На правах рукописи

Курдюков Николай Станиславович

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СЕРВИС-ОРИЕНТИРОВАННЫХ СИСТЕМ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ЯЗЫКОВ ДЕСКРИПТИВНОЙ ЛОГИКИ

Специальность 05.13.11 –
«Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Рязань 2014

Работа выполнена на кафедре «Вычислительная и прикладная математика» ФГБОУ ВПО «Рязанский государственный радиотехнический университет» (ФГБОУ ВПО «РГРТУ»).

Научный руководитель: Каширин Игорь Юрьевич, доктор технических наук, профессор, профессор каф. ВПМ ФГБОУ ВПО «РГРТУ»

Официальные оппоненты: Романчук Виталий Александрович, кандидат технических наук, доцент, ФГБОУ ВПО «Рязанский государственный университет имени С.А. Есенина», заместитель декана по науке физико-математического факультета

Сальников Игорь Иванович, доктор технических наук, профессор, ФГБОУ ВПО «Пензенский государственный технологический университет», заведующий кафедрой «Вычислительные машины и системы»

Ведущая организация:

Федеральное государственное бюджетное учреждение науки Институт программных систем им. А.К. Айламазяна Российской академии наук

Защита состоится "25" марта 2015 г. в 12.00 часов на заседании диссертационного совета Д212.211.01 в ФГБОУ ВПО «Рязанский государственный радиотехнический университет» по адресу: 390005, г. Рязань, ул. Гагарина, д. 59/1.

С диссертацией можно ознакомиться в библиотеке ФГБОУ ВПО «Рязанский государственный радиотехнический университет».

Автореферат разослан "__" _____ 201_ г.

Ученый секретарь
диссертационного совета
кандидат технических наук, доцент В. Н. Пржегорлинский

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы

Рост популярности за последние годы в сети Internet таких технологий, как облачные вычисления (cloud computing), семантические сети и Semantic Web, демонстрирует актуальность задачи организации взаимодействия Internet-ресурсов между собой. Обеспечить такое взаимодействие позволяет сервис-ориентированная архитектура (SOA) разработки программного обеспечения. SOA-системы легко масштабируются, а их сервисные структуры облегчают повторное использование компонентов. SOA-архитектура обеспечивает создание прикладных программных систем, построенных посредством распределенных взаимодействующих сервисов.

На данный момент существует множество систем для создания web-приложений с SOA архитектурой: IBM Lotus Notes, Microsoft .NET, Oracle SOA Suite. В то же время не существует эффективных систем, способных автоматически строить интерфейсы и превращать популярные сайты прежнего поколения в элементы композитных служб или сервисных кластеров. Необходимость таких систем обусловливается проблемами реализации EAI (Enterprise Application Integration – интеграции корпоративных приложений).

EAI является интеграционной архитектурой, состоящей из набора технологий и методов для интеграции систем и приложений в масштабах корпоративных архитектур. Приложения управления цепочками поставок, документооборота, управления взаимоотношениями с клиентами, бизнес-аналитики и другие, как правило, не могут взаимодействовать друг с другом в целях обмена данными или правилами бизнес-процессов без создания дополнительных программных решений. Следовательно, появляются проблемы с автоматизацией процессов и избыточными данными, хранящимися в нескольких местах. Интеграция корпоративных приложений – средство, обеспечивающее процесс связывания таких приложений в пределах одной организации вместе для упрощения и автоматизации бизнес-процессов. Процесс связывания должен протекать, не внося радикальных изменений в существующие приложения или структуры данных. Вследствие этих факторов построение SOA-системы с композитной структурой обычно сводится к методам EAI.

В 2003 году стало известно, что 70 % всех EAI-проектов потерпело неудачу. Председатель европейского консорциума по интеграции Стив Краггс отметил проблемы реализации EAI-систем. Главная проблема заключается в том, что элементы EAI-системы постоянно изменяются. Требуются ресурсы на постоянное переделывание EAI-системы и ее расширение.

Помимо проблем реализации EAI-систем, поддержке SOA-систем, активно работающих с сетью Internet, мешает фактор постоянного развития всемирной

паутины. При этом постоянное изменение архитектуры и разработка заново уже существующих интерфейсов требуют непрерывной работы специалистов.

Построение эффективной SOA-системы, лишенной или частично лишенной проблем реализации EAI-систем, целесообразно решить с помощью применения парадигмы основанного на онтологиях доступа к данным (OBDA – Ontology-Based Data Access), продвигаемых благодаря ученому из Италии Диего Кальванезе. OBDA – доступ к данным через высокоуровневый интерфейс онтологии. Центральным процессом основанного на онтологиях доступа к данным является представление базы данных на языке дескриптивной логики (ДЛ). Такой подход позволяет использовать преимущества онтологий и ДЛ. Развитие технологии языков дескриптивной логики связано с такими учеными, как Ф.Баадер, Д.Кальванезе, И.Ю.Каширин, Я.Хоррокс, К.Луц, Р.Контчаков. Вследствие применения OBDA-технологий упростилась работа по изменению структуры SOA-системы по сравнению с ведущими SOA-системами, указанными ранее. Стало достаточно для изменения структуры SOA-системы отредактировать онтологию системы, что не составляет труда для человека без специального образования, с помощью редактора онтологий. При этом открывается возможность построения интерфейсов на базе семантической сервис-ориентированной архитектуры (SSOA) продвигаемой создателем всемирной паутины Тимоти Бернерс-Ли, чем обеспечивается дальнейшее эффективное развитие технологии Semantic Web.

Цель работы

Целью диссертационной работы является создание формализма описания взаимодействия сервисов и алгоритмов построения интерфейсов web-сервисов для реализации эффективной SOA-системы с применением парадигмы основанного на онтологиях доступа к данным.

Реализация OBDA-системы позволяет извлекать данные из баз данных на основе логического анализа решателя над базой знаний. При этом эффективность доступа к данным достигается за счет отображения запросов в среду реляционной СУБД, располагающей эффективными средствами обработки запросов и другими преимуществами. TBox (набор аксиом) должен использоваться для обеспечения необходимой семантической информации на более высоком концептуальном уровне.

Для реализации SOA-системы необходимо создать формализм для описания процессов системы с помощью онтологии и эффективные алгоритмы или усовершенствовать существующие OBDA-алгоритмы под работу с SOA-системами.

Основные задачи

Для достижения цели диссертации необходимо решение следующих задач.

1. Провести анализ основных видов структур взаимодействия SOA-сервисов. Привести краткий обзор экспериментальных реализаций систем OBDA, работающих с семейством языков дескриптивной логики DL-Lite. Подтвердить практическое применение дескриптивной логики и языков web-онтологий в целях предоставления онтологического доступа к данным.

2. Создать формализм для описания и анализа концептуальной модели SOA-системы, ориентированный на онтологический подход и позволяющий адекватно описывать взаимодействие web-сервисов.

3. Спроектировать алгоритмы автоматизированного построения интерфейсов web-сервисов для SOA-архитектуры, дающие возможность автоматизированно строить интерфейсы web-сервисов на основе онтологии.

4. Разработать программную реализацию SOA-системы, использующую полученные алгоритмы автоматизированного построения интерфейсов взаимодействия web-сервисов. Для перехода системы на SSOA-архитектуру структура онтологии, необходимой для работы системы, должна иметь OWL-разметку стандарта Semantic Web.

