Воронежский государственный университет

На правах рукописи

БОЛОТОВА Светлана Юрьевна

Разработка и исследование метода релевантного обратного вывода

специальность 05.13.17 –
теоретические основы информатики

ДИССЕРТАЦИЯ

на соискание ученой степени
кандидата физико-математических наук

Научный руководитель –
доктор физико-математических наук,
доцент С.Д. Махортов

Воронеж – 2013

Оглавление

Введение 4

Глава 1. Основы теории LP-структур 14

1.1. Базовые сведения о бинарных отношениях и решетках 14

1.2. Понятие LP-структуры. Логические отношения 17

1.3. Логическое замыкание и эквивалентные преобразования 19

1.4. Структура логических связей 26

1.5. Логическая редукция 29

1.6. Продукционно-логические уравнения 35

Глава 2. Релевантный обратный вывод 42

2.1. Продукционная система и ее представление LP-структурой 42

2.2. Обратный вывод на основе решения уравнений 46

2.3. Алгоритмы релевантного вывода 51

2.4. Стратегии подсчета релевантности 55

2.5. Использование параллельных вычислений 63

Глава 3. Компьютерная реализация 73

3.1. Общие принципы реализации 74

3.2. Кодирование LP-структур 76

3.3. Архитектура класса ParallelLPStructure 78

3.4. Основная функциональность LP-структуры 80

3.5. LPExpert – новая версия IDE 90

Глава 4. Статистический анализ 96

4.1. Основные теоретические сведения 97

4.2. Исследование алгоритма релевантного LP-вывода 104

4.2.1. Сравнение дисперсий генеральных совокупностей 104

4.2.2. Сравнение средних генеральных совокупностей 105

4.2.3. Исследования в пакете Statistica 6 106

4.3. Исследование кластерно-релевантного LP-вывода 111
4.4. Применение пропорционального подсчета релевантности . 118
4.5. Исследование выполнения параллельных алгоритмов 123
4.5.1. Сравнение дисперсий генеральных совокупностей..... 124
4.5.2. Сравнение средних генеральных совокупностей..... 125
4.5.3. Исследования в пакете Statistica 6..... 126
4.6. Выводы..... 130
Заключение..... 131
Приложение А. Таблицы результатов тестов 133
А.1. Тесты релевантного вывода..... 133
А.2. Тесты кластерно-релевантного вывода 135
А.3. Тесты пропорционального подсчета релевантности 136
А.4. Тесты параллельных алгоритмов 137
Приложение В. Некоторые тексты программ 139
В.1. Модули класса ParallelLPStructure 139
В.2. Подсчет релевантности..... 143
Литература 149

Введение

Информационные технологии на современном этапе развития общества глубоко проникают практически во все сферы деятельности человека, включая материальную, интеллектуальную, социально-культурную и политическую. Подобные тенденции наблюдаются в технике, экономике, медицине, коммерции, образовании, связи, в военной области и так далее. Указанное обстоятельство порождает неуклонно возрастающую зависимость общества от уровня эффективности и надежности информационных систем, обеспечивающих процессы его жизнедеятельности. Информационные системы становятся глобальными, эволюционирующими, социо-критическими, экологически опасными [109]. Соответственно возрастают риски чрезвычайных ситуаций, экологических катастроф, материальных и других потерь национальных масштабов [75–76].

Ответом на подобные вызовы современности служат применяемые в процессах разработки инновационные решения, повышающие эффективность, надежность и безопасность. К таким решениям вполне могут быть отнесены методы интеллектуализации информационных систем [108].

Искусственный интеллект, зародившийся как научная дисциплина еще в 50-х годах прошлого столетия, имеет богатую и противоречивую историю. Так и не заменив интеллект человека, он тем не менее стал неотъемлемой частью современных информационных технологий. *Интеллектуальная система* – это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти такой системы [49]. Подобные системы характеризуются, в частности, способностью «логически мыслить», то есть осуществлять логический вывод (*прямой* или *обратный*) некоторых фактов или решений на основе формальной логики [73].

Однако формальные логики и соответствующие им процессы логического вывода нередко обладают весьма ресурсоемкой архитектурой, как правило,

имеющей экспоненциальный или NP-полный характер [77, 106], как с точки зрения требуемого объема памяти, так и сложности вычислений [7–8]. Именно поэтому, несмотря на бурное развитие компьютерной техники, до сих пор более или менее широкое практическое применение получили лишь такие интеллектуальные системы, которые основаны на упрощенных логиках. К таковым, в частности, относятся продукционные экспертные системы [9].

Интеллектуальные системы продукционного типа представляют важное направление исследований в области искусственного интеллекта. Они используются в теории обучающих систем, систем принятия решений, а также при разработке практических экспертных систем, охватывающих различные прикладные области, такие как медицина, проектирование, геологоразведка и другие [16]. Уже прошедшие пик интереса со стороны исследователей в 80–90 годах, они все же остаются достаточно актуальными, что подтверждается значительным числом публикаций последних лет, посвященных как их теоретическим исследованиям, так и решению прикладных задач [4–6, 24–25, 28, 29, 31, 39, 45, 53, 65, 68, 72, 89].

Роль аппарата продукционно-логических систем как **актуального предмета исследований** повышается и другими факторами. В работе [94] Д.А Поспелов рассматривает системы продукций в широком смысле, как основу для моделирования рассуждений в архитектуре ЭВМ, роботах, экспертных системах и т.д. В последние годы в ряде публикаций [83, 85] был показан и изучен продукционный характер многих различных моделей в информатике, в том числе – непосредственно не относящихся к области искусственного интеллекта. Кроме того, имея относительно несложную структуру, системы продукционного типа могут привлекаться в качестве стартового «полигона» для создания и изучения методов управления формальными знаниями, которые в дальнейшем могут быть применены и для построения более сложных интеллектуальных систем.

С другой стороны, в связи с кардинальным усложнением решаемых задач, повышается ответственность разработчиков информационных систем,

увеличивается цена сделанной ошибки. Как следствие, оказывается весьма востребованным создание строгой математической базы, которая теоретически обосновывала бы корректность и надежность таких систем, а также возможности их автоматической оптимизации.

Эффективным средством формального построения и исследования компьютерных программ, основанных на различных парадигмах, являются алгебраические системы [67, 90, 93]. Это положение в полной мере относится к логическому программированию, особенно в части представления знаний [88]. Алгебраическим методам представления знаний посвящены статьи [38, 43], а также монографии [54, 104].

Математическую основу для создания и исследования моделей знаний предоставляет алгебраическая логика. Ее начала были заложены в работах А.Линденбаума, А.Тарского, систематическое изложение дано в монографиях [20, 95]. Теория Линденбаума-Тарского рассматривает логику нулевого порядка как универсальную алгебру, операции которой соответствуют логическим связкам пропозиционального языка. Примеры алгебраического представления предикатов первого порядка представляют полиадические алгебры Халмоша [19] и цилиндрические алгебры Хенкина-Тарского [22]. Обзор методов алгебраизации кванторов содержится в монографии [36].

Однако общая алгебраическая логика, расширяя возможности исследования логических теорий, не повышает эффективности их практического применения. В силу своей универсальности она не решает важных частных проблем, связанных с упомянутыми выше логическими системами продукционного типа. На это обстоятельство, в частности, указывается в книгах [91, 105]. К задачам такого рода относятся вопросы эквивалентных преобразований, верификации продукционных и подобных им систем, рассматривавшиеся частными методами в работах [1, 14, 26, 30, 37]. Общие обзоры имеющихся подходов к верификации знаний содержатся в [17, 100].

В перечисленных выше работах, посвященных продукционным системам, не было строго обоснованного алгебраического подхода, универсального в рамках широкого спектра систем продукционного типа, который мог бы быть применен для решения задач эквивалентных преобразований, верификации и оптимизации логического вывода.

Возможности алгебраического исследования продукционно-логических систем содержит теория ультраоператоров А.В.Чечкина [107]. В приложении к математической логике она предлагает рассматривать импликации в виде отдельного соответствия. Однако в печати не представлены исследования ультраоператоров, связанные со свойством транзитивности логического вывода.

