МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра общей информатики

Направление подготовки: 230100 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Магистерская программа: Технология разработки программных систем

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

# Автоматизированные методы пополнения знаний на основе онтологий

Гуревич Михаил Ильич

Тема диссертации утверждена распоряжением по НГУ №___ от «___»_____20__г.
Тема диссертации скорректирована распоряжением по НГУ №___ от «___»_____20__г.

**«К защите допущена»**
Заведующий кафедрой,
д.ф.-м.н., доцент
Пальчунов Д.Е./……..
(фамилия , И., О.) / (подпись, МП)
«……»……20…г.

**Научный руководитель**
Профессор, зав. кафедрой, ИМ СО РАН,
д.ф.-м.н., доцент
Пальчунов Д.Е./……...
(фамилия , И., О.) / (подпись, МП)
«……»……20…г.

Дата защиты: «20» июня 2014г.
Автор Гуревич М.И./…….
(фамилия , И., О.) / (подпись)

Новосибирск, 2014г.

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра общей информатики

(название кафедры)

Направление подготовки: 230100 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Магистерская программа: Технология разработки программных систем

УТВЕРЖДАЮ
Зав. кафедрой Пальчунов Д.Е.
(фамилия, И., О.)
…….
(подпись, МП)
«……»……20…г.

**ЗАДАНИЕ**
**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**
**МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

Студенту Гуревичу Михаилу Ильичу

(фамилия, имя, отчество)

Тема Автоматизированные методы пополнения знаний на основе онтологий

(полное название темы магистерской диссертации)

Исходные данные (или цель работы): разработка алгоритма и создание программы выполняющей пополнение и логический вывод знаний из множества бесквантерных предложений на основе данных о предметной области.

Структурные части работы: работа включает в себя изучение предметной области, описание главной идеи, а также описание реализованной программы

Научный руководитель
Профессор, зав. кафедрой, ИМ СО РАН,
д.ф.-м.н., доцент
Пальчунов Д.Е./……..
(фамилия , И., О.) / (подпись)
«…»……20…г.

Задание принял к исполнению
Гуревич М.И./……..
(ФИО студента) / (подпись)
«…»……20…г.

# Оглавление

## ВВЕДЕНИЕ

Данная работа выполнена в Институте математики им. С.Л. Соболева СО РАН и представляет собой исследование и реализацию нового алгоритма пополнения заданного множества бесквантерных предложений новыми, заведомо истинными предложениями, путем объединения данных истинности предложений и информации о предметной области в онтологии. Результаты и методы, описываемые в данной работе, могут быть использованы, как дополнительный этап работы в программах, работающих с неполными данными, такими как системы логического вывода и принятия решений. Также результаты данной работы могут быть использованы в качестве дополнительного этапа вычисления значений истинности из данных, полученных в работе Махасоевой Ольги "Автоматизированные методы построения атомарных диаграмм моделей по текстам естественного языка".

Системы, которые описывают некоторую предметную область, зачастую бывают неполны. Это происходит из-за невозможности внесения больших объемов данных, из-за нехватки знаний человека, либо человечества в целом или из-за нехватки времени. В любом случае, построение модели предметной области является сложной, а порой невыполнимой задачей. Таким образом, работа с неполными или нечеткими данными является важным объектом исследований.

Стоит отметить, что настоящая работа является продолжением развития представленных исследований. В [1] изложены методы автоматизированной разработки онтологий предметных областей на основе языка лингвистических шаблонов и наборов правил для выделения данных отношений между понятиями. В [2] исследованы методы порождения новых знаний, основанные на интеграции знаний, представленных в разных текстах естественного языка, и на осуществление логического вывода из имеющихся знаний. В [3] изложен теоретико-модельный подход к обработке нечёткой и неполной информации, основанный на теории нечётких моделей. На основе этого подхода разработана программная система управления рисками при обеспечении информационной безопасности предприятия. В [4] рассмотрены современные подходы к задаче обеспечения информационной безопасности компании. Проанализированы предложенные этими подходами методы работы с начинающимся нападением. Предложен новый подход ранней диагностики кибератаки. Данный подход основан на построении формальных моделей прецедентов компьютерных атак и адаптации к этим моделям ДСМ-метода автоматического порождения гипотез. Изложены математические основы и

описана программная реализация предлагаемого подхода. В [5] рассматривается проблема согласования знаний по компьютерной безопасности, извлеченных из разных текстов на естественном языке. Дается описание поставленной задачи с помощью теоретико-модельного формализма. Знание о конкретной компьютерной атаке формализуется в виде недоопределенной алгебраической системы (названой обобщенным прецедентом). База знаний представляет собой множество обобщенных прецедентов. Согласованное значение истинности предложения вычисляется в виде интервала, определенного на отрезке [0, 1]. В статье приведены алгоритмы вычисления согласованного значения истинности, описана программная реализация разработанных методов.

В настоящей работе исследуется возможность пополнения известного множества предложений из знаний о предметной области, которую они описывают. Целью настоящей работы является разработка и реализация алгоритма пополнения множества бесквантворных предложений новыми, а также обозначение необходимых условий такого пополнения. Актуальность проводимого исследования определяется возможностью использования подхода и программной реализации в реальных системах с неполной информацией, в областях распознавания образов, логического вывода, теории игр, оптимизации и других. Для достижения цели были поставлены следующие задачи:

1. Анализ свойств предметных областей и их онтологий.
2. Постановка и обоснование метода пополнения множества бесквантворных предложений новыми.
3. Разработка прототипа, реализующего разработанный алгоритм.

Данная работа состоит из трех основных глав: первая часть содержит общую проблематику и анализ существующих решений, вторая часть общую идею и описывает архитектуру программы и используемые в ней компоненты, третья часть содержит некоторые примеры работы программы.

# 1 Формулировка задачи

В данной главе приведено краткое описание предметной области, даны основные определения, описаны существующие проблемы, а так же уже существующие близкие программные решения.

### Описание предметной области

Человеческие знания, в удобном для обработки компьютером виде, могут храниться в различных системах. К таким системам можно отнести базы знаний, экспертные системы, семантические сети и другие. Каждая такая система описывает важные параметры, факты, объекты системы и их связи. Благодаря данным, которые находятся в таких системах, а так же механизмам автоматического логического вывода, становится возможным находить новые связи в данных, производить вывод новых знаний, а так же доказывать гипотезы. Данные системы в настоящий момент довольно хорошо развиты и используются во многих областях. Так, например, экспертные системы позволяют решать такие задачи, как диагностика заболеваний [6], мониторинг и управление различных объектов и многое другое. Одними из наиболее важных требований к таким системам являются актуальность и достоверность информации в системах, а так же релевантность информации, которую можно получить благодаря правилам вывода.

Создание модели знаний предметной области является сложной и трудоемкой задачей. Можно выделить следующие основные этапы:

1. Изучение предметной области. На данном этапе происходит поиск основных понятий и взаимосвязей. Так же проводится отбрасывание несущественных понятий для ускорения обработки данных на последующих этапах.
2. Извлечение знаний из экспертов предметной области.
3. Формализация полученных знаний на языке системы.