5. Обосновать теоретически сложность и завершаемость принятых решений

Объект исследования

Объектом исследования настоящей диссертации являются SOA-системы, а также их оптимизация с помощью парадигмы основанного на онтологиях доступа к данным.

Методы исследования

Для решения поставленных задач использовались методы общей алгебры, теории алгоритмов, объектно-ориентированного проектирования, теории унификации.

Научная новизна

1. Получен математический формализм, описывающий функционирование сервис-ориентированной системы. На основе предложенного формализма разработаны алгоритмы построения web-сервисов, использующие языки дескриптивной логики и работающие быстрее аналогов на 24 — 41%.

2. Получен метод оптимизации алгоритмов трансформации запросов под SIR-систему, отличающийся от сторонних алгоритмов трансформации запросов более быстрой работой за счет игнорирования несущественных элементов входных данных. Разница в скорости работы между полученным алгоритмом трансформации запросов и аналогами зависит от количества несущественных элементов в онтологии.

3. Доказаны теоремы о завершаемости и вычислительной сложности SIR-алгоритмов для онтологических SOA-систем.

4. Доказана корректность и сложностные оценки построенных алгоритмов

На защиту выносятся

1. Формализм для описания и анализа концептуальной модели SOA-системы, ориентированный на онтологический подход и позволяющий описывать взаимодействие web-сервисов.

2. Новые алгоритмы автоматизированного построения интерфейсов web-сервисов для SOA-архитектуры, дающие возможность автоматизированно строить интерфейсы web-сервисов.

3. Программная реализация SOA-системы, предназначенная для использования в основе систем автоматизированного построения интерфейсов взаимодействия web-сервисов.

4. Структура онтологии, необходимой для работы SOA-системы, имеющая OWL-разметку стандарта Semantic Web.

5. Сложностные оценки алгоритмов

Теоретическая значимость

1. Получена точная семантика SOA-системы, дающий возможность описывать SOA-системы в языках дескриптивной логики. Показана эффективность математического формализма для задач логического анализа прикладных онтологий и описания сервис-ориентированных систем.

2. Показана необходимость применения алгоритмов автоматического построения web-интерфейсов.

3. Доказаны теоремы о низкой вычислительной сложности, разрешимости и конечности SIR-алгоритмов.

Практическая значимость

Получена система SIRSystem, показывающая состоятельность систем автоматизированного построения интерфейсов взаимодействия web-сервисов.

Структура онтологии, необходимой для работы системы, имеет OWL-разметку стандарта Semantic Web. Этот фактор открывает возможность перехода системы на базу семантической сервис-ориентированной архитектуры (SSOA), ориентированной на Semantic Web сервисы. Результаты тестирования подтверждают корректность полученной реализации SOA-системы. Использование дескриптивных логик гарантирует динамичность и простоту масштабирования системы, разработанной на базе представленных в диссертации алгоритмов. Небольшое количество времени, требуемое для работы SIR алгоритма, и приемлемое количество порожденных запросов доказывают эффективность алгоритма системы автоматизированного построения интерфейсов взаимодействия web-сервисов.

Реализация и внедрение результатов диссертационной работы

Научные и практические результаты диссертации Н.С. Курдюкова нашли отражение в следующем программном обеспечении, работающем в рамках образовательного портала для управления образования и молодежной политики г. Рязани:

– подсистема, реализующая SIR алгоритм автоматического построения интерфейсов веб-сервисов;

– автоматизированная программная система «Система SIRSystem v. 1.0»;

– онтология «SAMAO».

Перечисленные системы внедрены в промышленную эксплуатацию в числе подсистем образовательного портала администрации города Рязань. Эксплуатация этих систем показала их работоспособность, высокие характеристики надежности и эффективности.

Результаты исследований, полученные в диссертации Курдюкова Н.С., внедрены в учебный процесс по направлениям 231000 – «Программная инженерия» и 230700 «Прикладная информатика». Разработанные в диссертационной работе алгоритмы автоматического построения интерфейсов web-сервисов посредством логического анализа онтологий, использованы в дисциплинах «Проектирование систем искусственного интеллекта», «Информационные технологии», «Мировые информационные технологии», что подтверждается соответствующими актами.

Кроме того, была проведена государственная регистрация программы SIRGen генерации и отправки SOAP-отчетов для системы SIRSystem v.1.0, что подтверждается свидетельством о государственной регистрации программы для ЭВМ № 2014660476, Роспатент, 2014.

Апробация работы

Основные положения диссертационной работы отражены в докладах:

− Международной научно-практической конференции «Общество, современная наука и образование: проблемы и перспективы» (Тамбов, 2012);

− 8-й Международной научно-практической конференции «Achievement of high school» (Болгария, София, 2012);

− VIII Международной научно-практической конференции «Wyksztalcenie I nauka bez granic» (Польша, Пшемысль, 2012);

− XVIII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов «Новые информационные технологии в научных исследованиях» (Рязань, 2013);

− X Международной научно-практической конференции «Strategiczne pytania światowej nauki - 2014» (Польша, Пшемысль, 2014).

Публикации. Основные результаты диссертации опубликованы в 16 работах. В том числе: 3 статьи в изданиях, рекомендуемых ВАК, 4 статьи в сборниках научных трудов, 9 тезисов докладов на международных и всероссийских конференциях.

Структура и объем диссертации. Диссертация состоит из введения (8 с.), 4-х глав (110 с. – 30 рисунков и 12 таблиц). Библиографический список включает 101 наименование (10 с.).

СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность темы, сформулированы цель диссертации, научная новизна, приведены сведения о практическом использовании полученных научных результатов и представлены основные положения, выносимые на защиту.

В первой главе приведен анализ современных технологий для проектирования в рамках парадигмы SOA. Приведена классификация аналогичных технологий и подходов. Рассмотрены языки описания взаимодействия элементов SOA-систем. Выявлены недостатки существующих подходов к проектированию интерфейсов web-сервисов и EAI-систем. Проанализированы основные виды структур взаимодействия SOA-сервисов.

Преимущество SOA-технологий заключается в многократном использовании программного кода для создания сложносоставных распределённых программных систем. Сервис-ориентированная архитектура обеспечивает кроссплатформенность и независимость от средств разработки,

позволяя легко управлять и интегрировать создаваемые программные системы. Например, программные реализации компонентов SOA-системы, написанные на языке программирования Java, могут без труда быть вызваны компонентами, написанными на языке программирования C# в рамках одной или нескольких SOA-систем. Основным преимуществом SOA-архитектуры является способность к масштабируемости - управлению ростом корпоративных систем, способность предоставления и использования услуг в масштабах сети Internet, способность к сокращению затрат на организацию взаимодействия систем. Ниже приведены примеры проекта, для которого подходит SOA-архитектура. Крупной компании, которая приобретает другие небольшие компании, необходимо определить, как интегрировать приобретенные ИТ-инфраструктуры в свой проект. Для этого можно построить интерфейсы между этими инфраструктурами, тогда такие инфраструктуры целесообразно представлять в виде сервисов. Или существует большое количество web-ресурсов, которые должны взаимодействовать между собой в автоматическом или автоматизированном режиме. Тогда для каждого из web-ресурсов целесообразно построить web-сервис и рассматривать взаимодействие сервисов как сервис-ориентированную систему(SOA-систему). Благодаря способности к масштабированию и интеграции SOA-архитектура позволяет проектам адаптироваться к нуждам конкретной предметной области или процессам, проходящим в рамках этой предметной области. Инфраструктура SOA способствует также более гибкой и оперативной работе системы, чем цельная система, построенная на огромном числе парных интерфейсов. Таким образом, SOA-архитектура обеспечивает прочную основу для систем, отвечающих условиям бизнес-гибкости и адаптивности.