В работах С.Д. Махортова [80–81] была сформулирована основанная на решетках новая алгебраическая теория (LP-структур). Она представляет общую модель продукционно-логических систем широкого спектра. Данная теория позволяет с единых позиций рассматривать и успешно решать перечисленные выше задачи. Одно из направлений ее применения – оптимизация обратного продукционно-логического вывода и связанная с ним верификация знаний. В частности, в работе [86] высказана принципиально новая идея о минимизации числа обращений к внешним устройствам в процессе обратного вывода и указан способ ее реализации (релевантный LP-вывод). Поскольку время обмена данными с внешними устройствами и, тем более, интерактивным пользователем, на порядки превышает время выполнения команд процессором, отмеченное направление оптимизации способно принести качественно новые результаты.

Данная идея не пересекается с известными методами быстрого вывода в продукционных системах, а дополняет их. Алгоритмы RETE [13] и TREAT [32] разработаны для стратегии *прямого* вывода и использовались в продукционных системах с прямым выводом (например, OPS5 [12], CLIPS [23]). Алгоритм LEAPS [33] компилирует множество правил продукционной системы OPS5 в язык C. В последующих модификациях наиболее популярный алгоритм RETE адаптировался для смешанного

(двунаправленного) вывода [23, 27]. Предложенная в статье [81] и развиваемая в настоящей диссертации концепция использования уравнений в LP-структурах предназначена для оптимизации исключительно *обратного* вывода с точки зрения минимизации обращений к внешним устройствам. Она не требует для своей реализации громоздких структур данных, свойственных указанным выше алгоритмам, в случаях, когда нет потребности в прямом (либо смешанном) выводе. Кроме того, можно адаптировать имеющиеся ускоренные алгоритмы обратного вывода (например, [60, 68]) для решения рассматриваемых в работе продукционно-логических уравнений. Такой подход может оказаться интересным направлением дальнейшего развития теории LP-структур и ее приложений.

Далее отметим, что в упомянутых выше работах С.Д. Махортова релевантный LP-вывод остался практически на уровне общей идеи. В этих статьях, в частности, не были рассмотрены и сопоставлены различные стратегии выбора параметров релевантности. Не было проведено ни достаточного количества тестов, ни, тем более, каких-либо статистических исследований, которые могли бы действительно обосновать и оценить эффективность LP-вывода. Наконец, в представленной ранее его реализации не использовались параллельные вычисления, которые, безусловно, могли еще более повысить быстродействие нового метода.

С учетом изложенного выше можно констатировать, что имеется **актуальная практическая задача** сокращения обмена данными с внешними устройствами и, тем более, интерактивным пользователем. Для ее решения необходимо рассмотреть **научную задачу** разработки и исследования метода релевантного LP-вывода, обеспечивающего ускорение продукционно-логического вывода и верификацию знаний.

Основная цель диссертации – повышение эффективности обратного продукционно-логического вывода и автоматизированных процессов верификации знаний с помощью метода релевантного LP-вывода, основанного на усовершенствованной схеме решения логического уравнения,

предложенных способах выбора параметров релевантности и разработанных параллельных алгоритмах.

Перечисленные выше вопросы относятся к области исследования моделей знаний, методов работы со знаниями, а также определения принципов построения программных средств автоматизации управления знаниями. Данные положения непосредственно отражены в формуле и паспорте научной специальности 05.13.17 – Теоретические основы информатики (пп. 4, 8, 14).

Использованные в работе **методы исследований** основаны на теориях множеств, решеток, бинарных отношений, универсальных алгебр, алгебраической логики, теории графов, математической статистики, параллельных вычислений. При описании приложений применяются методы анализа алгоритмов, теории и технологий программирования.

Объектом исследования являются продукционно-логические системы. **Предмет исследования** – процессы обратного логического вывода в продукционных системах.

Научная новизна диссертации заключается в следующих положениях.

- Сформулирована и исследована обобщенная модель LP-структуры, усовершенствована схема процесса решения продукционно-логического уравнения.
- Разработан новый метод обратного вывода в логических системах продукционного типа, ориентированный на снижение числа обращений к внешним устройствам.
- Предложены и апробированы различные способы выбора параметров релевантности для процессов релевантного и кластерно-релевантного LP-вывода.
- Разработаны параллельные алгоритмы релевантного и кластерно-релевантного LP-вывода, а также параллельные алгоритмы вычисления истинных прообразов в LP-структурах.
- Проведено подробное исследование полученных результатов методами математической статистики. Таким образом формально обоснованы

преимущества нового метода обратного вывода в различных его модификациях.

Работа имеет в основном теоретический характер. Построенные в ней модели, методы и алгоритмы представляют научно обоснованные технологические решения, позволяющие существенно повысить эффективность обратного продукционно-логического вывода и автоматизированных процессов верификации знаний.

Ценность диссертационной работы дополняется возможностью практического применения установленных в ней теоретических результатов.

Практическая значимость работы заключается в программной реализации разработанных параллельных алгоритмов, реализующих LP-вывод, и построенной на их основе новой версии программной системы LPExpert, которая обладает эффективными возможностями создания, эксплуатации и верификации продукционно-логических баз знаний для различных предметных областей.

Достоверность представленных результатов обеспечивается строгими математическими формулировками и доказательствами, а также многочисленными тестами и статистическими исследованиями их результатов.

Результаты диссертации докладывались на следующих научных конференциях и семинарах:

- IV Международной научной конференции «Современные проблемы прикладной математики, теории управления и математического моделирования» (ПМТУММ-2011) (Воронеж, 2011);
- III Всероссийской конференции с международным участием «Знания – Онтологии – Теории» (ЗОНТ-2011) (Новосибирск, 2011);
- X Всероссийской научной конференции «Нейрокомпьютеры и их применение» (НКП-2012) (Москва, 2012);

- Международной молодежной конференции-школе «Современные проблемы прикладной математики и информатики» (MPAMCS’2012) (Дубна, 2012);
- V Международной научной конференции «Современные проблемы прикладной математики, теории управления и математического моделирования» (ПМТУММ-2012) (Воронеж, 2012);
- Международной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2012);
- XI Всероссийской научной конференции «Нейрокомпьютеры и их применение» (НКП-2013) (Москва, 2013);
- International Conference “Distributed Intelligent Systems and Technologies (DIST’2013)” (St. Petersburg, July 1–4, 2013);
- International Conference “Mathematical Modeling and Computational Physics” (MMCP’2013) (Dubna, July 8–12, 2013);
- Всероссийской конференции с международным участием «Знания – Онтологии – Теории» (ЗОНТ-2013) (Новосибирск, 2013);
- научном семинаре «Проблемы современных вычислительных систем» механико-математического факультета МГУ имени М.В. Ломоносова, рук. В.А. Васенин (Москва, 2013);

а также научных сессиях Воронежского госуниверситета (2011–2013).

По теме диссертации опубликовано 14 научных работ, список которых приведен в ее заключительной части. Статьи [1–4] соответствуют Перечню ВАК РФ. Опубликованные работы вполне отражают содержание диссертации. Получено свидетельство о регистрации компьютерной программы [15].

В диссертационной работе изложены результаты, полученные лично автором, включая исследование проблематики, решения задач, алгоритмы и программную реализацию. Из четырех работ, опубликованных совместно с научным руководителем, в диссертацию вошли только результаты автора.

Диссертация состоит из введения, четырех глав, заключения, приложения и списка литературы (для удобства пользования публикации автора перечислены отдельно).

В *первой главе* излагаются базовые положения общей теории LP-структур. Первоначально они были введены и обоснованы в работах С.Д. Махортова. В этой главе они получили уточнение и развитие. Представленные здесь результаты основаны на более слабых по сравнению с работой [80] условиях на LP-структуру. Приведенный здесь материал, включая обозначения, терминологию, формулировки теорем и т.д., необходим для последующего понимания основных результатов диссертационного исследования.