Обычно формализаций полученных от эксперта знаний занимается специальный человек или группа людей, которые интегрируют знания экспертов предметной области в некоторую связную и непротиворечивую систему. Такая интеграция связана с необходимостью нахождения и обработки всех нюансов, связанных с различием подачи знаний экспертами, а так же глубокого изучения предметной области. Однако, довольно объемная часть знаний содержится в опыте и голове эксперта, которые он не в состоянии

выразить. Для того, чтобы формализовать такие знания, применяются специальные методы их извлечения. Однако данный этап составления модели предметной области до сих пор остается самым сложным и трудоемким.

В качестве объекта исследования в настоящей работе были выбраны множества бескванторных предложений логики первого порядка. Напомним, что логика первого порядка это формальное исчисление относительно переменных, фиксированных функций и предикатов [7]. Данный выбор обусловлен тем, что логика предикатов первого порядка является полной и непротиворечивой. Данные свойства делают её привлекательной для формализации знаний в описанных ранее системах. Напомним основные определения, которые будут использоваться далее:

Константа — функциональный символ длинны 0.

Предикат — символ, обозначающий какое-то свойство или отношение.

Терм — константа.

Атом — предикат от термов.

Формула — либо атом, либо одна из следующих конструкций: $\neg A$, $A \cap B$, $A \cup B$ и $A \rightarrow B$, где A и B — формулы.

Так же стоит напомнить, что формула может принимать два значения — истина и ложь. А итоговое значение сложной формулы вычисляется исходя из значений истинности и ложности входящих в неё подформул.

В качества источника знаний для пополнения исходного множества предложений исследуются знания из онтологии описываемой предметной области. Существует множество определений этого понятия. Дадим одно из них, которое дано в [8]: формальной онтологией предметной области SD назовём пару $O=<A,\sigma>$, где $\sigma$ — множество ключевых понятий предметной области, и A — множество аналитических предложений, описывающих смысл этих ключевых понятий. Множество T предложений, которые являются верными в каждом примере предметной области, будем называть теорией предметной области SD.

В последние годы формальное описание предметных областей становится все популярнее. Создаются множества словарей в различных областях, например, медицинские SNOMED, UMLS и онтология товаров и услуг UNSPSC. Таким образом, в настоящей работе исследуется возможность использования таких систем для пополнения знаний.

**Описание задачи**

Рассмотрим пример. Пусть имеется некоторый эксперт предметной области, которая описывается языком $\delta_a$. Он знаком и встречался с некоторым конечным набором ситуаций из этой предметной области. Это множество можно рассматривать, как вероятностное пространство, где элементарные исходы - экземпляры предметной области. Очевидно, что эксперт не знает полного описания каждой ситуации, он тем более не знает всех ситуаций, рассматриваемых другими экспертами в разное время (в частности, поскольку количество таких предложений бесконечное множество). Однако, существует некоторое конечное множество предложений, которые эксперт точно знает и может формализовать. Обозначим это множество предложений $U_a$.

Пусть также имеется онтология описываемой предметной области. Тогда, если эксперт сумел описать лишь некоторые логические закономерности, то сравнивая эти данные с данными из онтологии можно найти новые неописанные экспертом знания. Например, если эксперт утверждает, что некоторый объект является "полем", то, если мы рассматриваем предметную область алгебры, то тогда автоматически следует, что этот объект является "кольцом". Данные рассуждения были бы не верны, если бы описывалась предметная область футбола. Таким образом, такие выводы можно делать имея лишь представления о предметной области, что позволяет автоматически пополнять имеющиеся знания эксперта.

Рассмотрим другой пример, который лучше раскрывает эту идею. Пусть имеется сигнатура $\zeta =< R, C >$ и некоторая модели $\mathfrak{A}$ данной сигнатуры, где R и C - кортежи предикатов и констант соответственно.

Пусть известно следующее предложение:

в_комнате_находится(лайка), где в_комнате_находится - одноместный предикат из R, а лайка - константа из C.

Сможет ли система, построенная на простом логическом выводе, выдать ответ истинности следующего предложения:

в_комнате_находится(собака).

Очевидно, что если системе не известны отношения между терминами "лайка" и "собака", то логический вывод невозможен. Однако, если каким-либо образом мы сможем ей сообщить, что "лайка" является породой собаки, то ответ истинности в этом примере будет очевиден. Заметим, что сообщая эти знания системе, мы описываем

рассматриваемую предметную область.

Описанные в настоящей работе алгоритм пополнения будет очень полезен в системах, которые извлекают знания из имеющихся источников в автоматическом режиме. Так, например, в работе Махасоевой Ольги по порождению математических моделей из текста на естественном языке, существования модуля пополнения знаний, извлеченных из теста на основе онтологии рассматриваемой предметной области, могло бы существенно улучшить количество извлеченных знаний.

Таким образом, была поставлена задача разработки алгоритма и автоматического программного комплекса пополнения знаний на основе данных онтологии предметной области.

## Сравнение с другими продуктами

В настоящий момент существуют программы помощи создания баз знаний. Если рассматривать программы, которые работают с логикой первого порядка, то нельзя не упомянуть программу Пролог [9]. Это система логического программирования, основанная на языке логики предикатов первого порядка. Она позволяет оперировать фактами, правилами логического вывода и запросами, которые могут описывать базы знаний. Опишем плюсы подобных систем:

1. Они реализованы и способны описывать базы знаний;
2. Способны производить логический вывод на основе имеющихся данных;
3. Продолжают развиваться и расширяться на другие парадигмы и языки программирования.

Наиболее важным преимуществом, с точки зрения данной работы, является то, что предлагаемая система пополнения предложений способна работать на более высоком уровне и оперировать не только данными об истинности предложений, но и информацией о предметной области в онтологии. Более того, предлагаемая система работает напрямую со множествами предложений и множествами возможных форматов описания онтологий, что делает ее независимой от других систем.

Другим близким решением является система Cyc [10]. Опишем плюсы данной системы:

1. Богатая система описания предметной области;
2. Существование коллекций констант и возможность определения их отношений;

3. Поддержка кванторов всеобщности и существования.

К минусам можно отнести следующее:

1. Огромная сложность системы из-за её архитектуры;
2. Добавление всех данных осуществляется вручную;
3. Система не доработана до конца;
4. Отсутствие документации и примеров.

Предлагаемое в данной работе решение позволяет решить наиболее важные проблемы - сложность архитектуры и добавление данных. Так как предполагается использование стандартных средств описания онтологий, то это позволит свести к минимуму необходимость изучения внутренней структуры системы, а так же позволит использовать повторно уже готовые онтологии предметных областей, который в настоящий момент создано большое количество. Также, как уже говорилось ранее в работе, система пополнения работает на более высоком уровне. Таким образом, пользователь будет в праве выбирать какой системой агрегирования знаний и логического вывода ему пользоваться.