Центральным понятием SOA-архитектуры является сервис (service – услуга). Существительное «услуга» определяется в словарях как «выполнение работ (функций) друг для друга». Однако надо помнить, что термин сочетает в себе следующие взаимосвязанные идеи:

- возможность выполнять работу для клиентов;
- четкую формализацию работ, предоставляемых клиентам;
- предложение выполнить работу для клиента.

Описание сервисов также передает информацию о задачах службы и условия для использования сервиса. Ключевым компонентом сервисов являются строго детерминированные интерфейсы. Интерфейсы могут быть вызваны стандартными способами, даже если данные о вызывающем их приложении не известны сервису. В то же время объектам, взаимодействующим с сервисами, не известны данные о механизмах выполнения службами их целевой задачи. Эти эффекты достигаются за счет инкапсуляции деталей реализации от остальных частей системы.

Основанный на онтологиях доступ к данным как эффективный интерфейс для представления данных web-сервисов в формате Semantic Web является перспективной областью научно-практического исследования. Реализация данного интерфейса должна обеспечивать эффективное выполнение запросов, не уступающее по скорости запросам к современным системам управления базами данных (СУБД) при более абстрактном представлении баз данных. Это даст возможность использовать неопределенные ранее отношения и позволит обнаруживать противоречивость данных.

Для того чтобы придать формальную семантику свойствам стандартных концептуальных моделей, таких как ER и UML-диаграммы, и обеспечить эффективный доступ к большому объему данных, хранящихся в базах данных, было введено семейство дескриптивных логик DL-Lite. Из этих фактов следует, что DL-Lite подходит для задач основанного на онтологиях доступа к данным. Вычислительная сложность всех задач логического анализа в основных логиках семейства DL-Lite не превосходит полиномиальной за счет низкой выразительной силы, достаточной для описания ER-моделей. Для ответа на запросы сложность не превышает LogSpace за счет алгоритмов переписывания запросов. Под переписыванием запросов подразумевается преобразование запроса в объединение конъюнктивных запросов на отношениях реляционной базы данных.

Рассмотрены вопросы экспериментальной реализации систем OBDA, работающих с семейством дескриптивных логик DL-Lite. Наличие таких реализаций подтверждает практическое применение дескриптивных логик и языков web-онтологий в целях предоставления онтологического доступа к данным. Поставлена задача диссертации.

Вторая глава посвящена созданию формализма для описания и анализа концептуальной модели SOA-системы, ориентированного на онтологический подход. В качестве основной теоритической концепции используется сервис-ориентированная архитектура (SOA), являющаяся парадигмой разработки программного обеспечения посредством использования распределённых слабо связанных компонентов. При разработке системы для сети Internet основным объектом концепции SOA выступает web-сервис — идентифицируемая web-адресом программная система, оснащённая стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам. Основой работы web-сервисов может послужить предоставление определенных данных БД web-сервиса. В том случае если сервис предлагает услуги по обработке данных, он может запрашивать данные у другого сервиса и обрабатывать эти данные.

Для построения интерфейса i_{SOA} между двумя web-сервисами необходимо иметь следующую информацию в виде тройки значений:

$$i_{SOA} = <C_S, Q_S, S_S>,$$

где C_S – переменная, содержащая адрес web-сервиса, запрашивающего информацию (адрес клиентской части); Q_S – множество запрашиваемых информационных ресурсов; S_S - переменная, содержащая адрес web-сервиса, предоставляющего информацию (адрес серверной части).

Тогда $ServI(C_S,Q_S,S_S)$ – соответствие интерфейсов взаимодействия web-сервисов C_s и S_s, выражающее структуру интерфейсов, реализованных между сервисом по адресу C_s и сервисом по адресу S_s, и передающей информацию Q_s между ними.

Автоматизированное построение интерфейсов выражается в виде следующего соответствия φ_1.

Соответствие φ_1 является отображением области определения $ServIRend(C_S,Q_S,S_S)$ в область значений

$$\bigcup_{i_{SOA} \in I_{SOA}} ServI(c_i, q_i, s_i),$$

где C_s, Q_s, S_s – более общие параметры для множественного создания сервисов; c_i,q_i,s_i – частные параметры для создания сервисов; i_{SOA} – элемент множества I_{SOA}; I_{SOA} – множество интерфейсов, которые подходят по данным параметрам.

Для работы системы авто построения интерфейсов web-сервисов необходимо описать концептуальную модель взаимодействия web-сервисов.

На рис. 1 приведен пример композитных сервисов и сервисных кластеров в SOA-системе.