Вторая глава посвящена методологии релевантного и кластерно-релевантного обратного вывода. Она направлена на снижение числа обращений к внешним источникам информации. Предложены, обоснованы и апробированы различные способы выбора параметров релевантности. Дано описание алгоритмов решения логических уравнения в слоях, а также алгоритмов исследования начальных прообразов на предмет истинности. Далее описаны алгоритмы параллельного решения указанных задач. Результаты могут быть также применены для верификации соответствующих баз знаний.

В *третьей главе* описана компьютерная реализация методов релевантного и кластерно-релевантного обратного вывода. Обсуждаются принципы кодирования решеток и LP-структур, используемые для эффективной программной реализации. Представлена архитектура объектно-ориентированного класса ParallelLPStructure, инкапсулирующего свойства и методы описанной в главах 1–2 теории LP-структур, включая нахождение логической редукции и решение продукционно-логических уравнений с параллельным вычислением прообразов.

Описывается созданная автором новая версия интегрированной среды разработки и эксплуатации продукционных систем LPExpert, использующая класс ParallelLPStructure. Основные расширения возможностей LPExpert таковы: различные стратегии подсчета релевантности, параллельный LP-вывод, параллельное исследование прообразов, более гибкая и эффективная структура

представления решеток с возможностью использования 64-разрядной архитектуры компьютера.

Четвертая глава содержит описание проведенных массированных экспериментов, в том числе со случайно генерируемыми базами знаний. Они призваны создать достаточный объем данных для исследования методами математической статистики. Таким образом, формально обосновываются преимущества новой методологии обратного вывода в различных ее модификациях.

В *Заключении* подводятся итоги и обсуждаются некоторые направления дальнейших исследований.

Глава 1. Основы теории LP-структур

Последующие главы диссертации будут посвящены моделированию и исследованию продукционно-логических систем, в части обратного вывода и верификации знаний, с применением специального класса алгебраических LP-структур. Понятие LP-структуры представляет собой абстрактное описание теоретико-множественной модели продукционной системы. Такое обобщение достигается использованием математических решеток в качестве основы алгебраической системы. На решетке задается бинарное отношение, содержащее семантику продукционно-логического вывода.

В данной главе излагаются базовые положения общей теории LP-структур. Первоначально они были введены и обоснованы в работах С.Д. Махортова [80–81]. Теория была названа общей, поскольку дальнейшие исследования в области LP-структур [82, 85, 87] основаны на тех же положениях, продолжая их в том или ином специальном направлении.

В настоящей главе указанные положения получили уточнение и развитие. Представленные здесь результаты основаны на более слабых по сравнению с работами [80–81] условиях на LP-структуру. Основное изменение связано с использованием в формулируемых построениях вместо решеточного отношения частичного порядка $\supseteq$, отношения вида $\supseteq_R$, содержащего лишь необходимое для исследования отношения R подмножество $\supseteq$ в основных определениях, формулировках теорем и доказательствах.

Приведенный здесь материал, включая обозначения, терминологию, теоремы и т.д., необходим для последующего понимания основных результатов диссертационного исследования.

1.1. Базовые сведения о бинарных отношениях и решетках

Введем основные обозначения и определения, необходимые для дальнейшего изложения.

Пусть R – бинарное отношение на некотором множестве F. Для каждой упорядоченной пары $(a,b) \in R$ элемент a будем называть ее левой частью, а элемент b – правой частью. Бинарное отношение R на произвольном множестве F называется:

- рефлексивным, если для любого $a \in F$ справедливо $(a,a) \in R$;
- транзитивным, если для любых $a,b,c \in F$ из $(a,b),(b,c) \in R$ следует $(a,c) \in R$.

Напомним, что для частично упорядоченных множеств различаются понятия *минимального* элемента (для него нет меньшего элемента) и *наименьшего* элемента (он меньше всех) [55].

Как известно, существует замыкание R^* произвольного отношения R относительно свойства транзитивности – *транзитивное замыкание*. *Транзитивная редукция* отношения R означает минимальное отношение R' такое, что его транзитивное замыкание совпадает с транзитивным замыканием R.

В [2] приведен алгоритм построения транзитивной редукции ориентированных графов. Доказано, что эта задача вычислительно эквивалентна построению транзитивного замыкания, а также установлена единственность транзитивной редукции ациклического графа. Как показано в [40], построение наименьшей транзитивной редукции произвольного графа является NP-полной задачей.

Решеткой $\mathbb{F}$ [55] называется множество с частичным порядком $\leq$ («не больше», «содержится»), на котором для любой пары элементов определены операции $\wedge$ («пересечение») и $\vee$ («объединение»). Элемент $c = a \wedge b$ – это точная нижняя грань элементов a,b, то есть наибольший элемент решетки, удовлетворяющий неравенствам $c \leq a$ и $c \leq b$. Соответственно $d = a \vee b$ – точная верхняя грань a,b, то есть наименьший элемент решетки, для которого выполнено $a \leq d$ и $b \leq d$. Таким образом, при любых $a,b \in \mathbb{F}$ справедливо:

- $a \wedge b \leq a$, $a \wedge b \leq b$;

- если $c \in \mathbb{F}$ и $c \le a, c \le b$, то $c \le a \wedge b$;
- $a \le a \vee b$, $b \le a \vee b$;
- если $c \in \mathbb{F}$ и $a \le c, b \le c$, то $a \vee b \le c$.

Пример решетки – множество $\lambda(F)$ всех конечных подмножеств некоторого универсума F.

Решетка $\mathbb{F}$ называется ограниченной, если она содержит общие нижнюю и верхнюю грани, а именно – такие два элемента O, I, что $O \le a \le I$ для любого $a \in \mathbb{F}$. Пример ограниченной решетки – булеан 2^F (множество всех подмножеств) на некотором универсуме F.

Атомом ограниченной (снизу) решетки $\mathbb{F}$ называется минимальный элемент ее подмножества $\mathbb{F} \setminus \{O\}$. Решетка называется атомно порожденной, если каждый ее элемент представим в виде объединения атомов. Например, в булеане атомы – это все подмножества, состоящие ровно из одного элемента универсума. Таким образом, булеан является атомно порожденной решеткой. Для атома a элемента A будем также использовать обозначение $a \in A$ (наряду с $a \le A$).

Решетка $\mathbb{F}$ называется дистрибутивной, если в ней при любых a, b, c выполняются следующие равенства:

- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$;
- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

Можно показать, что каждое из этих тождеств следует из другого [102].

Под дополнением элемента a в ограниченной решетке $\mathbb{F}$ подразумевают элемент $a' \in \mathbb{F}$ такой, что $a \wedge a' = O$ и $a \vee a' = I$. Ограниченная решетка, в которой любой элемент имеет дополнение, называется решеткой с дополнениями. Решетка, в которой каждый замкнутый интервал является решеткой с дополнениями, называется решеткой с относительными дополнениями.

Дистрибутивная решетка с дополнениями называется булевой. В булевой решетке каждый элемент имеет единственное дополнение, причем справедливы тождества (законы де Моргана):

- $(a \wedge b)' = a' \vee b'$;
- $(a \vee b)' = a' \wedge b'$.

Приведем одно полезное утверждение, относящееся к общей теории решеток и отношений. Пусть дана решетка $\mathbb{F}$. Пусть c – некоторое свойство, сформулированное для элементов $\mathbb{F}$. Произвольный элемент $A \in \mathbb{F}$ может обладать или не обладать этим свойством.

Замыканием элемента $A \in \mathbb{F}$ относительно свойства c называется элемент $c(A) \in \mathbb{F}$, являющийся наименьшим в подмножестве элементов решетки, которые содержат A и обладают свойством c.

Замыкание произвольного элемента A в приведенной формулировке существует не для любого свойства c. В качестве отрицательного примера можно привести свойство на булеане «множество содержит не менее n элементов» при мощности A, меньшей n. Если такое замыкание существует, то в силу определения оно должно быть единственным.

