно сказать, что время выполнения алгоритма SIR(C, Q, S, T) является полиномиально зависимым от размера T. Ч.Т.Д.

Экспериментальные исследования

Для подтверждения правильности теоремы на практике создан и опробован на тестовых данных программный код, реализующий алгоритм.

Программный код алгоритма написан на языке программирования Java. Тестировался на ОС Windows 8, 3 гб ОЗУ, AMD Phenom(tm) II N850 Triple-Core Processor 2,20 Гц. Для тестирования использовалась онтология SAMAO. Эта онтология написана с помощью редактора онтологий Protégé на языке веб онтологий OWL 2 QL. Данные из Abox в этой реализации хранятся в СУБД MySQL. Для основных задач логического анализа онтологий использовался Hermit. Для описания интерфейса веб-сервисов использованы стандарты SOAP и WSDL. Для сравнения использовались 2 алгоритма переписывания запросов: PerfectRef [7] и Requiem [8].

Тестовые данные, на которых анализировался алгоритм, состоят из онтологий, написанных или адаптированных под язык OWL 2 QL для тестирования алгоритмов логического анализа. Далее приведено описание тестовых онотологий[8].

VICODI – онтология, содержащая информацию о Европейской истории, разработанной Европейским проектом Евросоюза. VICODI обозначена в табл. 2 буквой V.

StockExchange – онтология содержит информацию о финансовом институте Евросоюза. Разработана для архитектуры OBDA. StockExchange обозначена в табл. 2 буквой S.

LUMB6 – DLlite версия онтологии для тестирования, разработанной Легигским университетом для тестирования алгоритмов логического анализа. Онтология содержит данные об организационной структуре университета. LUMB6 – обозначена в табл. 2 буквой L.

SAMAO – Онтология государственных и муниципальных учреждений, разработанная специально для системы авто построения интерфейсов взаимодействия веб-сервисов. Набор аксиом(Tbox) онтологии SAMAO содержит данные о ведомствах и их территориальном расположении. Набор утверждений(Abox) онтологии SAMAO содержит данные об адресах веб-сервисов и данные, предоставляемые веб-сервисами. SAMAO – обозначена буквами SM в табл. 2. Ниже представлен отрывок из онтологии SAMAO.

TypeOfEstablishment ⊑ Think
Education ⊑ TypeOfEstablishment
GOUVPO ⊑ Education
GeneralEducationEstablishment ⊑ Education
PreschoolEducationEstablishment ⊑ Education
HighSchool ⊑ GeneralEducationEstablishment
…

Входные параметры алгоритма SIR(C,Q,S,T) для тестовых онтологий выбирались из тестов запросов к онтологиям[8]. SIR(C = Q2, Q = Q4, S = Q3, T = Tbox онтологии).

Входные параметры алгоритма SIR(C,Q,S,T) для онтологии SAMAO приведены ниже.

Параметры без зависимых условий.
C(x1, x2) ← EducationPortal(x1), RyazanCity(x2)
Q(x3) ← Phone(x3)
S(x4, x5) ← Education(x4), RyazanArea(x5)
Параметры с зависимыми условиями.
C(x1, x2, x3) ← EducationPortal(x1), CentralRegion(x2), City(x3)
Q(x4) ← Phone(x4)
S(x5, x3) ← Education(x5), City(x3)

Время переписывания запросов для параметров без зависимых условий рассчитывалось по следующей формуле:

$$t(x) = t(q) + t(s) + t(c), \qquad (6)$$

где $t(q)$, $t(s)$, $t(c)$ – это время срабатывания алгоритма переписывания соответствующих запросов.

Время переписывания запросов для параметров с зависимыми условиями рассчитывалось по следующей формуле:

$$t(x) = t(q) + t(s) + t(d) + \sum_{k=0}^{n} t(u_k), \qquad (7)$$

где $t(q)$, $t(s)$ – это время срабатывания алгоритма переписывания соответствующих запросов. $t(u_k)$ – время срабатывания алгоритма переписывания для массива, порожденного функцией DTRew, $t(d)$ – время работы функции DTRew.

При работе алгоритма порождаются новые запросы к базам данных. Порожденные запросы сохраняются в текстовом формате. Для вычисления занимаемой памяти результатов работы алгоритма следует подсчитать количество порожденных запросов. Количество порожденных запросов было подсчитано по формуле, идентичной формуле расчета времени работы алгоритма, с той лишь разницей, что вместо функции подсчета времени $t(x)$ использовалась функция подсчета количества порожденных запросов.

Результаты тестирования приведены в табл. 2.