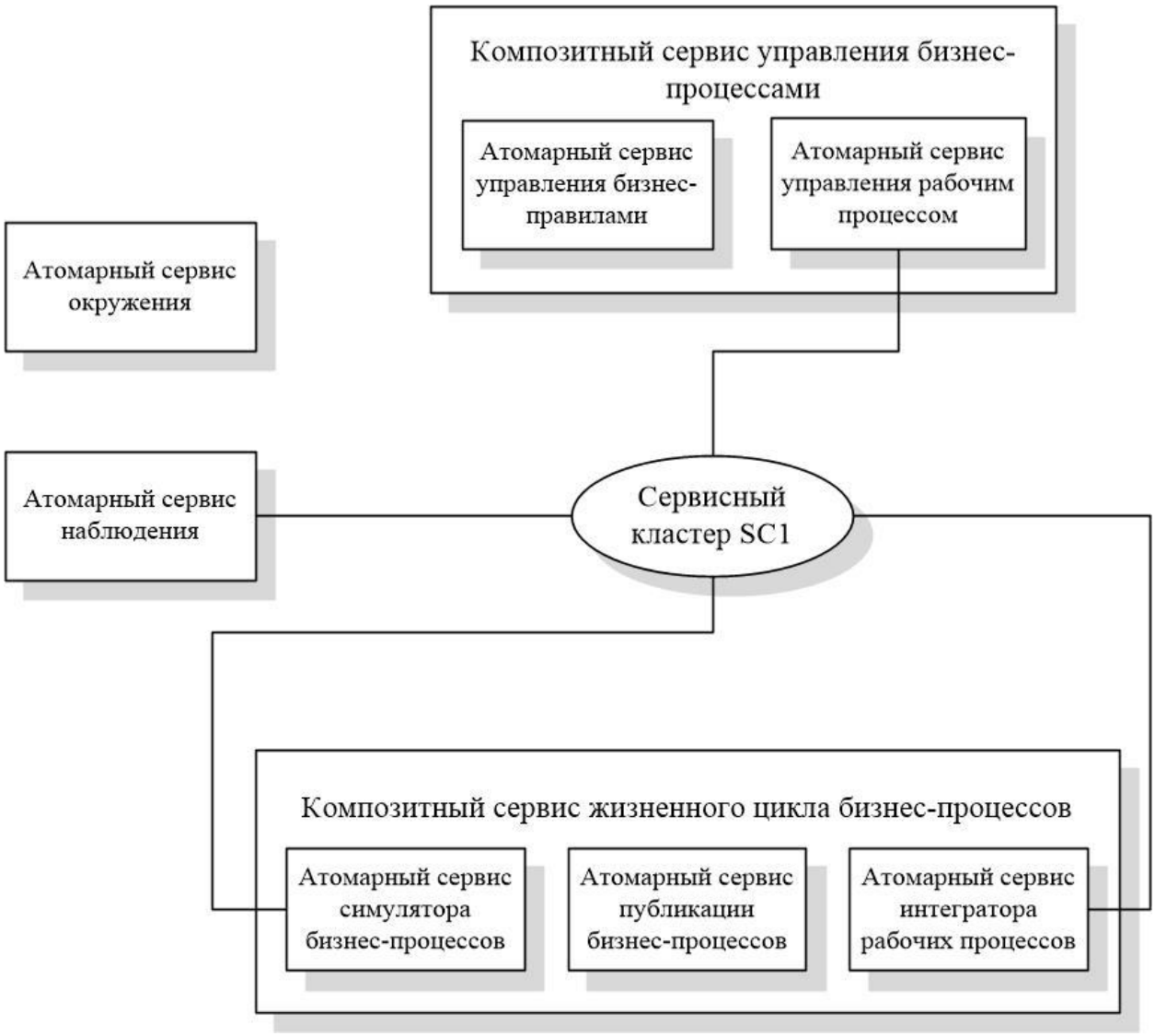

Рис. 1. Пример композитных сервисов и сервисных кластеров управления бизнес процессами

Далее в диссертации дается определение дескриптивной логики, и приводятся примеры работы с языками логики. Устанавливается, можно ли описать концептуальную модель работы SOA-системы средствами дескриптивной логики. Для этого сопоставим интерпретации конструкций дескриптивной логики, эквивалентные конструкциям языка теории множеств для элементов SOA-архитектуры, приведенным ранее. Результат сопоставления отображен в таблице.

Интерпретация I в дескриптивной логике – двойка

$$I = <\Delta, \bullet^I>,$$

где Δ – домен данной интерпретации, являющийся непустым множеством; $\bullet^I$ – интерпретирующая функция, сопоставляющая любому атомарному концепту A некоторое подмножество $A^I \subseteq \Delta$ и любой атомарной роли R некое подмножество $R^I \subseteq \Delta \times \Delta$.

Если сервисы b_s, b_{s1}, b_{s2} – элементы множества B_s, определенные категории B_s, B_{s1}, B_{s2}, B_{s3} – множества, входящие в универсальное множество U, интерфейс

R_s – отношение, то система множеств, описывающая SOA-архитектуру, подпадает под следующие ограничения (см. таблицу).

Синтаксис языка SOA-отношений

Теоретико-множественное описание элементов концептуальной модели SOA-архитектуры	Эквивалентные выражения интерпретации I в языке дескриптивной логики $\mathcal{ALC}$	Синтаксис языка дескриптивной логики $\mathcal{ALC}$
$b_s \in B_s$	$b^I \in B^I$	$B(b)$
$B_{s1} \subseteq B_{s2}$	${B_1}^I \subseteq {B_2}^I$	$B_1 \sqsubseteq B_2$
$B_{s1} \subseteq B_{s2} \cap B_{s3}$	${B_1}^I \subseteq {B_2}^I \cap {B_3}^I$	$B_1 \sqsubseteq B_2 \sqcap B_3$
$B_{s1} \cup B_{s2} \subseteq B_{s3}$	${B_1}^I \cup {B_2}^I \subseteq {B_3}^I$	$B_1 \sqcup B_2 \sqsubseteq B_3$
$B_{s1} \cup B_{s2} = \emptyset$	${B_1}^I \cup {B_2}^I = \emptyset$.	$B_1 \sqcup B_2 \sqsubseteq \bot$
$(b_{s1}, b_{s2}) \in R$	$({b_1}^I, {b_2}^I) \in R^I$	$R(b_1, b_2)$

Далее необходимо усовершенствовать формальное описание SOA-системы до возможности поддержки нюансов деления на ABox (набор утверждений) и TBox (набор аксиом) в дескриптивной логике. Следовательно, необходимо расширить язык SOA-отношений элементом конструкции полного экзистенциального ограничения роли $\exists R.B$ с семантикой {e $\in \Delta$ | существует $d \in \Delta$ такой, что $<e,d> \in R^I$ и $d \in B^I$}, для присваивания отношений к концептам.

Требуется установить необходимую выразительность языка дескриптивной логики для формального описания SOA-системы на основе необходимых элементов конструкции:

- включение концептов;
- пересечение с правой стороны аксиомы;
- объединение с левой стороны аксиомы;
- дизъюнкция концептов;
- полное экзистенциальное ограничение.

Из таблицы следует, что необходимая выразительность формального описания SOA-системы соответствует выразительности логики $\mathcal{ALC}$.

В то же время, использование дескриптивной логики $\mathcal{ALC}$ предоставляет дополнительную выразительность. Самая значимая из возможностей – это элемент конструкции отрицания концепта. Отрицание концепта позволяет обозначить, какие интерфейсы и сервисы не должны создаваться, а попытка их создания является ошибкой.

Сервисы внутри SOA-системы могут относиться к нескольким категориям, определенным разработчиками, например делящим сервисы по территориальным и функциональным признакам. Категории, в свою очередь, могут включаться в другие категории. Каждый сервис может иметь интерфейс взаимодействия с другими сервисами. Следовательно, категоризацию композитных сервисов и сервисных кластеров можно представить в виде следующей формализации в рамках теории множеств.

При построении онтологической модели SOA-системы следует учесть активную работу SOA-архитектуры с базами данных (БД) и возможность использования БД для обработки больших объемов данных. Такую систему можно построить с помощью технологии OBDA. Этот подход использует преимущества ДЛ для работы с онтологиями с целью извлечения запросов к БД. Основной принцип работы систем OBDA заключается в трансформации (переписывания) запросов к базе знаний в соответствующие запросы к БД. Базы данных в технологии OBDA являются структурой ABox дескриптивной логики.

Конъюнктивный запрос к базе знаний K – конъюнктивный запрос, атомы которого имеют вид $A(z)$ или $P(z_1, z_2)$, где A и P – соответственно атомарные концепты и атомарные роли в K; z, z_1, z_2 – константы в K или переменные. Пусть запрос q – конъюнктивный запрос или объединение конъюнктивных запросов. Ответ на q к базе знаний K есть множество $ans(q, K)$ кортежей $\vec{a}$ из констант, входящих в K, таких что $\vec{a}^M \in q^M$ для каждой модели M_c в K_c. По определению $ans(q, K)$ конечно, так как база знаний K конечна, и число констант, входящих в K, конечно. Отметим также, что кортеж $\vec{a}$ может быть пустым в случае, в котором q является булевым конъюнктивным запросом. В этом случае множество $ans(q, K)$ состоит из единственного пустого кортежа тогда и только тогда, когда формула q верна в каждой модели K.

Формальное обозначение конъюнктивного запроса

$$q(\vec{x}') \leftarrow conj'(\vec{x}', \vec{y}'),$$

где $conj'(\vec{x}', \vec{y}')$ – список атомов в $conj(\vec{x}, \vec{y})$, полученный после приравнивания переменных $\vec{x}$, $\vec{y}$ в соответствии с равенством в $conj(\vec{x}, \vec{y})$. Элемент конструкции $q(\vec{x}')$ является заголовком запроса q, в то время как $conj'(\vec{x}', \vec{y}')$ - телом. Переменные в $\vec{x}'$ называются распознаваемыми переменными. Переменные в $\vec{y}'$ являются нераспознаваемыми переменными.