Лемма 1.1.1. [80] Пусть $A_1, A_2 \in \mathbb{F}$ и существуют их замыкания $c(A_1), c(A_2)$ относительно некоторого свойства c. Пусть также выполнены условия:

$c(A_2) \supseteq A_1$; $c(A_1) \supseteq A_2$.

Тогда $c(A_1) = c(A_2)$. □

1.2. Понятие LP-структуры. Логические отношения

Под *LP-структурой* [82] подразумевается алгебраическая система, представляющая решетку, на которой задано дополнительное бинарное отношение, обладающее некоторыми продукционно-логическими свойствами. Решетка в контексте данного определения рассматривается в широком смысле, и ее тип может уточняться в конкретных моделях.

Отношение называется *продукционно-логическим*, если оно обладает рефлексивностью, то есть содержит все пары вида (a, a), транзитивностью и другими свойствами, которые также определяются конкретной моделью.

Одно из таких свойств – *дистрибутивность*. Неформально дистрибутивность отношения означает возможность логического вывода по частям и объединения его результатов на основе решеточных операций $\wedge$ и $\vee$. Заданное на абстрактной решетке отношение R называется:

- $\wedge$-дистрибутивным, если из $(a,b_1),(a,b_2) \in R$ следует $(a,b_1 \wedge b_2) \in R$;
- $\vee$-дистрибутивным, если из $(a_1,b),(a_2,b) \in R$ следует $(a_1 \vee a_2,b) \in R$.

Отношение называется *дистрибутивным* при наличии обоих указанных свойств.

В настоящей работе в качестве основы LP-структур рассматриваются решетки с семантикой подмножеств – $\lambda(F)$ (множество конечных подмножеств F) или булеан 2^F (множество всех подмножеств F). Соответственно вместо символов $\leq$, $\geq$, $\wedge$ и $\vee$ используются знаки теоретико-множественных операций $\subseteq$, $\supseteq$, $\bigcap$ и $\bigcup$, а элементы решетки обозначаются, как правило, большими буквами. Под дистрибутивностью отношения подразумевается лишь второе из двух указанных выше свойств. В такой нотации $\bigcup$-дистрибутивность трактуется в следующем смысле: из $(A,B_1),(A,B_2) \in R$ следует $(A,B_1 \bigcup B_2) \in R$. Именно это свойство в контексте диссертации называется *дистрибутивностью* отношения R.

Пусть на решетке $\mathbb{F}$ задано некоторое бинарное отношение R. Совокупность всех атомов решетки, содержащихся в элементах пар отношения R, будем называть множеством атомов, которыми оперирует отношение R. Построенное на объединениях этих атомов подмножество исходной решетки обозначим $\mathbb{F}_R$. Очевидно, $\mathbb{F}_R$ – подрешетка в $\mathbb{F}$.

Для заданного R введем бинарное отношение $\supseteq_R$ – такое подмножество решеточного частичного порядка $\supseteq$, что элементы всех пар $\supseteq_R$ принадлежат $\mathbb{F}_R$. Обозначим также $\supset_R$ подмножество $\supseteq_R$, не содержащее рефлексивных пар (вида (A,A)).

Итак, в рамках настоящей работы действует следующее определение.

Определение 1.2.1. Бинарное отношение R на решетке называется продукционно-логическим (или просто *логическим*), если оно содержит $\supseteq_R$, дистрибутивно и транзитивно.

В отличие от работ [80–81], использующих в аналогичной ситуации отношение $\supseteq$, определение 1.2.1 формулирует *более слабое условие* на определяемое понятие и, следовательно, открывает возможности для получения новых результатов.

Рассматриваемый тип отношений относится к так называемым *монотонным* отношениям. Аналогично [80] можно показать, что при определении логического отношения условие дистрибутивности можно заменить условием монотонности:

$$\text{если } (A,B)\in R, \text{ то } (A, A\bigcup B)\in R \quad (\forall A,B\in\mathbb{F}).$$

Под LP-структурой подразумевается решетка с заданным на ней логическим отношением.

1.3. Логическое замыкание и эквивалентные преобразования

Подобно транзитивным замыканию и редукции отношения на обычном множестве, в теории LP-структур рассматриваются и решаются более сложные задачи нахождения логического замыкания и логической редукции отношений на иерархических множествах – различных видах решеток. *Логическим замыканием* заданного на решетке произвольного бинарного отношения R называется наименьшее продукционно-логическое отношение, содержащее R.

В работе [80] решен вопрос о существовании логического замыкания произвольного отношения в терминах той работы. Здесь рассматривается аналогичная задача в менее строгой постановке (в смысле определения 1.2.1). Из определения не следует существования логического замыкания или редукции для произвольного бинарного отношения. Эти вопросы обсуждаются ниже.

Определение 1.3.1. Пусть задано отношение R на решетке $\mathbb{F}$. Будем говорить, что упорядоченная пара $A,B \in \mathbb{F}$ логически связана отношением R (обозначим этот факт $A \xrightarrow{R} B$), если выполнено одно из следующих условий:

1) $(A,B) \in R$;

2) $A \supseteq_R B$;

3) существуют такие $B_1, B_2 \in \mathbb{F}$, что $B_1 \cup B_2 = B$, причем $A \xrightarrow{R} B_1, A \xrightarrow{R} B_2$;

4) существует элемент $C \in \mathbb{F}$ такой, что $A \xrightarrow{R} C$ и $C \xrightarrow{R} B$.

Определение 1.3.1 по заданному R формулирует еще одно бинарное отношение $\xrightarrow{R}$ на решетке $\mathbb{F}$. Как видно из определения, любая логическая связь $A \xrightarrow{R} B$ образуется на основе некоторого *конечного* подмножества пар отношения R. Важно также отметить, что отношения R и $\xrightarrow{R}$ оперируют общим множеством атомов решетки.

Условия 1)–4) определения 1.3.1 будем также называть правилами вывода в LP-структуре. При выводе логической связи $A \xrightarrow{R} B$ шагом вывода будем называть применение ровно одного правила, возможно, одновременно к некоторому конечному множеству элементов решетки. Например:

- если $A_i \xrightarrow{R} B_i^1$, $A_i \xrightarrow{R} B_i^2$, $i=1,...,n$, то $A_i \xrightarrow{R} B_i^1 \cup B_i^2$, $i=1,...,n$;
- если $A_i \xrightarrow{R} C_i$, $C_i \xrightarrow{R} B_i$, $i=1,...,n$, то $A_i \xrightarrow{R} B_i$, $i=1,...,n$.

Уровнем рекурсии в соотношении $A \xrightarrow{R} B$ будем называть количество шагов вывода, необходимое для получения этой связи. При этом учитываются лишь применения правил 3)–4) определения 1.3.1. Для связи по правилу 1) или 2) уровень рекурсии считается равным нулю.

Поскольку в общем случае связь $A \xrightarrow{R} B$ может быть получена не единственным набором правил, обычно будем лишь оценивать сверху ее уровень рекурсии, не указывая его точного значения.

Применительно к шагам вывода соотношения $A \xrightarrow{R} B$ будем употреблять слова «начальный», «последний», а также «предыдущий»,

«следующий» и т.д. При этом имеется в виду продвижение в направлении прямого логического вывода, то есть от пар исходного отношения ($R \cup \supseteq_R$) к выводимой паре отношения $\xrightarrow{R}$. Заметим, что рекурсивное определение 1.3.1 сформулировано в контексте обратного вывода.

Лемма 1.3.1. Пусть R – логическое отношение на решетке $\mathbb{F}$ и $A,B \in \mathbb{F}$. Тогда, если справедливо $A \xrightarrow{R} B$, то $(A,B) \in R$.

Доказательство. Проведем его с помощью индукции по m – верхней оценке уровня рекурсии в соотношении $A \xrightarrow{R} B$. При $m = 0$ имеет место одно из условий 1)–2) определения 1.2.1. Случай 1) непосредственно означает требуемое утверждение. Если же справедливо 2), то и тогда $(A,B) \in R$, поскольку логическое отношение R по определению содержит такие пары.