Главным преимуществом данной системы является её универсальность и возможность работы с различными системами, которые используют логику первого порядка.

## Вывод

В данной главе было рассмотрено несколько аспектов работы. Была описана предметная область, обозначены основные определения и область исследования. Озвучены существующие проблемы, которые решает настоящая работа. Были описаны существующие на данный момент средства. Обозначены их плюсы и минусы, а так же было дано обоснование значимости проводимой работы. Так же была описана задача диплома и дано краткое описание идеи реализации программного продукта.

# 2 Формулировка идеи и реализации продукта

### Идея

Рассмотрим подробнее какие отношения может описывать онтология предметной области. Существуют различные категории отношений:

1. Отношения между классами;
2. Отношения между индивидами;
3. Отношения между индивидами и классами;
4. Отношения между объектом и группой объектов;
5. Отношения между группами объектов.

Предположим у нас имеется некоторая логическая модель, которая описывает некоторую предметную область. Пусть также мы имеем онтологию этой области. Так как модель и онтология описывают по существу одно и тоже, но с разных сторон, то можно попытаться произвести их сравнение.

Действительно, константы это индивиды и классы в онтологии. Таким образом между константами в логической модели существуют такие же отношения, что существуют между этими понятиями в онтологии. Выделим самые значимые из них для этой работы:

1. A is-a-superclass-of B - объект A является прародителем объекта B;
2. A is-a-subclass-of B - объект A является потомком объекта B;

Другими словами объект A в первом случаем является более общим понятием, чем объект B. Рассмотрим приводимый ранее пример в новом контексте:

в_комнате_находится(лайка) и в_комнате_находится(собака).

В данном случае факт того, что в рассматриваемой предметной области лайка и собака связаны отношением 2 (собака это более общее понятие, чем лайка), может являться правилом логического вывода, что если истинен предикат от константы "лайка", то истинен предикат от константы "собака". На самом деле, мы можем продолжить наш ряд обобщений: лайка, собака, домашнее животное, животное, существо, нечто. Однако будут ли последующие обобщения значимыми? Самым очевидным ограничением для подобного обобщения является присутствие понятий в рассматриваемом множестве констант имеющейся сигнатуры. Также огромное значение имеет тот факт, что подобные обобщения вообще могут быть применены в текущей предметной области. Так как, если слова "лайка" и "собака" не связаны, то подобные рассуждения будут неверны. Таким

образом, используемые важнейшим фактом возможности такого логического вывода должна быть согласованность онтологии и логической модели.

Также стоит заметить, что использование понятия обобщения можно и в обратную строну. Например, если нам известно, что в_комнате_находится(собака) ложно, то отсюда автоматически следует, что в_комнате_находится(лайка) также ложно, при учете всех вышеперечисленных замечаний.

Другое важное отношение, которое отдельно выделяется в некоторых онтологиях - отношение синонимии. Таким образом, если в предметной области имеются понятия, которые означают одно и тоже, то истинность предложений от одного понятия также будут истинны и от другого.

Таким образом, главная идея настоящей работы заключается в том, чтобы объединить знания о предметной области, а именно отношения между понятиями и логическую модель этой области. Благодаря такому объединению будет возможно возложить на систему автоматическое добавление заведомо истинных предложений. Более того, созданные общие и специфические онтологии предметных областей можно будет повторно использовать в других работах.

## Обоснование идеи и алгоритм пополнения

Предлагается использовать следующий алгоритм пополнения множества предложений:

1. Программе задается множество бесквантворных предложений известной сигнатуры;
2. Извлекаются все уникальные константы из имеющихся предикатов;
3. Для каждой пары констант ищутся все связи вида синонимии и обобщения. Если найдено отношение вида синонимия, то в каждом предложении одна константа заменяется на другую и наоборот. Если найдено отношение обобщения, то алгоритм разбивается на два этапа. Сначала мы ищем все истинные предложения с менее общим понятием и заменяем его на более общение. Далее мы ищем все ложные предложения с более общим и заменяем их менее общим. Таким образом мы получаем новые истинные предложения.

Предположим, что мы дополнили множество предложений некоторыми новыми по описанному выше алгоритму. Теперь необходимо оценить насколько истинными они

могут быть и от чего это зависит. Так как мы используем только заведомо существующие во входных данных предложения, то степень истинности полученных предложений сводится к релевантности произведенных замен констант. Будем считать, что константы в сигнатуре предметной области соответствуют тем, что содержатся в онтологии. Для определенности будем считать, что эта база составлена экспертом предметной области. Таким образом, произведенные замены можно считать возможными.

Действительно, пусть дана сигнатура $\zeta =< R, C >$ и некоторое множество бесквантворных предложений этой сигнатуры, где R и C - кортежи предикатов и констант соответственно.

Рассмотрим предикат $p_1(c_1)$, где $p_1$ это одноместный предикат из R, а $c_1$ - константа из C. Пусть этот предикат находится во входном множестве предложений. Предположим, что существует константа $c_n$ из C такая, что в онтологии можно найти цепочку $c_1 c_2 c_3 \dots c_n$ связей вида обобщение. Из этих понятий выберем те, которые являются константами в заданной сигнатуре, т.е. принадлежат C. Заметим, что константы $c_1$ и $c_n$ уже заведомо принадлежат C. Таким образом, исключив лишние понятия мы получим следующую цепочку: $c_1 c_{i^0} \dots c_{i^{k-1}} c_n$. Так как онтология создавалась экспертом предметной области и понятия отражают объекты реального мира, а предикат $p_1$ используется без кванторов, то если истинно утверждение для $c_1$, то оно истинно и для других констант в этой цепочке. Таким образом, мы смогли получить k новых предложений. Заметим, что проведенные рассуждения можно обобщить для случая с предикатами с большим числом констант.

Рассмотрим предикат $p_2(c_n)$, где $p_2$ это одноместный предикат из R, а $c_n$ - константа из C. Пусть отрицание этого предиката находится во входном множестве предложений. Также предположим, что существует константа $c_1$ из C такая, что в онтологии можно найти цепочку $c_1 c_2 c_3 \dots c_n$ связей вида обобщение. Рассуждая аналогично можно прийти к выводу, что мы сможем пополнить входное множество отрицаниями этого предиката с различными константами из этой цепочки.

Случай с использованием связей вида синонимии является тривиальным. Если в онтологии находятся два синонима, которые являются константами в заданной сигнатуре, то происходит замена понятий друг на друга. А истинность полученных предложений прямо следует из синонимии констант.

## Описание используемых технологий

В данной главе описываются технологии и процесс создания программного продукта.

Для программного продукта были выработаны следующие требования:

1. Наличие графического интерфейса;
2. Работа с уже существующей базой понятий;
3. Работа с уже существующей системой логического вывода;
4. Поддержка различных операционных систем;
5. Удобство и простота использования.

Под эти требования подходило большое число языков программирования, однако основной выбор пал на объектно-ориентированные.