Обозначения дизъюнкции конъюнктивных запросов в формате хранилищ данных обозначаются в виде множества:

$$q = \{q_1, ..., q_n\},$$

где каждый q_i – выражение, соответствующее конъюнктивному запросу:

$$q_i = \{\vec{x} \mid \exists \vec{y}_i . conj(\vec{x}, \vec{y}_i)\}, i = 1,2,...,n.$$

Входными значениями C_s,Q_s,S_s для *ServIRend*(C_s,Q_s,S_s) будут конъюнктивные запросы следующего вида:

$$C_s(\vec{x}'_C) \leftarrow conj'(\vec{x}'_C, \vec{y}'_C),$$

$$Q_s(\vec{x}'_Q) \leftarrow conj'(\vec{x}'_Q, \vec{y}'_Q),$$

$$S_s(\vec{x}'_S) \leftarrow conj'(\vec{x}'_S, \vec{y}'_S),$$

где $\vec{x}'_C$, $\vec{y}'_C$ – переменные запроса C_s, $\vec{x}'_Q$, $\vec{y}'_Q$ – переменные запроса Q_s, $\vec{x}'_S$, $\vec{y}'_S$ – переменные запроса S_s.

Выходными данными отображения φ_1 в область значения являются:

$$\bigcup_{i \in P} ServI_i(c_i, q, s_i),$$

где для каждого элемента c_i из *ans*(C, K) между парой c_i и s_i из *ans*(S, K) строится интерфейс, по которому передается обработанная структура в виде множества *ans*(Q, K). Тогда

$$p_{SOA} = a_{ans} * b_{ans},$$

где p_{SOA} – количество элементов множества I_{SOA}; a_{ans} – количество элементов множества *ans*(C, K); а b_{ans} – количество элементов множества *ans*(S, K).

Значения для области определения соответствия называются входными параметрами с зависимыми условиями, если $x \in x_C'$, $x \in x_S'$, где x – произвольный элемент условия запроса. В остальных случаях значения для соответствия называются входными параметрами без зависимых условий.

Далее представлена оптимизация множества *ans* для соответствия *ServIRend*. Роли в онтологической SOA-модели используются для регистрирования информации о добавляемых интерфейсах. Определим множество аксиом и утверждений базы знаний K, содержащих роли, как *RBox*. При обработке входных параметров соответствия *ServIRend* информации о ролях не требуется. Следовательно, нет необходимости учитывать *RBox* при выполнении реализации представления *ServIRend*. Рассмотрим интерпретацию ответа на запрос к базе данных без учета *RBox*. Пусть запрос q – конъюнктивный запрос или объединение конъюнктивных запросов, а база знаний $K_c = K/RBox$, тогда ответ на q к K_c есть множество *servAns*(q, K_c) кортежей $\vec{a}$ из констант, входящих в K_c, таких что $\vec{a}^{M_c} \in q^{M_c}$, для каждой модели M_c в K_c.

Отсутствие *RBox* по определению уменьшает размер *TBox* и *ABox*, что частично решает проблему разрастания запроса при ответе на OBDA-запрос.

Для описания OBDA-онтологий используется язык *DL-Lite*$_{\mathcal{R}}$ семейства логик *DL-Lite*. Этот язык подходит для задач OBDA. На логике *DL-Lite*$_{\mathcal{R}}$

основан язык OWL 2 QL консорциума W3C и входящего в состав формального инструментария технологий Semantic Web.

Язык *DL-Lite*$_{\mathcal{R}}$ в отличие от языка логики $\mathcal{ALC}$ оптимизирован для работы с OBDA. Класс вычислительной сложности логики *DL-Lite*$_{\mathcal{R}}$ меньше, чем логики $\mathcal{ALC}$. Вследствие этих факторов для описания онтологической SOA-модели эффективнее использовать логику *DL-Lite*$_{\mathcal{R}}$. *DL-Lite*$_{\mathcal{R}}$ обладает низкой выразительностью. В то же время для редукции до поддерживаемых языком конструкций можно использовать приведение к доступным элементам конструкции без увеличения вычислительной сложности. Далее представлены выражения, доступные при редукции:

$$B_1 \sqcup B_2 \sqsubseteq C \quad \equiv_T \quad B_1 \sqsubseteq C, B2 \sqsubseteq C;$$

$$B \sqsubseteq C_1 \sqcap C_2 \quad \equiv_T \quad B \sqsubseteq C_1, B \sqsubseteq C_2;$$

$$B_1 \equiv B_2 \quad \equiv_T \quad B_1 \sqsubseteq B_2, B_2 \sqsubseteq B_1;$$

$$B \sqsubseteq \exists R.\top \quad \equiv_T \quad B \sqsubseteq \exists R;$$

$$B \sqsubseteq \exists R.\bot \quad \equiv_T \quad B \sqsubseteq \neg\exists R;$$

$$B \sqsubseteq \exists R_0.C \quad \equiv_T \quad B \sqsubseteq \exists R_1, R_1 \sqsubseteq R_0, \exists R^{-}_1 \sqsubseteq C,$$

где $\equiv_T$ – таксономическая эквивалентность. Для удобной визуализации можно преобразовывать онтологии в формат логики $\mathcal{ALC}$. В связи с наличием в $\mathcal{ALC}$ полного экзистенционального ограничения появляется возможность отображать онтологии стандартными классификаторами и редакторами онтологий.

Далее представляются преимущества онтологического подхода к построению SOA-системы по сравнению с существующими аналогами.

1. Скорость выполнения алгоритмов для задач логического вывода у технологий дескриптивной логики выше, чем у структурированных алгоритмов.

2. Возможность взаимодействия SOA-системы с другими алгоритмами логического вывода, онтологиями и языками ДЛ. Как показано на примере использования логик *DL-Lite*$_{\mathcal{R}}$ и $\mathcal{ALC}$ для разных задач, благодаря схожести принципов их организации, можно использовать эффективность определенных инструментариев этих логик для определенных задач.

3. Интеграция SOA-системы с реализациями технологии Semantic Web и последующее их развитие. Интеграция возможна благодаря подобию языка web-онтологий OWL 2 QL и логики *DL-Lite*$_{\mathcal{R}}$. Актуальность технологии Semantic Web описана в разделе 1.3 диссертации.

4. Удобство и простота расширения SOA-системы. При увеличении количества элементов сервис-ориентированной системы не требуется расширения алгоритмов новыми структурами, требуется только расширение онтологии необходимыми аксиомами и утверждениями.

Далее обусловливается необходимость применения алгоритмов авто-построения интерфейсов web-сервисов.

В третьей главе приведены описание и принципы работы SIR-алгоритмов. Далее представлено описание нового алгоритма *SIR1* для обработки входных параметров построения интерфейсов. Алгоритм возвращает $\bigcup_{i \in P} Serv_i(c_i,q,s_i)$ для последующего построения интерфейсов взаимодействия web-сервисов. Входными параметрами алгоритма запросов к онтологии являются TBox T и тройка $i_{SOA}(C, Q, S)$, называющаяся далее SIR-триплет. Выходными данными алгоритма является объединение(*SIRSet*) SIR-триплетов(*SIRTrip*) с запросами к БД. На рис. 2 представлена блок-схема алгоритма *SIR1, основанного на алгоритме SIR*.