Предположим далее, что лемма верна при некотором $m \geq 0$, и докажем ее утверждение при уровне $m+1$. В этом случае новые для рассмотрения варианты могут дать правила 3)–4).

Если имеет место 3), то по предположению индукции уровень рекурсии в соответствующих соотношениях $A \xrightarrow{R} B_1, A \xrightarrow{R} B_2$ не превосходит m, поэтому $(A,B_1) \in R, (A,B_2) \in R$. Тогда, в силу свойства дистрибутивности логического отношения R, получим $(A,B) \in R$. Вариант происхождения связи $A \xrightarrow{R} B$ из условия 4) рассматривается аналогично. □

Теорема 1.3.1. Для произвольного бинарного отношения R на решетке логическое замыкание существует и совпадает с множеством $\xrightarrow{R}$ всех упорядоченных пар, логически связанных отношением R.

Доказательство. Покажем, что при произвольном отношении R соответствующее отношение $\xrightarrow{R}$ является логическим. Действительно, в силу п.2) определения 1.3.1 оно содержит вложения, из п.3) следует его дистрибутивность, а п.4) означает транзитивность данного отношения. Далее, в силу п.1) определения 1.3.1, отношение $\xrightarrow{R}$ содержит R. Для

доказательства теоремы осталось показать, что это – наименьшее из таковых отношений.

Пусть R' – другое логическое отношение, содержащее R. Тогда очевидно, что если $A \xrightarrow{R} B$, то $A \xrightarrow{R'} B$. Отсюда по лемме 1.3.1 имеем $(a,b) \in R'$. Следовательно, отношение $\xrightarrow{R}$ содержится в произвольно выбранном R', и поэтому является наименьшим логическим отношением, содержащим R. □

Продукционно-логические отношения служат математической основой решения прикладных задач, связанных с автоматизацией логического вывода и другими аспектами логического программирования. В связи с этим обстоятельством возникают вопросы автоматического преобразования отношений, при которых логическое замыкание остается без изменения. Такие преобразования могут быть использованы, например, для приведения базы знаний к некоторому каноническому виду, который удобен для исследования и реализации. Полученные выше факты позволяют ввести понятие эквивалентных отношений.

Два отношения R_1, R_2 называются *эквивалентными*, если их логические замыкания совпадают. Для таких отношений используется обозначение $R_1 \sim R_2$. *Эквивалентным преобразованием* данного отношения называется некоторая замена подмножества его пар, приводящая к эквивалентному отношению.

Следствие 1.3.1. Пусть R – отношение на решетке и $A_j \xrightarrow{R} B_j$, $j = 1,...,m$. Тогда отношение $R' = R \bigcup \{(A_j, B_j) \mid j = 1,...,m\}$ эквивалентно R.

Доказательство. Заметим вначале, что отношения R и R' оперируют общим множеством атомов решетки, отсюда $\supseteq_R = \supseteq_{R'}$. Далее, по определению любое логическое отношение является собственным же логическим замыканием. Такой вывод в силу теоремы 1.3.1 относится и к отношению $\xrightarrow{R}$. Далее, из определения 1.3.1 следует, что для любых $R_1 \subseteq R_2$ справедливо $\xrightarrow{R_1} \subseteq \xrightarrow{R_2}$. В нашем случае по построению отношения R'

выполнены включения $R \subseteq R' \subseteq \xrightarrow{R}$. Переходя к логическим замыканиям и учитывая сказанное выше, получаем, что отношения R и R' имеют общее логическое замыкание $\xrightarrow{R}$. □

Теорема 1.3.2. Пусть R_1, R_2, R_3, R_4 – отношения на общей решетке. Если при этом $R_1 \sim R_2$ и $R_3 \sim R_4$, то $R_1 \cup R_3 \sim R_2 \cup R_4$.

Доказательство. Из условия теоремы нетрудно заметить, что отношения $R_1 \cup R_3$ и $R_2 \cup R_4$ оперируют общим множеством атомов решетки ($\mathbb{F}_{R_1 \cup R_3} = \mathbb{F}_{R_2 \cup R_4}$). Требуется лишь доказать равенство $\xrightarrow{R_1 \cup R_3} = \xrightarrow{R_2 \cup R_4}$ как соотношение двух множеств. Пусть $A \xrightarrow{R_1 \cup R_3} B$. Докажем, что при этом справедливо $A \xrightarrow{R_2 \cup R_4} B$. Для этого вновь применим метод индукции по m – верхней оценке уровня рекурсии в соотношении $A \xrightarrow{R_1 \cup R_3} B$.

При $m = 0$ имеет место одно из условий 1)–2) определения 1.3.1. Если справедливо 2), то тогда $A \xrightarrow{R_2 \cup R_4} B$, поскольку логическое отношение $\xrightarrow{R_2 \cup R_4}$ содержит необходимые включения. Если выполнено условие 1), то имеем $(A, B) \in R_1 \cup R_3$. Пусть, для определенности, $(A, B) \in R_1$ (вариант $(A, B) \in R_3$ симметричен). В этом случае имеет место $A \xrightarrow{R_1} B$, откуда в силу условия эквивалентности $R_1 \sim R_2$ получим $A \xrightarrow{R_2} B$. Следовательно, справедливо и $A \xrightarrow{R_2 \cup R_4} B$. Таким образом, при $m = 0$ требуемое утверждение доказано.

Предположим далее, что оно верно для некоторого $m \geq 0$, и докажем его при уровне рекурсии не превосходящем $m+1$. В этом случае новые для рассмотрения варианты для $A \xrightarrow{R_1 \cup R_3} B$ могут дать условия 3)–4) определения 1.3.1.

Если имеет место 3), то по предположению индукции уровень рекурсии в соответствующих соотношениях $A \xrightarrow{R} B_1, A \xrightarrow{R} B_2$ не превосходит m, поэтому $(A, B_1) \in R, (A, B_2) \in R$. Тогда, в силу свойства дистрибутивности логического отношения R, получим $(A, B) \in R$. Аналогично рассматривается случай применения правила 4). □

Следствие 1.3.2. Пусть R_1, R_2, R – отношения на общей решетке. Если при этом $R_1 \sim R_2$, то $R_1 \bigcup R \sim R_2 \bigcup R$.

Следствие 1.3.2 обосновывает принцип локальности эквивалентных преобразований логических отношений: если часть данного отношения заменить эквивалентной частью, то и в целом новое отношение будет эквивалентным исходному. Этот результат открывает возможности автоматических преобразований баз знаний.

Лемма 1.3.2. Пусть R – отношение на решетке $\mathbb{F}$. Пусть также $(A,B) \in R$, причем $B = \bigcup_i B_i$ – конечное объединение элементов $B_i \in \mathbb{F}$ $(1 \le i \le n)$. Тогда отношение R', полученное из R заменой пары (A,B) совокупностью всех пар (A,B_i), эквивалентно R.

Доказательство. Из условия леммы следует, что оба сравниваемых отношения (R и R') оперируют общим множеством атомов решетки, т.е. $\mathbb{F}_R = \mathbb{F}_{R'}$. Введем обозначения $R_1 = \{(A,B)\}$; $R_2 = \{(A,B_1),...,(A,B_n)\}$ и $R^- = R \setminus R_1$. Таким образом, $R = R^- \bigcup R_1; R' = R^- \bigcup R_2$. Рассмотрим логические замыкания $\xrightarrow{R_1}$ и $\xrightarrow{R_2}$ отношений R_1 и R_2 соответственно. Нетрудно также увидеть, что $\supseteq_{R_1} = \supseteq_{R_2}$. Поскольку по определению $\xrightarrow{R_1}$ содержит отношение $\supseteq_{R_1}$, то в силу своей транзитивности оно включает и R_2, то есть $\xrightarrow{R_1} \supseteq R_2$. С другой стороны, отношение $\xrightarrow{R_2}$ из-за дистрибутивности включает R_1. Таким образом, R_1 и R_2 как элементы решетки отношений удовлетворяют лемме 1.1.1. По этой лемме имеем $\xrightarrow{R_1} = \xrightarrow{R_2}$. Последний факт означает, что отношения R_1 и R_2 эквивалентны. Применяя теперь к R_1, R_2, R^- следствие 1.3.2, получим, что $R \sim R'$. □

Теорема 1.3.3. Если в отношении R на решетке $\mathbb{F}$ каждую пару вида (A,B), где $B = \bigcup_i B_i$ – конечное объединение элементов $B_i \in \mathbb{F}$ $(1 \le i \le n)$, заменить совокупностью пар $\{(A,B_1),...,(A,B_n)\}$, то полученное отношение R' будет эквивалентно исходному R.