Так же были сформулированы не функциональные требования:

1. Быстрота и удобство разработки;
2. Возможность устройства модульной архитектуры;
3. Знания и умения разработки на выбранном языке программирования.

Так как быстрота работы программы не были озвучены в требованиях, то был выбран кросс платформенный язык программирования Java, так как он удовлетворял всем поставленным требованиям.

Существуют различные системы, позволяющее хранить онтологию предметной области, однако было решено использовать два основных представления: специализированный словарь типа WordNet и стандартизованный формат хранения онтологий OWL [11, 12, 13]. WordNet был выбран исходя из сложности его интеграции и степени покрытия терминов русского языка в доступных версиях, а OWL, как общепринятый стандарт хранения онтологий. Для каждого такого представления существует множество словарей для различных предметных областей на многих языках.

Словарь WordNet состоит из сетей для нескольких частей речи: существительных, глаголов, наречий и прилагательных. Узлами сети являются "синсеты" - объединение схожих по смыслу слов: синонимов. "Синсеты" соединены между собой различными семантическими отношениями, такими как: гипероним, гипоним, мероним и другими. Таким образом, для реализации предложенного алгоритма нас интересуют только "синсеты" и связи вида гипероним. В качестве реализации была выбрана база русского языка "Русский WordNet" [14].

Также в системе используется "Тезаурус РуТез". В используемом контексте,

данная онтология относится к тому же классу, что и WordNet. Её объем составляет около 158 тысяч слов и выражений, уложенных в сеть 55 тысяч понятий с более 210 тысяч отношений [15].

В качестве примера использования онтологии в формате OWL, была выбрана открытая биологическая онтология биомедицинских исследований (Ontology for biomedical investigations). Данная онтология - часть большого проекта по созданию серии справочных онтологий в биомедицинской области [16].

Данные онтологии используются в программе в качестве источников знаний о предметных областях. Подробное описание использования онтологий будет описано далее в главе "Алгоритм пополнения", однако кратко его можно описать так: извлеченные константы из множества известных предложений ищутся во всех используемых онтологиях, а далее на основе отношений между найденными понятиями в каждой из онтологий, происходит пополнение новыми предложениями.

Системы, реализующее логический вывод, реализуют один из трех главных алгоритмов: прямой логический вывод, обратный логический вывод и вывод на основе резолюций. Прямой логический вывод обычно используется в системах баз знаний, то есть в системах логики первого прядка без функциональных символов. Суть метода заключается в том, что на каждом этапе алгоритма мы добавляем новые предложения, которые можем вывести из уже известных. Критерий остановки - доказательство необходимого утверждения, либо невозможность вывести новое предложение. Проблема этого метода в его не направленности, а именно вывод множества предложений, которые не связаны с искомым. Обратный логический вывод действует строго наоборот. Он идет от цели и применяет правила логического вывода в обратном направлении, пытаясь найти известные факты, которые поддерживают доказательство. Основное преимущество данного метода перед прямым заключается в том, что алгоритм не строит бесконечное множество предложений. Однако такой подход страдает от проблем повторяющихся состояний и неполноты. Метод резолюции это полная процедура логического вывода для пропозициональной логики на основе опровержений. Суть метода заключается в том, чтобы найти во входном множестве и отрицании доказываемого утверждения противоречие. Если противоречие найдено, то это означает, что доказываемое утверждение выводимо из входного множество, а иначе нет.

Таким образом, в качестве модуля, производящего логический вывод, был выбран автоматический решатель теорем для логики первого порядка Prover9 [17]. Данный

решатель является приемником первого высокопроизводительного и широко распространенного решателя Otter. Оба используют оптимизированный вариант доказательства с помощью правила резолюций. Решатель позволяет задавать входные данные, как любое множество предложений, описываемых в рамках логики первого порядка. Из-за возможности его свободного использования, полноты и проверки доказательств сторонними программами, он был выбран в качестве средства логического вывода.

**Архитектура**

Программа представляет из себя файл с расширением .jar, который можно запустить из консоли с помощью команды «java -jar имя_файла.jar». Структура программы представлена следующими директориями:

- Часть классов библиотеки jwnl;
- Основные классы программы:
    - Пакет, отвечающий за возможность графической и консольной работы;
    - Пакет, представления данных;
    - Пакет классов, реализующих основную логику приложения;
    - Пакет классов утилит.

В программе используются дополнительные библиотеки:

- Apache Commons;
- JWNL - библиотека работы с тезаурусом WordNet;
- OWLAPI - библиотека для работы с OWL онтологиями.

Присутствие перегруженных классов библиотеки JWNL связано с тем, что данная библиотека была рассчитана только на работу с английской версией словаря, из-за чего работа с русской версией была невозможна. Поэтому некоторые классы этой библиотеки были перегружены и переписаны для обеспечения совместимости с русской версией WordNet.

Сама программа сделана, основываясь на архитектурном шаблоне MVC. Так же были применены некоторые паттерны проектирования, такие как Абстрактная Фарбрика, Слушатель и другие.

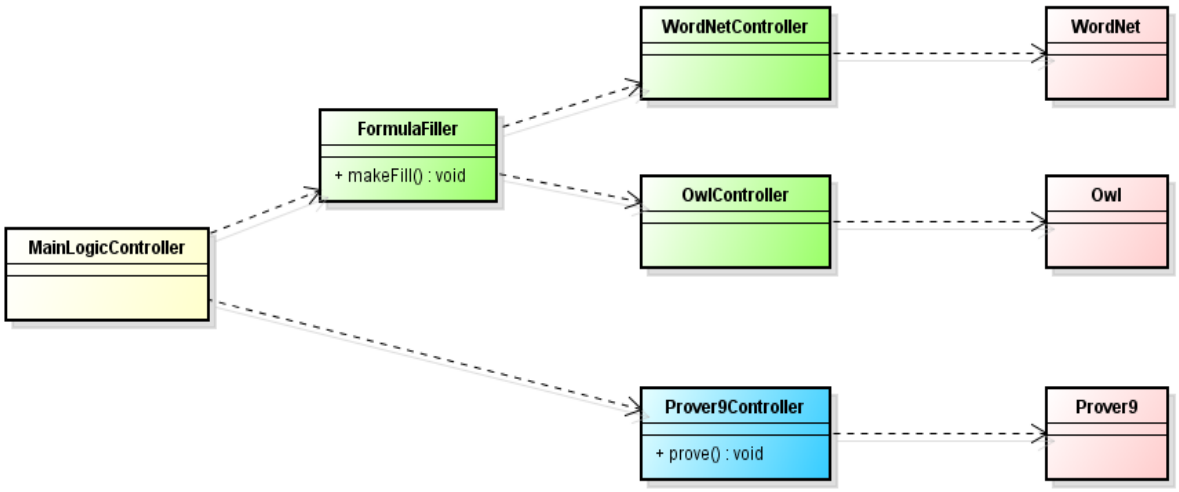


Рис 1. Класс-диаграмма программного продукта

На рисунке 1 представлена краткая класс-диаграмма программного продукта. Желтым цветом отражен главный компонент системы - MainLogicController, который отвечает за ввод-вывод данных и основной ход работы программы. Данный компонент использует FormulaFiller и Prover9Controller для пополнения и логического вывода соответственно. Эти два компонента инкапсулируют свою логику отдельно от основной логики приложения.