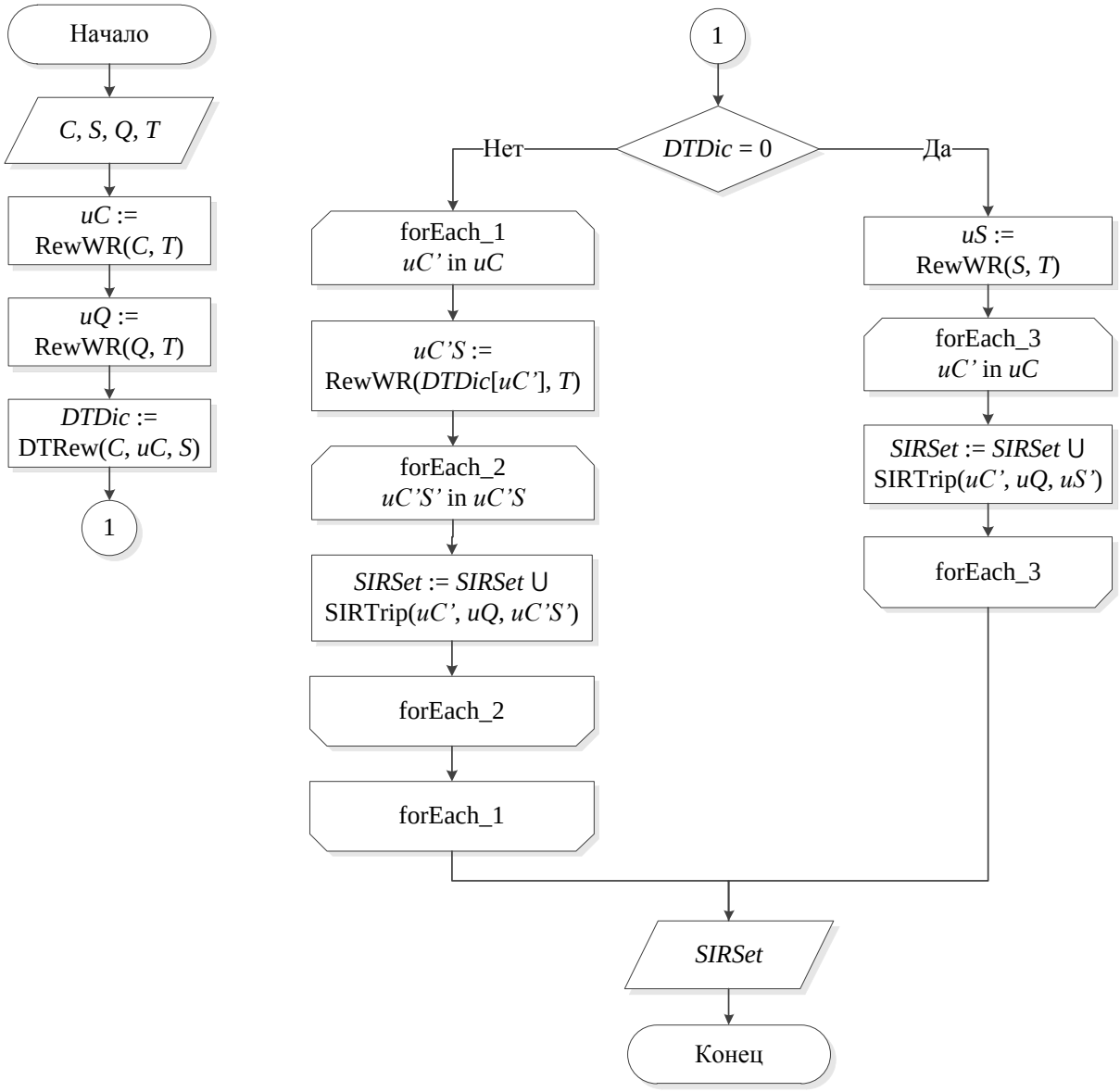

Рис. 2. Блок-схема алгоритма *SIR1*

Далее представлен принцип работы алгоритма *SIR1*. За переписывание запросов отвечает алгоритм *RewWR*. Алгоритм *SIR1* состоит из следующих действий.

1. Посредством алгоритма переписывания запросов переписать множество *C* и поместить объединение запросов в множество *uC*.

2. Переписать запрос *Q* и поместить объединение запросов в множество *uQ*.

Если входные параметры не содержат зависимых элементов, то произведутся следующие действия.

1. Переписать запрос *S* и поместить объединение запросов в множество *uS*.

2. Для каждого *uC'* из *uC* создать SIR-триплет, где утверждение о клиенте в ABox = *uC'*, утверждение о передаваемой информации в ABox = *uQ'* из множества *uQ*, утверждение о сервисе в ABox = *uS'* из множества *uS*.

3. Поместить триплет в массив *SIRSet*.

Если входные параметры содержат зависимые элементы, то произведутся следующие действия.

1. Для обработки параметров с зависимыми элементами будет использоваться функция *DTRew*, представленная ранее.

2. После завершения работы функции для каждого элемента *uC'* из множества *uC* переписать запрос *DTDic*[*uC'*] и поместить результат в множество *uC'S*.

3. Для каждого *uC'S'* из *uC'S* создать SIR-триплет, где утверждение о клиенте в ABox = *uC'*, утверждение о передаваемой информации в ABox = = *uQ'* из *uQ*, утверждение о сервисе в ABox = *uC'S'* из множества *uC'S*.

В разделе 3.3 диссертации работы доказывается лемма о завершаемости алгоритма *SIR1*.

В разделе 3.3 диссертации доказывается лемма о том, что время выполнения алгоритма *SIR1* является полиномиально зависимым от размера *T*.

В разделе 3.3 диссертации доказывается лемма о том, что вычислительная сложность алгоритма *SIR1* для параметров без зависимых элементов не превысит полиномиальную сложность от размера TBox.

Далее требуется реализовать в SOA-системе возможность использования CRUD (Create Read Update Delete). CRUD – сокращенное название стандартных операций при работе с хранением данных: создание, просмотр, обновление и удаление.

Реализацией операции «создание» является алгоритм *SIRCreate*. В этом алгоритме, пользуясь ранее рассмотренными алгоритмами, необходимо получить информацию из ABox и обработать ее, формируя при этом конструкции для построения интерфейсов. Этот алгоритм можно использовать

для операции обновления, так как при повторном запуске алгоритма на уже использованных параметрах необходимые данные перепишутся без повреждения структуры системы.

Пусть $K = <T, A>$ – база знаний логики $DL\text{-}Lite_{\mathcal{R}}$, C, Q, S – конъюнктивные запросы. На рис. 3 представлен алгоритм *SIRCreate* с входными параметрами: *C, S, Q, T, A*.