Доказательство. По-прежнему важно заметить, что оба сравниваемых отношения (R и R') оперируют общим множеством атомов решетки, т.е. $\mathbb{F}_R = \mathbb{F}_{R'}$. Согласно лемме 1.3.2, описанная замена одной пары или конечного их числа приводит к эквивалентному отношению. В этой связи сомнения может вызывать лишь замена в R бесконечного подмножества пар. Для прояснения этого случая рассмотрим логические замыкания $\xrightarrow{R}$ и $\xrightarrow{R'}$ отношений R и R'. Пусть $A \xrightarrow{R} B$. Согласно определению 1.3.1, любая подобная логическая связь формируется на основе некоторого *конечного* числа пар $(A_i, B_i) \in R,\ i = 1,...,n$. Эту совокупность пар можно рассматривать как конечное отношение на решетке $\mathbb{F}$. Обозначив его R_1, имеем $x \xrightarrow{R_1} y$. Проведем для R_1 все описанные в условии теоремы замены. Так как рассматриваются лишь конечные объединения, то получим некоторое конечное отношение R_2. Как замечено в начале доказательства, $R_1 \sim R_2$, отсюда следует $A \xrightarrow{R_2} B$. Поскольку при этом, очевидно, $R' \supseteq R_2$, то имеем $A \xrightarrow{R'} B$. Таким образом, $\xrightarrow{R} \subseteq \xrightarrow{R'}$.

Для доказательства обратного включения ($\xrightarrow{R'} \subseteq \xrightarrow{R}$) рассмотрим упорядоченную пару $A \xrightarrow{R'} B$. По определению 1.3.1 данная логическая связь реализуется посредством некоторого конечного числа пар $(A_j, B_j) \in R',\ j = 1,...,m$. Множество пар $\{(A_j, B_j)\}$ может быть получено применением описанной в условии теоремы процедуры к некоторому конечному $R_1 \subseteq R$, состоящему не более чем из m пар. При этом оно содержится в соответствующем конечном множестве $R_2 \subseteq R'$, полученном из R_1. Остальные рассуждения аналогичны первой части доказательства. □

В качестве применения перечисленных выше результатов рассмотрим важный частный случай эквивалентного преобразования. Отношение R на атомно-порожденной решетке $\mathbb{F}$ называется *каноническим*, если оно задано множеством пар вида (A, a), где $A \in \mathbb{F}$, a – атом в $\mathbb{F}$.

Следствие 1.3.3. Согласно теореме 1.3.3, для любого отношения на атомно-порожденной решетке существует эквивалентное ему каноническое отношение.

1.4. Структура логических связей

В следующем подразделе будет выясняться вопрос о возможности в процессе построения логического замыкания выделить этап, соответствующий транзитивному замыканию. Положительный ответ позволит свести изучение некоторых важных вопросов, касающихся логических отношений, к соответствующим проблемам транзитивных отношений. В частности, построение логического замыкания или редукции можно будет осуществлять с помощью быстрых алгоритмов (типа Уоршолла [2, 46–47]).

С указанной целью исследуем ряд свойств продукционно-логических связей на решетке.

Лемма 1.4.1. Пусть R – отношение на решетке и $A_i \xrightarrow{R} B_i, i = 1,...,n$. Тогда справедлива логическая связь $\bigcup_{i=1,...,n} A_i \xrightarrow{R} \bigcup_{i=1,...,n} B_i$.

Доказательство. Введем обозначения $\tilde{A} = \bigcup_{i=1,...,n} A_i, \tilde{B} = \bigcup_{i=1,...,n} B_i$. Поскольку $\tilde{A} \supseteq A_i, i = 1,...,n$, то в силу $A_i \xrightarrow{R} B_i$ и условия 4) определения 1.3.1 справедливо $\tilde{A} \xrightarrow{R} B_i, i = 1,...,n$. Применяя к последнему соотношению правило 3), получим $\tilde{A} \xrightarrow{R} \tilde{B}$. □

Следствие 1.4.1. На основе леммы 1.4.1 можно ввести обобщенное правило вывода, соответствующее правилу 3) определения 1.3.1: упорядоченная пара $A, B \in \mathbb{F}$ логически связана отношением R ($A \xrightarrow{R} B$), если выполнено следующее условие:

3') существуют такие $A_i, B_i \in \mathbb{F}, i = 1,...,n$, что $\bigcup_{i=1,...,n} A_i = A, \bigcup_{i=1,...,n} B_i = B$, причем $A_i \xrightarrow{R} B_i, i = 1,...,n$.

Очевидно, что правило 3) определения 1.3.1 является частным случаем правила 3').

Лемма 1.4.2. Пусть R – отношение на решетке $\mathbb{F}$. Тогда при выводе любой логической связи $A \xrightarrow{R} B$ все применения правила 3') могут быть произведены без участия правила 2) определения 1.3.1 со строгим вложением $\supset_R$. Роль последнего можно свести к присутствию лишь в качестве компонента для транзитивного правила 4).

Доказательство. Предположим, что при получении вывода $A \xrightarrow{R} B$ использовалось условие 3'). Разобьем имеющееся множество пар $\{(A_i, B_i) \mid i = 1,...,n\}$ на 2 подмножества. В первое из них войдут такие пары $(A_j, B_j), j = 1,...,n_1$, для которых выполнено $A_j \supset_R B_j$. Ко второму подмножеству отнесем все остальные пары $(A_j, B_j), j = 1,...,n_2$. Тогда, вводя обозначения $\bigcup_{j=1,...,n_1} A_j = A^1, \bigcup_{j=1,...,n_1} B_j = B^1, \bigcup_{j=1,...,n_2} A_j = A^2, \bigcup_{j=1,...,n_2} B_j = B^2$, получим $A = A^1 \cup A^2$, $B = B^1 \cup B^2$. При этом $A^1 \supseteq_R B^1$ (соответственно $A^1 \xrightarrow{R} B^1$) и по условию 3') имеет место $A^2 \xrightarrow{R} B^2$. Очевидно, что указанное выше применение правила 3') для $\{(A_i, B_i) \mid i = 1,...,n\}$ можно заменить аналогичным правилом для $(A^1, B^1), (A^2, B^2)$.

Если при этом окажется, что $n_1 = 0$, то для данного вывода $A \xrightarrow{R} B$ сформулированное в лемме утверждение относительно правила 3') выполнено сразу. Если же $n_2 = 0$ либо $A^2 \supseteq_R B^2$, то применение правила 3') не требуется вовсе. Рассмотрим оставшийся нетривиальный случай ($n_1 \neq 0, n_2 \neq 0$ и не выполнено $a^2 \supseteq_R b^2$), в котором поступим следующим образом. Применим правило 3') к соотношениям $B^1 \xrightarrow{R} B^1, A^2 \xrightarrow{R} B^2$, тогда получим $B^1 \cup A^2 \xrightarrow{R} B^1 \cup B^2$. Возвращаясь к имеющемуся вложению $A^1 \supseteq B^1$, заметим, что в силу свойств решетки $\mathbb{F}$ из него следует неравенство

$A^1 \bigcup A^2 \supseteq_R B^1 \bigcup A^2$. Поэтому, применяя к парам $A^1 \bigcup A^2 \supseteq_R B^1 \bigcup A^2$ и $B^1 \bigcup A^2 \xrightarrow{R} B^1 \bigcup B^2$ транзитивное правило 4), приходим к соотношению $A^1 \bigcup A^2 \xrightarrow{R} B^1 \bigcup B^2$, то есть $A \xrightarrow{R} B$. При этом исключено участие в применяемом правиле 3') строгих вложений вида $A_j \supset_R B_j$. Таким образом, утверждение леммы доказано. □

Лемма 1.4.3. Пусть R – бинарное отношение на решетке. Тогда при выводе любой логической связи $A \xrightarrow{R} B$ все применения транзитивного правила 4) могут быть исключены либо перенесены в заключительную стадию данного процесса.