FormulaFiller способен использовать возможности пополнения из WordNetController и OwlController. Эти компоненты работают с WordNet и Owl представлениями онтологий соответственно. Настройка каждого компонента может быть осуществлена независимо. В данный момент WordNetController использует два, описанных ранее, тезауруса в своей работе: тезаурус "Русский WordNet" и "Тезаурус РуТез", а OwlController использует "Ontology for biomedical investigations".

Prover9Controller использует Prover9 для осуществления логического вывода.

Таким образом, программу можно разделить на три части: ввод данных, реализация пополнения и осуществление логического вывода. Детали реализации каждого компонента будут рассмотрены далее.

## Описание работы программы

### Ввод данных

При запуске программного продукта открывается главное окно программы, представленное на рисунке 2. На данном этапе происходит ввод множества бесквантерных предложений. Поддерживаются связки И, ИЛИ, ИМПЛИКАЦИИ и отрицания.

Используется стандартизованный формат входных данных для системы автоматического доказательства Prover9 [18]. Ввод предиката осуществляется простым написанием его имени на новой строчке, а в скобках происходит указание на константы, которые включены в него. Например: man{socrat}. В данном случае определен предикат man от константы socrat. Конъюнкция обозначается с помощью связки "&", дизъюнкция - "|", импликация с помощью ">", а для отрицание необходимо добавить "-" перед именем предиката. Также существует поддержка сложных формул, включающих в себя множество предикатов, сгруппированных с помощью специальных символов приоритета - "(" и ")".

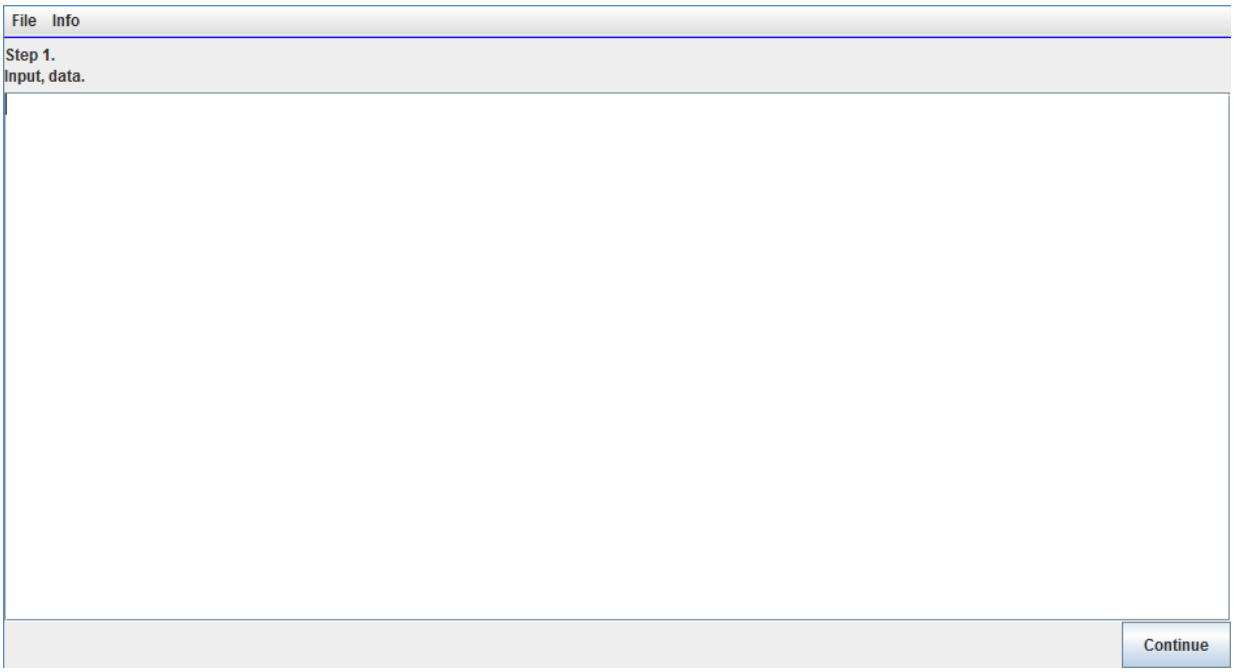


Рис 2. Начальный экран программы.

Ввод данных может быть осуществлен, как вручную в окне программы, так и с помощью загрузки данных из файла.

Все названия предикатов и констант могут быть введены на английском или русском языке. Однако строго рекомендуется использовать какой-либо один язык для работы в программе. Данное ограничение связано с особенностью работы механизмов поиска констант и логического вывода в Prover9. Также стоит отметить, что введенные данные являются регистронезависимыми.

После ввода множества известных бесквантворных предложений и нажатия кнопки "Continue" открывается похожее окно для ввода предложения с неизвестным значением истинности. В виду того, что определение истинности предложения не было основной

задачей работы, в данный момент существует возможность ввода только одного предложения для вычисления.

### Алгоритм пополнения

После нажатия кнопки "Continue" происходит запуск основной логики программы. Опишем её основные этапы.

Первым шагом создается модуль, который отвечает за пополнение множества предложений. Результирующее пополнение будет результатом работы следующего алгоритма.

1. Рабочие данные инициализируем множеством известных предложений;
2. Подаем рабочие данные на вход OwlController для пополнения;
3. Результат пополнения соединяем с рабочими данными;
4. Подаем рабочие данные на вход WordNetController;
5. Результат пополнения соединяем с рабочими данными;
6. Если было осуществлено пополнение, то переходим на шаг 2, иначе возвращаем в качестве результата рабочие данные.

Работа WordNetController происходит следующим образом: он связывается с установленной в системе программой WordNet и инициализирует её. Далее происходит поиск и преобразование всех констант, используемых во входном множестве предложений. Так как в данный момент используется общий словарь WordNet, то из-за особенностей его реализации для одного понятия может существовать несколько синсетов. Таким образом, мы находим все возможные синсеты для каждой константы из сигнатуры нашей модели. Стоит отметить, что если использовать специализированный справочник, составленный экспертом предметной области, то такая проблема перестанет существовать. В нашем же примере, использование нескольких сиснетов для одной константы не скажется на возможностях пополнения. Это связано с тем, что на последующих этапах невозможные синсеты будут попросту отброшены.

После этого между каждыми двумя синсетами ищутся связи вида обобщения. Если такие связи были найдены, то далее мы следуем описанному ранее алгоритму пополнения. Стоит отметить, что в текущей реализации связи вида синонимии не ищутся. Это связано с тем, что из-за особенностей поиска связей вида обобщения, если два понятия находятся в одном синсете, то он будет считаться результатом работы поиска алгоритма обобщения.