Таблица 2

	Зависимые условия	Количество порожденных запросов			Время, мс		
		DTRew	PerfectRef	Requiem	DTRew	CGRRL	Requiem
SM	Без ЗУ	–	75	72	–	19	16
	С ЗУ	3	147	121	1	33	26
V	Без ЗУ	–	115	140	–	47	53
	С ЗУ	10	830	1015	1	139	152
S	Без ЗУ	–	2009	1753	–	1450	1200
	С ЗУ	0	2009	1753	1	1450	1200
L	Без ЗУ		2781	1967	–	1011	603
	С ЗУ	285	792585	560595	8	130324	75927

Заключение

Результаты тестирования показали, что выполнение SIR алгоритма требует сравнительно небольшого количество процессорного времени и оперативной памяти. Эти выводы сходятся с тем, что вычислительная сложность алгоритма SIR входит в класс P, являющийся классом быстрых алгоритмов. Более того, для онтологии, разработанной специально для системы на базе SIR, алгоритм выполняется быстрее, чем для произвольных онтологий (см. SM в табл. 2).

Результаты статьи были использованы для выбора решения при проектировании SOA-системы перед разработкой технической реализации. SOA-система была апробирована на данных веб-портала образовательных учреждений г. Рязани.

Список литературы

1. Курдюков Н.С. Описание ER-модели доступа к услугам веб-сервисов средствами дескриптивных логик // Отраслевые аспекты технических наук. – М.: ООО «ИНГН», 2012. – № 12 (24). – С.71–72

2. Курдюков Н.С. Основанный на онтологиях доступ к данным формата Semantic web. С. 67–68 // Общество, современная наука и образование: проблемы и перспективы: сборник научных трудов по материалам: Международной научно-практической конференции 30 ноября 2012г.ч 5; М-во обр. и науки РФ. – Тамбов Изд-во «Бизнес-Наука-Общество», 2012. – 163 с.

3. Курдюков Н.С. Вычислительная сложность как критерий выбора дескриптивной логики для описания онтологий в среде Semantic web. С. 57–59 // Achievement of high school: материали за 8-а международна научна практична конференция. – 2012. – Том 10. Икономики. Съвременни технологии на информации. – София. «Бял ГРАД-БГ» ООД. – 96 с.

4. Макконнелл Дж. Основы современных алгоритмов. – 2-е изд., доп. – М.: Техносфера, 2004. – 368 с.

5. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений = Introduction to Automata Theory, Languages, and Computation. – М.: «Вильямс», 2002. – С. 528.

6. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P.F. The description logic handbook: Theory, implementation, and applications. Cambridge University Press, 2nd edition, 2010.

7. Calvanese D., De Giacomo G., Lembo D., Lenzerini M., Rosati R. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. Journal of Automated Reasoning. 09/2007; 39(3):385–429.

8. Pérez-Urbina H., Horrocks I., Motik. B. Practical Considerations for Query Answering in OWL 2. In Proc. of the OWL: Experiences and Directions Workshop (OWLED2009). Chantilly, VA, USA. October, 2009.

9. Trotta G. Enterprise Application Integration Using Extensible Web Services. (2003-12-15). «Dancing Around EAI 'Bear Traps'». Retrieved 2006-06-27.

References

1. Kurdyukov N.S. Description ER-model access to Web services by descriptive logics. pp. 71–72, Scientific and practical magazine «Industry aspects of Science». Moscow: OOO «INGN», 2012, no. 12 (24).

2. Kurdyukov N.S. Ontology based access to the data format of Semantic web. pp. 67–68. Society, modern science and education: problems and prospects: a collection of research papers based on the International Scientific Conference on November 30 2012. part 5, Tambov, Publishing House of the «Business-Science-Society», 2012. p. 163

3. Kurdyukov N.S. Ryazan State Radio Engineering University, Russia. Computational complexity as a criterion for selecting a descriptive logic for describing ontologies among Semantic web. pp. 57–59, the material for the 8-and by international scientific practical conference, «Achievement of high school», 2012. Volume 10. Sofia. «Byal GRAD-BG» Ltd. 96 p.

4. McConnell J. Foundations of modern algorithms. 2nd revised edition. Moscow: Technosphere, 2004. 368p.

5. Hopcroft J., Motwani R., Ullman J. Introduction to Automata Theory, Languages, and Computation. M.: «Williams», 2002. pp. 528.

6. Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P.F. The description logic handbook: Theory, implementation, and applications. Cambridge University Press, 2nd edition, 2010.

7. Calvanese D., De Giacomo G., Lembo D., Lenzerini M., Rosati R. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. Journal of Automated Reasoning. 09/2007; 39(3):385–429.

8. Pérez−Urbina H., Horrocks I., Motik. B. Practical Considerations for Query Answering in OWL 2. In Proc. of the OWL: Experiences and Directions Workshop (OWLED2009). Chantilly, VA, USA. October, 2009.

9. Trotta G. Enterprise Application Integration Using Extensible Web Services. (2003-12-15). «Dancing Around EAI 'Bear Traps'». Retrieved 2006-06-27.

Рецензенты:

Овечкин Г.В., д.т.н., профессор, заведующий кафедрой ВПМ Рязанского государственного радиотехнического университета, г. Рязань;

Пылькин А.Н., д.т.н., профессор кафедры ВПМ Рязанского государственного радиотехнического университета, г. Рязань.

Работа поступила в редакцию 11.04.2013.