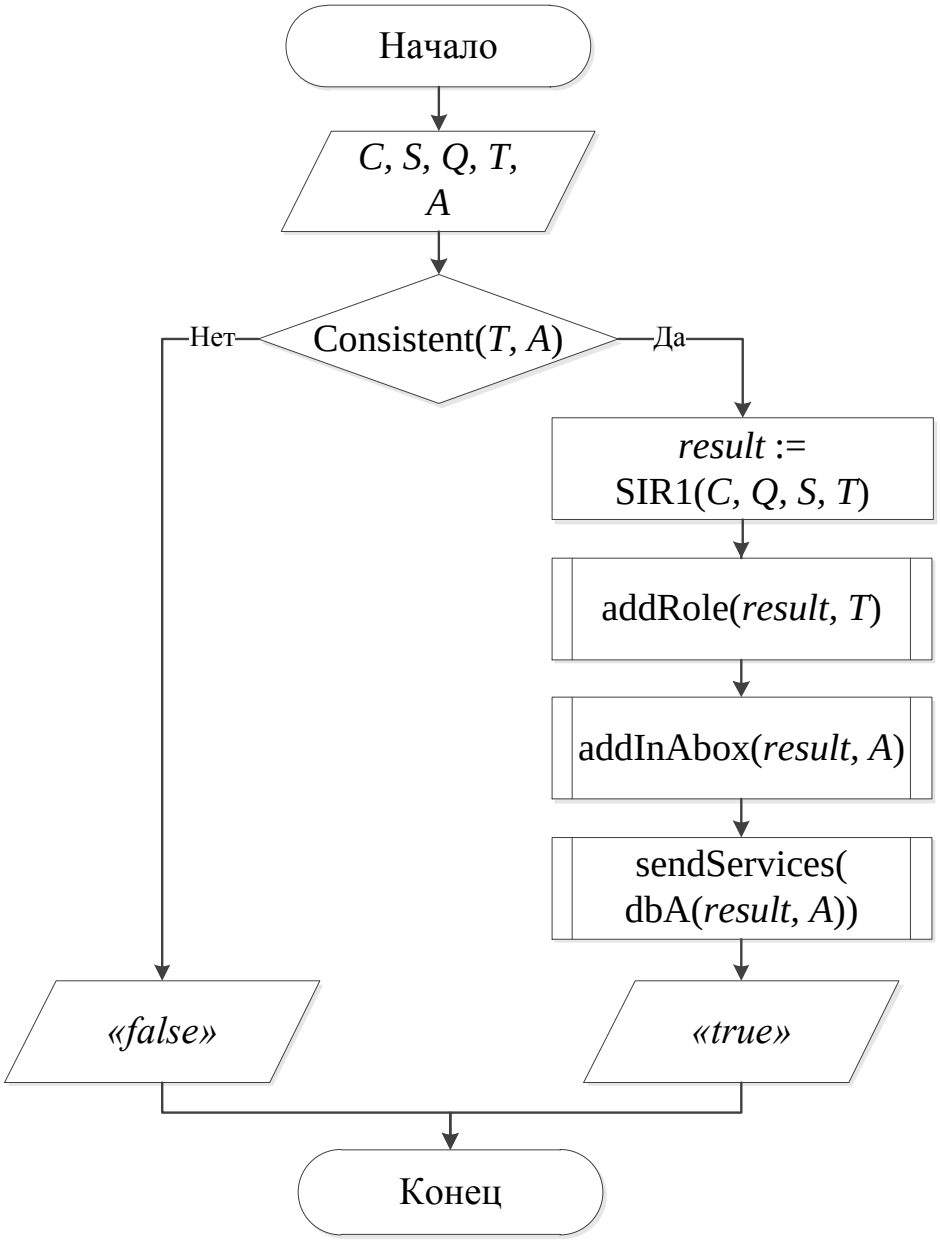

Рис. 3. Блок-схема алгоритма *SIRCreate*

Функция Consistent реализует проверку базы знаний на непротиворечивость. Если БД оказывается противоречивой, то нет смысла проводить дальнейшие действия и алгоритм завершается.

Процедура addRole добавляет информацию о новых интерфейсах в TBox *T*. Процедура addInAbox добавляет информацию о новых интерфейсах в ABox *A*.

Функция dbA выполняет запросы в триплете над ABox *A*, являющемся базой данных. Процедура sendServices отвечает за формирование кода с запросами под БД web-сервисов и его отправку по адресам клиента и сервера на основе ответа ABox SOA-системы.

В разделе 3.5 доказывается теорема о завершаемости алгоритма *SIRCreate* на основе леммы 3.6 из раздела 3.3 диссертации.

В разделе 3.5 доказывается теорема о том, что алгоритм *SIRCreate* является полиномиально зависимым от размера БЗ *K* на основе леммы 3.7 из раздела 3.3.

В четвертой главе представлена программная реализация онтологической SOA-системы SIRSystem. Программный код системы написан на языке программирования Java. Он тестировался на ОС CentOS, 3 ГБ ОЗУ, AMD Phenom(tm) II N850 Triple-Core Processor 2,20 Гц. Данные из ABox в этой реализации хранятся в СУБД MySQL. Для основных задач логического анализа онтологий использовался классификатор HermiT. Для описания интерфейсов web-сервисов использованы стандарты SOAP и WSDL. Для сравнения использовались 3 SIR-алгоритма: *SIRR* с использованием стороннего алгоритма PerfectRef; *SIRR,* с использованием стороннего алгоритма Requiem и алгоритм *SIR1,* оптимизированный под работу с SOA-системами.

Тестовые данные, на которых тестировался алгоритм, состоят из онтологий, написанных или адаптированных под язык OWL 2 QL для тестирования алгоритмов логического анализа (рис. 4) и онтологии SAMAO (рис. 5). SAMAO — онтология государственных и муниципальных учреждений, разработанная специально для системы автоматизированного построения интерфейсов взаимодействия web-сервисов.

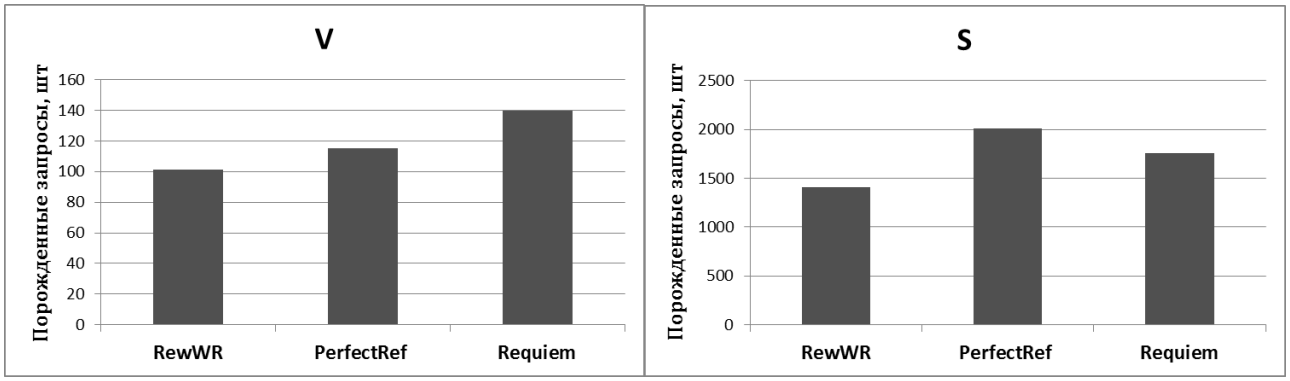

Рис. 4. Диаграма сравнения скорости работы алгоритмов с онтологией VICODI (слева) и StockExchange (справа)

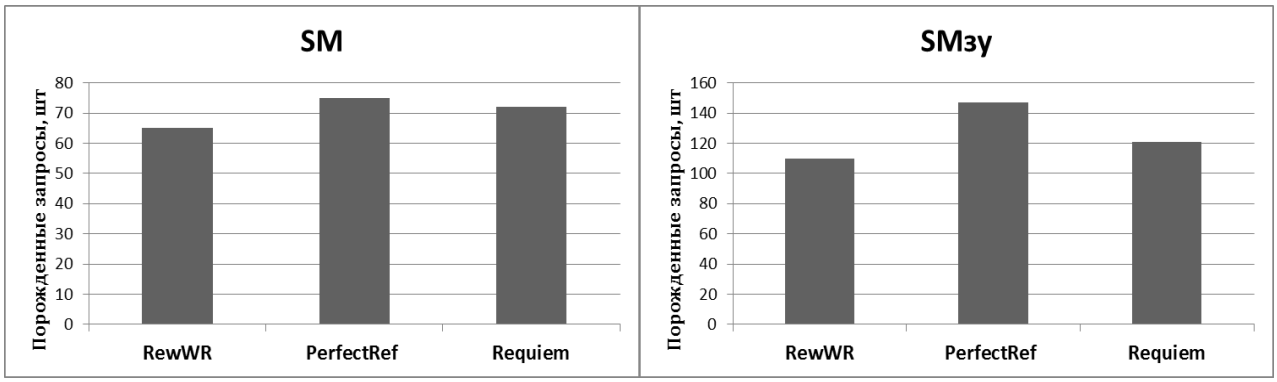

Рис. 5. Диаграма сравнения скорости работы алгоритмов с онтологией SAMAO без зависимых условий (слева) и с зависимыми условиями (справа)

В результате экспериментов зафиксировано среднее увеличение эффективности созданных web-сервисов на базе web-сайтов примерно на 23 %.