Аналогичная работа происходит и в OwlController. Таким образом, после прохождения этого этапа работы алгоритма, к известному множеству предложений будет добавлено некоторое новое заведомо истинное множество сгенерированных предложений. Это множество будет отражено на следующем экране.

### Логический вывод

Входными данными этого этапа является пополненное множество бесквантворных предложений. Данный этап можно разделить на следующие шаги:

1. Преобразование внутреннего формата в формат Prover9;
2. Запуск логического вывода с доказываемым утверждением, а потом с его отрицанием:
    a. Создание входного файла для Prover9;
    b. Запуск отдельного процесса автоматического доказательства из ранее сгенерированных данных;

Запуск двух процессов Prover9 необходим для того, чтобы пользователю можно было отличить случаи успешного доказательства факта против случая противоречивости входных данных. Данные два случая не различаются в виду того, что Prover9 построен на правиле резолюции, которое может найти противоречие не в связке с доказываемым утверждением, а целиком во входных данных.

Создание отдельного процесса для запуска Prover9 обусловлено его архитектурой, как внешнего бинарного файла. Существуют сборки для Windows и Mac. Для пользователей системы *nix предлагается использовать программу wine для запуска Windows программы.

Таким образом, если мы смогли доказать только само утверждение или его отрицание, то мы отдаем это в качестве ответа. А если получилось доказать оба утверждения, то считаем, что входные данные противоречивы.

### Вывод

В данной главе была раскрыта тема реализации программного продукта. Описаны требования, которые ставились перед программой, а так же обоснован текущий инструмент создания программы, который полностью им удовлетворял. Была описана структура программы, её основные модули и компоненты, а также используемые библиотеки.

Была описана главная идея алгоритма, а также используемый алгоритм пополнения множества бесквантворных предложений, применяемый в текущей программе.

# 3 Пример работы и вывода программы

В данном разделе будет продемонстрирована работа программного комплекса на конкретном примере.

Опишем сразу несколько примеров. Пример 1:

Пусть мы знаем, что человек заботится о своей кошке. Забота о любом животном означает, что человек ответственный.

Для дополнительно примера пополнения так же введем данные о том, что в комнате нет животного. Данное предложение служит только для демонстрации возможностей пополнения и никак не повлияет на главный вопрос.

В начале нужно задать известное множество предложений (рис 3.):

1. заботится{человек, кошка}
2. заботится{человек, животное} > ответственный{человек}
3. -в_комнате_находится{животное}

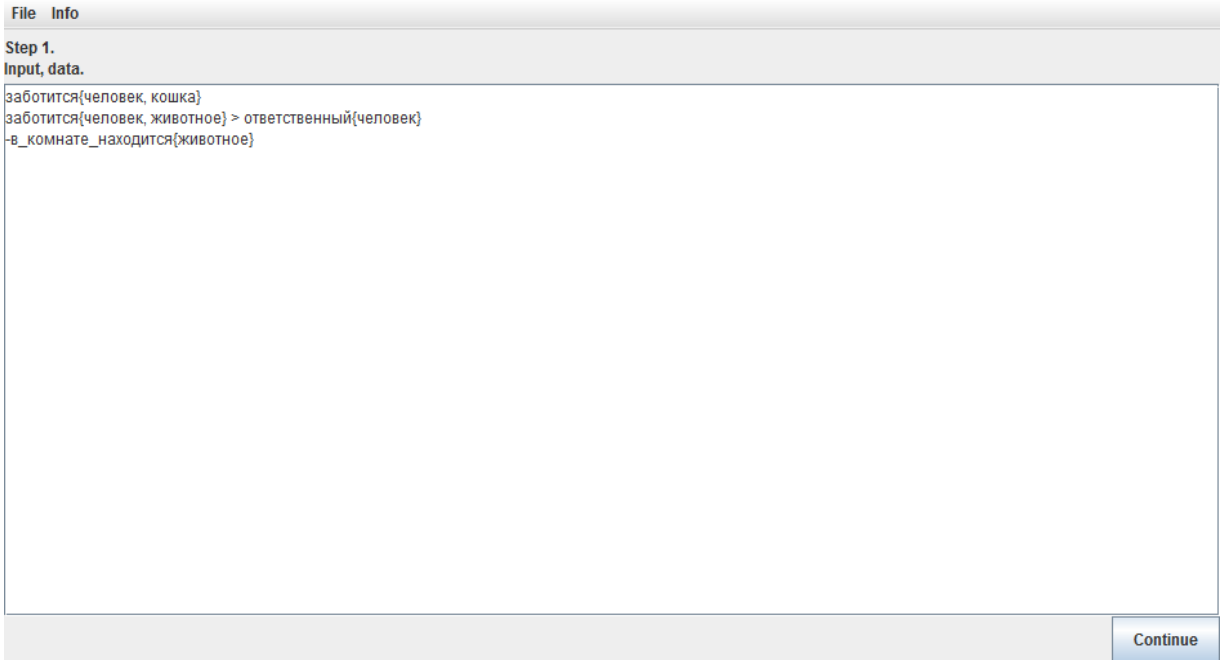


Рис 3. Ввод известного множества предложений.

После нажатия на кнопку "Continue" введенные данные проверяются и преобразуются во внутренний формат. После чего пользователю предлагается задать вопрос системе (рис 4):