Доказательство. Докажем сформулированное утверждение с помощью индукции по m – оценке уровня рекурсии в соотношении $A \xrightarrow{R} B$. При $m = 0$ для A, B имеет место одно из условий 1)–2) определения 1.3.1. В этом случае вывод $A \xrightarrow{R} B$ не содержит транзитивных связей. Как следствие, утверждение леммы выполнено.

Предположим далее, что оно верно для некоторого $m \geq 0$, и докажем это утверждение при уровне рекурсии $m+1$. По предположению индукции, первые m шагов вывода можно организовать так, что транзитивное правило 4) будет применяться лишь в конце процесса. В этом случае все зависит от того, какое правило вывода применено на последнем $(m+1)$-м шаге – 3') или 4). Если последним применено правило 4), то утверждение леммы сразу оказывается выполненным, так как все транзитивные связи использованы в заключительной стадии вывода. Таким образом, остается исследовать случай, когда на $(m+1)$-м шаге применено правило вывода 3').

Пусть связь $A \xrightarrow{R} B$ в конечном счете получена из условия 3'). Каждое базовое соотношение $A_i \xrightarrow{R} B_i$, $i = 1,...,n$ при этом имеет уровень рекурсии $\leq m$ и, по предположению индукции, все свои транзитивные связи использует лишь в конце вывода. Другими словами, при каждом $i = 1,...,n$ существует цепочка элементов $A_i = C_i^0, C_i^1, ..., C_i^{n_i} = B_i$ такая, что выполнены соотношения $C_i^{k-1} \xrightarrow{R} C_i^k$, $k = 1,..., n_i$, при выводе которых не используется

транзитивное правило 4). По причине конечности n величины n_i ограничены в совокупности, что означает $n_i \le N, i = 1,...,n.$ В связи с этим фактом будем считать все указанные выше цепочки элементов равными по длине N, дополнив каждую из них в случае необходимости справа повторяющимся элементом $C_i^{n_i} = B_i$. Тогда для цепочек $A_i = C_i^0, C_i^1, ..., C_i^N = B_i, i = 1,...,n$ построим элементы $\tilde{C}_k = \bigcup_{i=1,...,n} C_i^k, k = 0,...,N$. Очевидно, что $\tilde{C}_0 = A$ и $\tilde{C}_N = B$. Кроме того, поскольку $C_i^{k-1} \xrightarrow{R} C_i^k, i = 1,...,n$, то к этим парам можно применить правило 3'), то есть получаем $\tilde{C}_{k-1} \xrightarrow{R} \tilde{C}_k$, $k = 1,...,N$.

Таким образом, удалось показать, что в рассматриваемом случае все применения транзитивного правила 4) в базовых соотношениях $A_i \xrightarrow{R} B_i, i = 1,...,n$ можно заменить его применением на заключительной стадии вывода к построенным элементам $\tilde{C}_k, k = 0,...,N$. □

1.5. Логическая редукция

В любой парадигме программирования действует правило хорошего тона – избегать неоправданного дублирования кода или данных. В приложениях логических систем также естественным образом возникает вопрос об их эквивалентной минимизации. В общей математической логике минимальная система аксиом называется базисом. Проблемы существования базисов правил для различных логик рассматривались В.В. Рыбаковым [97–99]. Однако системы продукционного типа имеют особенности, дающие дополнительные возможности минимизации.

LP-структура представляет собой алгебраическую модель продукционной системы. Поэтому очевидное решение задачи минимизации дает такой способ представления бинарного отношения, когда в памяти компьютера хранится лишь уникальная часть информации о данном отношении, а остальная информация может быть получена из общих свойств логических отношений. Под уникальной частью подразумевается подобранное по

определенным критериям отношение $R_0 \subset R$, из которого R получается построением логического замыкания.

Логической редукцией отношения R на решетке $\mathbb{F}$ называется эквивалентное ему минимальное отношение R_0. Минимальность в данном определении понимается в смысле частично упорядоченных множеств. Во-первых, для данного R логическая редукция R_0 может быть не единственной. Во-вторых, после исключения из R_0 хотя бы одной пары, получается отношение, не эквивалентное R.

В работе [80] при более сильном условии на LP-структуру показано, что логическое замыкание данного отношения R является транзитивным замыканием некоторого другого отношения $\tilde{R} \supseteq R$. Данный результат полезен для разработки эффективных алгоритмов построения логического замыкания и редукции. Он позволяет свести исследование вопросов о нахождении логического замыкания и редукции отношений к рассмотрению соответствующих свойств транзитивных отношений. Здесь аналогичный вопрос рассматривается и решается на основе определения 1.2.1, то есть для более общей LP-структуры.

Для произвольного бинарного отношения R на решетке $\mathbb{F}$ рассмотрим отношение $\tilde{R}$, построенное по R последовательным выполнением следующих действий (шагов):

- добавить к R все пары вида (A, A), где $A \in \mathbb{F}_R$ (см. п. 1.2), и обозначить новое отношение R_1;
- добавить к R_1 всевозможные пары (A, B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i, B_i) $(i = 1, ..., n)$ принадлежат R_1;
- объединить полученное отношение с отношением включения $\supset_R$.

Замечание 1.5.1. Из предыдущих теорем следует, что отношение $\underset{\sim}{R}$ эквивалентно R.

Лемма 1.5.1. Пусть R – бинарное отношение на решетке. Тогда, если $A \xrightarrow{R} B$, и это соотношение может быть получено без использования транзитивного правила 4) определения 1.3.1, то $(A,B) \in \tilde{R}$.

Доказательство. Пусть имеет место $A \xrightarrow{R} B$, полученное без правила 4). Если указанное соотношение произошло непосредственно из условия 1) или 2) определения 1.3.1, то сразу имеем $(A,B) \in \tilde{R}$.

Остается рассмотреть случай применения правила 3'). Все необходимые для этого рефлексивные пары могут быть подготовлены в самом начале процесса вывода. При этом в силу леммы 1.4.2 пары вида $A \supset_R B$ не потребуются. Следовательно, правило 3') может лишь завершать этот процесс.

Если сопоставить указанные 2 этапа вывода с последовательностью построения отношения $\tilde{R}$, то выяснится, что вывод $A \xrightarrow{R} B$ соответствует построению некоторого подмножества $\tilde{R}$, что и доказывает включение $(A,B) \in \tilde{R}$. □

Теорема 1.5.1. Для произвольного отношения R на решетке $\mathbb{F}$ логическое замыкание $\xrightarrow{R}$ представляет собой транзитивное замыкание $\tilde{R}^*$ соответствующего отношения $\tilde{R}$.

Доказательство. Как было отмечено выше, отношение $\tilde{R}$ эквивалентно R. Следовательно, по определению логического замыкания, имеем $\tilde{R} \subseteq \xrightarrow{R}$. Отсюда, поскольку отношение $\xrightarrow{R}$ транзитивно, получаем $\tilde{R}^* \subseteq \xrightarrow{R}$.

Докажем обратное включение. Предположим, что $A \xrightarrow{R} B$. По лемме 1.4.3, при получении этого вывода все применения транзитивного правила 4) определения 1.3.1 могут быть перенесены в заключительную стадию процесса. Данное утверждение означает, что существует цепочка элементов $A = C_0, C_1, ..., C_N = B$ такая, что выполнены соотношения $C_{k-1} \xrightarrow{R} C_k$, $k = 1, ..., N$, при выводе которых правило 4) не используется. Тогда по лемме

1.5.1 имеем $(C_{k-1}, C_k) \in \tilde{R}$, откуда сразу получаем $(A,B) \in \tilde{R}^*$. Итак, $\xrightarrow{R} \subseteq \tilde{R}^*$. □

Следствие 1.5.1. Если $R_1 \subseteq R_2$, то $\xrightarrow{R_1} \subseteq \xrightarrow{R_2}$.