В заключении приводится обобщение результатов, полученных в процессе решения поставленной научно-технической проблемы, связанной с созданием онтологических SOA-систем.

Основные результаты работы

1. Проведен анализ современных технологий для проектирования в рамках парадигмы SOA. Приведена классификация аналогичных технологий и подходов. Рассмотрены языки описания взаимодействия элементов SOA-систем. Выявлены недостатки существующих подходов к проектированию интерфейсов web-сервисов и EAI-систем. Проанализированы основные виды структур взаимодействия SOA-сервисов. Приведен краткий обзор экспериментальных реализаций систем OBDA, работающих с семейством дескриптивных логик DL-Lite. Наличие таких реализаций подтверждает практическое применение дескриптивных логик и языков web-онтологий в целях предоставления онтологического доступа к данным.

2. Разработан новый *формализм* для описания и анализа концептуальной модели SOA-системы, ориентированный на онтологический подход и позволяющий описывать взаимодействие web-сервисов.

3. Получены новые алгоритмы автоматизированного построения интерфейсов web-сервисов и CRUD-алгоритмы для SOA-архитектуры, дающие возможность автоматизированно строить интерфейсы web-сервисов. Алгоритмы отличаются от аналогов скоростью построения и более общей моделью. Новый алгоритм трансформации запросов под SIR-систему отличается от более общих алгоритмов трансформации запросов более быстрой работой за счет игнорирования несущественных элементов входных данных. Разница в скорости работы между полученным алгоритмом и аналогами зависит от количества несущественных элементов в онтологии.

4. Получены эффективные проектные решения программной реализации SIR-алгоритмов на основе ООП.

5. Доказаны теоремы о завершаемости и вычислительной сложности SIR-алгоритмов для онтологических SOA-систем.

Список публикаций по теме диссертации

Статьи в изданиях по списку ВАК:

1. *Каширин И.Ю., Курдюков Н.С.* SIR – алгоритм автоматического построения интерфейсов взаимодействия web-сервисов // Вестник РГРТУ, № 3 (выпуск 45), Рязань, 2013. С. 75 – 79.

2. *Каширин И.Ю., Курдюков Н.С.* Доказательство эффективности SIR алгоритма авто-построения интерфейсов взаимодействия web-сервисов // Фундаментальные исследования № 6. Часть 2. Научный журнал. Издательский дом «Академия Естествознания». 2013. С. 267 – 273.

3. *Курдюков Н.С.* Доказательство эффективности алгоритма SIRCreate // Электронный научный журнал «Современные проблемы науки и образования» № 2, 2014. Режим доступа: http://www.science-education.ru/

Статьи в сборниках научных трудов:

4. *Курдюков Н.С.* Интеллектуальный анализ неструктурированных текстов как средство автоматизации процесса расширения ресурсов Semantic Web // Математическое и программное обеспечение вычислительных систем: Межвуз. сб. науч. Тр. / Под ред. А.Н. Пылькина – Рязань: РГРТУ, 2012. С. 64 − 67.

5. *Курдюков Н.С.* Доказательство эффективности алгоритма SIRDelete // Международный научно-исследовательский журнал, №2 (21) Часть 1. – Екатеринбург : МНИЖ, 2014. С.99 – 101.

6. *Курдюков Н.С.* Средства технологии Semantic Web // Математическое и программное обеспечение вычислительных систем: Межвуз. сб. науч. Тр. / Под ред. А.Н. Пылькина - Рязань(РГРТУ), декабрь 2013. С.32 − 35.

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

Тезисы докладов на международных и всероссийских конференциях:

8. *Курдюков Н.С.* Основанный на онтологиях доступ к данным формата Semantic Web // Общество, современная наука и образование: проблемы и перспективы: сборник науч. тр. по материалам Международной научно-практической конференции. Часть 5. Тамбов. Бизнес – Наука – Общество, 2012. С. 67 − 68.

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

10. *Курдюков Н.С.* Анализ вычислительной сложности логического вывода для языков web-онтологий // Wyksztalcenie I nauka bez granic. Materialy VIII miedzynarodovej naukowi-praktycznej konferencji. Vol. 34. Przemysl. Nauka i studia, 2012. P. 18 −20.

11. *Курдюков Н.С.* Реализации основанного на онтологиях доступа к данным формата Semantic Web // Общество, современная наука и образование: проблемы и перспективы: сборник научных трудов по материалам Международной научно-практической конференции. Часть 5. Тамбов. Бизнес-Наука-Общество, 2012. С. 68 − 70.

12. *Курдюков Н.С.* SOA-системы в технологиях Internet и Semantic Web // Новые информационные технологии в научных исследованиях: материалы XVIII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. Рязанский государственный радиотехнический университет, 2013. 344 с.

13. *Курдюков Н.С.* Запросы при описании SOA-архитектуры в рамках OBDA-технологии // Materiały X Międzynarodowej naukowi-praktycznej konferencji «Strategiczne pytania światowej nauki - 2014» Vol. 33. Przemysl, 2014. P. 9 – 11.

14. *Курдюков Н.С.* Логический анализ онтологий SOA-систем // Материали за X международна научна практична конференция «НАЙНОВИТЕ НАУЧНИ ПОСТИЖЕНИЯ - 2014». – Том 30. Съвременни технологии на информации. – София. «Бял ГРАД-БГ» ООД, 2014. С. 19 − 20.

15. *Igor Yu. Kashirin, Nikolay S. Kurdyukov.* Efficiency Proof of Sirread Algorithm // Proceedings of ICCTPEA-2014, Saint Petersburg State University, Edited by: E. I. Veremey, Saint-Petersburg, Russia, June 30 – July 04, 2014. P. 75.

16. *Курдюков Н.С.* Алгоритм проверки непротиворечивости баз знаний SOA-систем // Materiały X Międzynarodowej naukowi-praktycznej konferencji «Dynamika naukowych badan - 2014» Vol. 8. Przemysl, Nauka i studia, 2014. P. 90 – 92.

Курдюков Николай Станиславович

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СЕРВИС-ОРИЕНТИРОВАННЫХ СИСТЕМ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ЯЗЫКОВ ДЕСКРИПТИВНОЙ ЛОГИКИ

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Подписано в печать_____ Формат бумаги 60×84 1/16.
Усл. печ. л. 1,0. Тираж 100 экз.
Рязанский государственный радиотехнический университет.
390005, г. Рязань, ул. Гагарина, 59/1.