1. ответственный{человек}

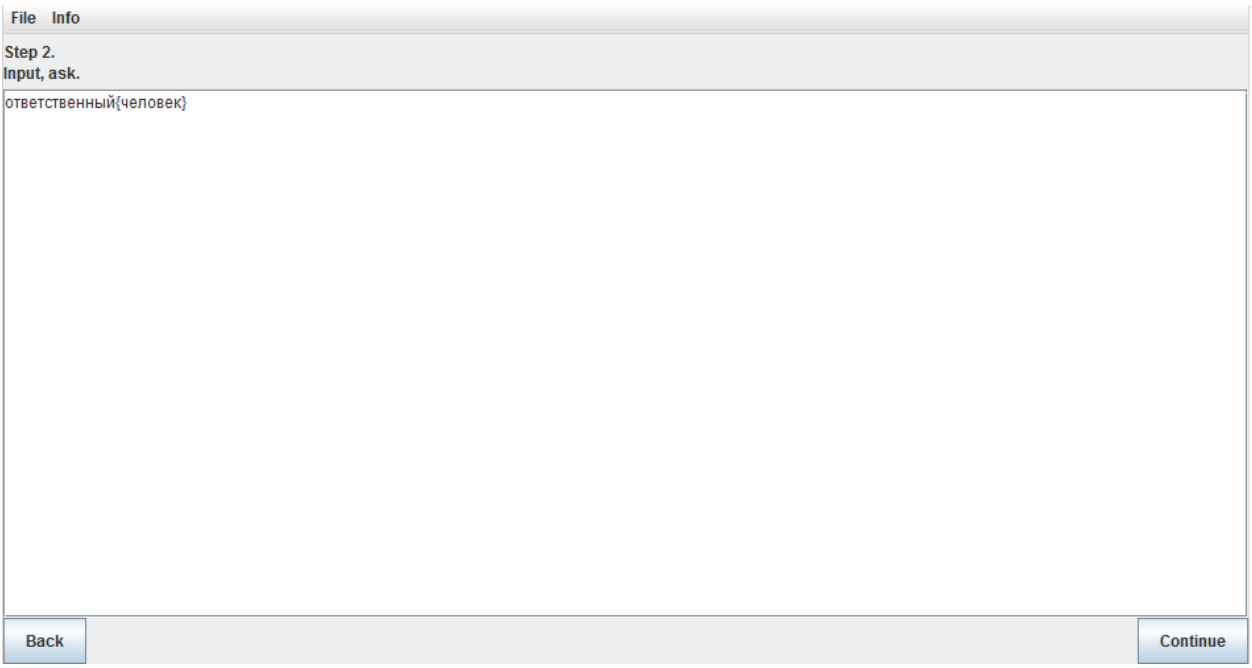


Рис 4. Ввод неизвестного предложения

Для обоих примеров после нажатия кнопки "Continue" в FormulaFiller будет произведен поиск введенных констант во всех используемых онтологиях (напомним, что их в данный момент три). Таким образом, будут найдены следующие последовательности:

Кошка, кот, кошечка, киска, животное_из_семейства_кошачьих, плотоядное_животное, позвоночное_животное, млекопитающее, животное;

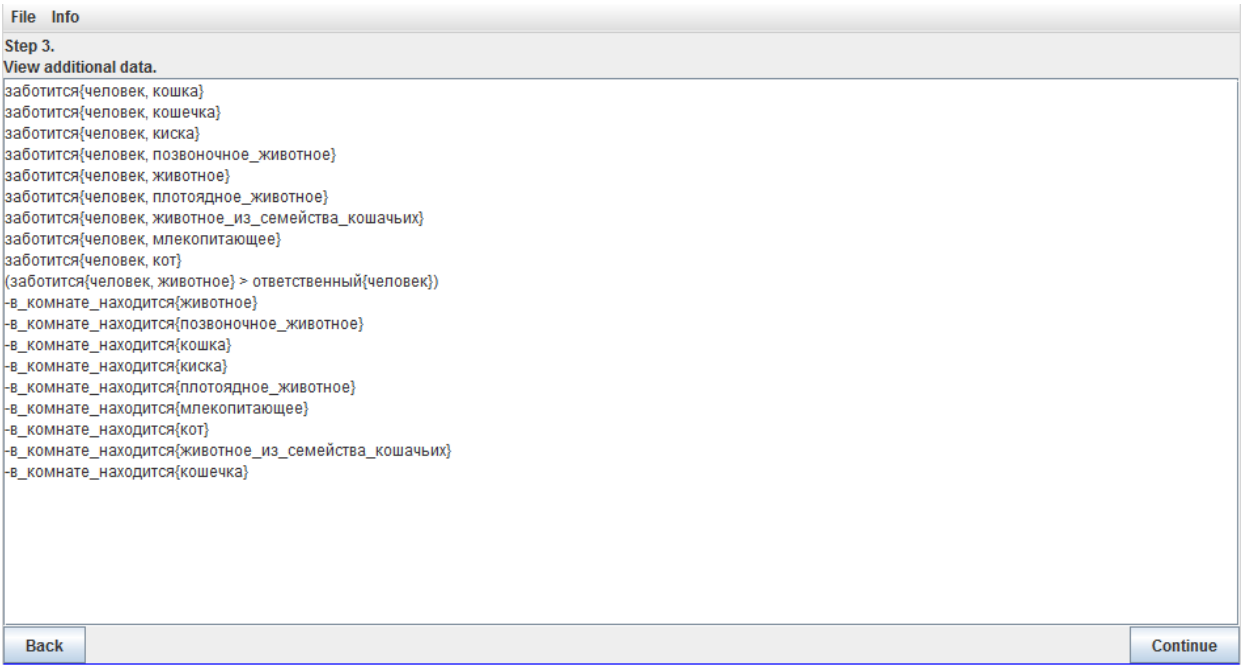


Рис. 5. Результат пополнения

После этого известные предложения будут пополнены. Для этого используется описанный ранее алгоритм. Таким образом, в наше исходное множество добавятся восемь новых истинных предложений и восемь предложений отрицаний. Таким образом, из трех известных предложений их стало девятнадцать. На рисунке 5 показаны все возможные предложения, которые программа смогла добавить в исходное множество предложений. Это множество можно отредактировать и удалить некоторые пополненные предложения.

Далее при нажатии "Continue", полученные после этапа пополнения предложения, будут использованы Prover9 для осуществления логического вывода. Как можно видеть на рисунке 6 ответ истинности - TRUE.

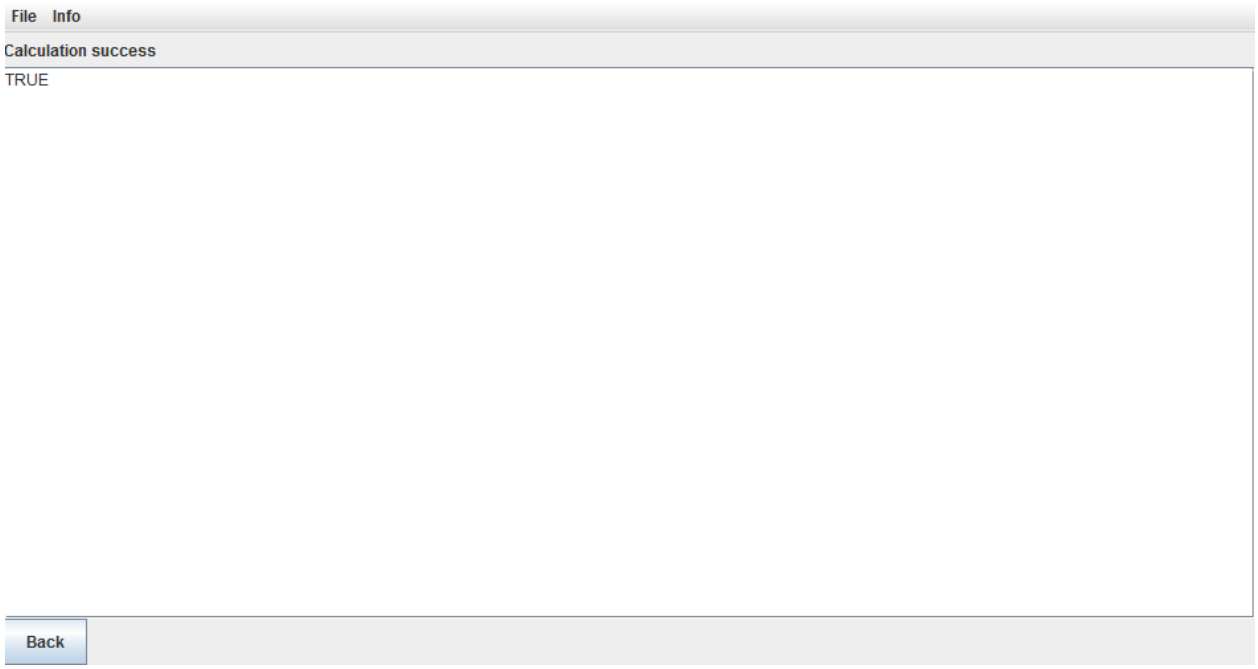


Рис. 5. Итоговый результат

## Вывод

В этой главе был описан один из примеров, показывающих возможность и полезность пополнения множества предложений. В качестве примера была выбрана простейшая задача, которую может решить человек, однако любая система логического вывода решить не сможет. Однако система, которая объединяет в себе знания логической истинности предложений и онтологию предметной области сумела пополнить исходное множество известных данных благодаря чему автоматический решатель теорем сумел

вывести искомое предложение.

Как видно из примера программа нашла множество связей вида обобщения и некоторые связи вида синонимии, произвела пополнение множества предложений новыми, а так же запустила алгоритм логического вывода на полученном множестве и вывела верный ответ. Таким образом, она выполнила поставленную перед ней задачу.

# ЗАКЛЮЧЕНИЕ

В данной работе произведен анализ возможностей автоматического пополнения множества предложений. В качестве объекта исследований выступают множества бесквантерных предложений, а в качестве источника пополнения - онтологии предметных областей.

Были достигнуты следующие результаты:

1. Обоснована возможность пополнения множества бесквантерных предложений на основе данных о предметной области в виде онтологии;
2. Показаны условия и сфера применимости разработанного алгоритма пополнения;
3. Создана программная реализация алгоритма пополнения с использованием онтологий "Русский WordNet", "Тезаурус РуТез" и "Ontology for biomedical investigations";
4. Создана программная реализация для автоматического логического вывода из пополненных данных с использованием Prover9;
5. Создан модуль импорта данных, полученных в результате работы программной системы извлечения знаний из текстов на естественном языке, разрабатываемой Махасоевой Ольгой.

Настоящая работа была представлена на следующих конференциях:

1. 51-я Международная научная студенческая конференция "Студент и научно-технический прогресс". - Новосибирск: НГУ, —2013г.
2. 52-я Международная научная студенческая конференция "Студент и научно-технический прогресс". - Новосибирск: НГУ, —2014г.

Результаты работы были опубликованы в [19] и [20], а также в рецензируемом журнале [21].

Кроме реализации текущего алгоритма были проведены исследования по дальнейшему усложнению программной системы. В будущей работе будут решаться следующие задачи:

1. Использование синонимии в названиях предикатов;
2. Использование контекстного пополнения на основе дополнительной информации из структуры предложений на естественном языке;
3. Оптимизация внутренних алгоритмов;

4. Поддержка пополнения благодаря другим связям из онтологии;
5. Доработка и поддержка кванторов существования и всеобщности;

**Литература**


1. Власов Д.Ю., Пальчунов Д.Е., Степанов П.А. Автоматизация извлечения отношений между понятиями из текстов естественного языка // Вестник НГУ. Серия: Информационные технологии. —2010, Т 8, выпуск 3. - С. 23–33.

2. Пальчунов Д.Е. Поиск и извлечение знаний: порождение новых знаний на основе текстов естественного языка // Философия науки. —2009 №4 (43). - С. 70–90.

3. Пальчунов Д.Е. Яхъяева Г.Э., Хамутская А.А. Программная система управления информационными рисками RiskPanel // Программная инженерия. —2011 №7. - С. 29–36.

4. Яхъяева Г.Э., Ясинская О.В. Применение методологии прецедентных моделей в системе риск-менеджмента, направленного на раннюю диагностику компьютерного нападения // Вестник НГУ. Серия: информационные технологии. —2012, Т 10, выпуск 2. - С. 106–115.

5. Яхъяева Г.Э., Ясинская О.В. Методы согласования знаний по компьютерной безопасности, извлеченных из различных документов // Вестник НГУ. Серия: информационные технологии. —2013, Т 11, выпуск 3. - С. 63–73.

6. Simptomus // [Электронный ресурс]. —2011. Режим доступа: http://simptomus.ru/, свободный. —Загл. с экрана.

7. Логика первого порядка // [Электронный ресурс]. —2014. Режим доступа: https://ru.wikipedia.org/wiki/Логика_первого_порядка, свободный. —Загл. с экрана.

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

9. Prolog documentation // [Электронный ресурс]. —2014. Режим доступа: http://progopedia.ru/language/prolog/, свободный. —Загл. с экрана.

10. Cycorg documentation // [Электронный ресурс]. —2014. Режим доступа: http://www.cyc.com/documentation, свободный. —Загл. с экрана.

11. Лингвистическая онтология "Тузаурус РуТез" // [Электронный ресурс]. —2014. Режим доступа: http://labinform.ru/ruthes/index.htm, свободный. —Загл. с экрана.

12. WordNet // [Электронный ресурс]. —2014. Режим доступа: https://en.wikipedia.org/wiki/WordNet, свободный. —Загл. с экрана.

13. Web Ontology Language // [Электронный ресурс]. —2012. Режим доступа: http://www.w3.org/2001/sw/wiki/OWL, свободный. —Загл. с экрана.

14. Русский WordNet // [Электронный ресурс]. —2011. Режим доступа:

http://wordnet.ru/, свободный. —Загл. с экрана.

15. Лукашевич Н.В. Тезаурусы в задачах информационного поиска. // -М.: Издательство Московского университета, —2011. – 512 с.

16. The Open Biological and Biomedical Ontologies // [Электронный ресурс]. —2014. Режим доступа: http://obofoundry.org/, свободный. —Загл. с экрана.

17. Prover9 // [Электронный ресурс]. —2009. Режим доступа: http://www.cs.unm.edu/~mccune/prover9/, свободный. —Загл. с экрана.

18. Prover9 Manual // [Электронный ресурс]. —2009. Режим доступа: http://www.cs.unm.edu/~mccune/prover9/manual/2009-11A/, свободный. —Загл. с экрана.

19. Гуревич М. И. Автоматизация построения нечеткого значения истинности на неполных данных // 51-я Международная научная студенческая конференция "Студент и научно-технический прогресс". - Новосибирск: НГУ, —2013г. - С. 224.

20. Гуревич М. И. Автоматизация логического вывода из имеющихся знаний на основе онтологий // 52-я Международная научная студенческая конференция "Студент и научно-технический прогресс". - Новосибирск: НГУ, —2014г. - С. 238.

21. Гуревич М. И. Разработка метода автоматизации пополнения множества бесквантерных предложений новыми на основе данных онтологии предметной области // Альманах современной науки и образования. Тамбов: "Грамота", —2014г №7 (85).