Доказательство. Во-первых, из описания процесса построения $\tilde{R}$ легко заметить, что если $R_1 \subseteq R_2$, то $\tilde{R}_1 \subseteq \tilde{R}_2$. Аналогичное вложение имеет место и для транзитивных замыканий этих отношений, что в силу теоремы 1.5.1 означает доказываемый факт. □

Далее для произвольного отношения R рассмотрим отношение $\underset{\sim}{R}$, построенное по данному R последовательным выполнением шагов, обратных построению $\tilde{R}$, а именно:

- исключить из R содержащиеся в нем пары вида $A \supset_R B$ и обозначить новое отношение R_{-1};
- исключить из R_{-1} всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i, B_i) $(i = 1,...,n)$ принадлежат R_{-1} и не совпадают с (A,B);
- исключить из полученного отношения все рефлексивные пары.

Замечание 1.5.2. Отношение $\underset{\sim}{R}$ эквивалентно R.

Заметим, что подобный подход к построению *транзитивной редукции* («удалить все транзитивные пары») был бы ошибочным. Причина в том, что в некоторых ситуациях (наличие в отношении циклов) транзитивная редукция не единственна [2], и одновременное удаление всех имеющихся транзитивных пар может привести к потере связей. Однако, поскольку сама решетка ациклична, удаление всех указанных выше «объединенных» пар (A,B) приводит к одному и тому же результату, независимо от порядка удаления. По этой причине можно говорить об одновременном удалении всех таких пар.

Лемма 1.5.2. Пусть R – бинарное отношение на решетке. Для того чтобы R являлось логической редукцией, необходимо и достаточно, чтобы R не

содержало ни одной такой пары (A,B), что выполнено соотношение $A \xrightarrow{R\setminus\{(A,B)\}} B$.

Доказательство. Пусть отношение R представляет собой логическую редукцию, то есть является минимальным логически эквивалентным себе отношением. Если бы существовала пара $(A,B) \in R$, логически связанная отношением $R\setminus\{(A,B)\}$, то в силу следствия 1.3.1 ее можно было бы исключить из R, получив при этом меньшее эквивалентное отношение. Таким образом, при наличии указанной пары отношение R не может быть логической редукцией.

Для доказательства обратного утверждения предположим, что не существует ни одной пары $(A,B) \in R$, для которой справедливо $A \xrightarrow{R\setminus\{(A,B)\}} B$. Необходимо доказать, что в этом случае R есть логическая редукция. Предположим противное – пусть существует отношение $R_0 \subset R$, эквивалентное R, причем $(A,B) \in R \setminus R_0$. Тогда, поскольку $(A,B) \in R$, в силу эквивалентности рассматриваемых отношений справедливо $A \xrightarrow{R_0} B$. Так как отношение R_0 не содержит пару (A,B), то $R_0 \subseteq R\setminus\{(A,B)\}$, и логическая связь $A \xrightarrow{R_0} B$ противоречит сделанному предположению – таких пар (A,B) в R нет. Полученное противоречие доказывает требуемое утверждение. □

Следующая теорема указывает достаточное условие существования и способ построения логической редукции данного отношения.

Теорема 1.5.2. Пусть для бинарного отношения R, заданного на решетке $\mathbb{F}$, построено соответствующее отношение $\tilde{R}$. Тогда, если для $\tilde{R}$ существует транзитивная редукция R^0, то соответствующее ей отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию исходного отношения R.

Доказательство. Из построения $\tilde{R}$ и $\underset{\sim}{R}$ (замечания 1.5.1–1.5.2) следует, что указанное в теореме отношение $\underset{\sim}{R}^0$ логически эквивалентно R. Осталось

показать, что $\underset{\sim}{R}^0$ является логической редукцией. Для этого достаточно проверить выполнение для $\underset{\sim}{R}^0$ условия леммы 1.5.2.

Пусть (A,B) – произвольная пара отношения $\underset{\sim}{R}^0$. Необходимо показать, что логическая связь $A \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,B)\}} B$ невозможна. Предположим противное, а именно, что эта связь существует. Тогда в силу следствия 1.3.1 отношение $\underset{\sim}{R}^0 \setminus \{(A,B)\}$ эквивалентно $\underset{\sim}{R}^0$. Заметим, что применение правила 1) определения 1.3.1 для вывода $A \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,B)\}} B$ невозможно, поскольку пара (A,B) не содержится в множестве $\underset{\sim}{R}^0 \setminus \{(A,B)\}$.

По лемме 1.4.3 любая логическая связь может быть построена таким образом, что все применения транзитивного правила будут осуществляться лишь в завершающей стадии ее вывода. Этот факт означает, что существует цепочка элементов $A = C_0, C_1, ..., C_N = B$ такая, что выполнены соотношения $C_{k-1} \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,B)\}} C_k$, $k = 1, ..., N$, при выводе каждого из которых правило 4) не применяется. Отсюда по лемме 1.5.1 имеем $(C_{k-1}, C_k) \in \tilde{R}$. Таким образом, при $N > 1$ пара (A,B) оказывается транзитивной в $\tilde{R}$. Следовательно, она не может содержаться в $\underset{\sim}{R}^0$, являющемся подмножеством транзитивной редукции отношения $\tilde{R}$. В результате получено противоречие исходному предположению $(A,B) \in \underset{\sim}{R}^0$.

Остается исследовать случай $N = 1$. В этой ситуации существует вывод логической связи $A \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,B)\}} B$, который может содержать применения правила 3') лишь в заключительной стадии. Полученная таким образом пара (A,B) описывается шагом 2) процесса построения отношения $\tilde{R}$. Соответственно при нахождении $\underset{\sim}{R}^0$ (обратный процесс) она будет исключена. Таким образом, снова приходим к противоречию исходному предположению $(A,B) \in \underset{\sim}{R}^0$.

Таким образом, исследованы потенциально возможные ситуации для предполагаемого логического вывода $A \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,B)\}} B$. В результате

установлено, что в каждом таком случае наличие связи $A \xrightarrow{\underset{\sim}{R}^0 \setminus \{(A,\,B)\}} B$ противоречит факту $(A,B) \in \underset{\sim}{R}^0$. Следовательно, отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию. □

Замечание 1.5.3. Полученные в настоящем разделе результаты содержат существенное усиление по сравнению с аналогичными теоремами работы [80], основанными на использовании в определении логических отношений операции $\supseteq$. Дело в том, что требование существования транзитивной редукции отношения $\tilde{R}$ при определенных условиях может оказаться избыточным для существования логической редукции исходного отношения R. Очевидно, что при конечном множестве R из него всегда можно последовательно удалить «лишние» пары, получив в результате логическую редукцию. Однако, если при этом сама решетка $\mathbb{F}$ бесконечна и имеет соответствующую структуру, то, объединяя R с отношением $\supseteq$, можно получить не имеющее транзитивной редукции отношение $\tilde{R}$. Таким образом, использование отношения $\supseteq_R$ способно исправить ситуацию.

Что касается оценок сложности алгоритмов построения логической редукции, то здесь результаты не оптимистичны. Как показано в [21], задача минимизации функций Хорна является NP-полной. В настоящей работе используется процедура нахождения не наименьшего, а минимального множества правил. При этом применение быстрого алгоритма построения транзитивной редукции [2] дает сложность $O(N^3)$, если удастся эффективно реализовать построение отношений $\tilde{R}$ и $\underset{\sim}{R}$. Однако при этом само число N элементов решетки может оказаться сопоставимым с мощностью булеана 2^n, где n – количество атомов решетки.

1.6. Продукционно-логические уравнения

В данном разделе вводится связанный с LP-структурами класс логических уравнений. Приводятся результаты о разрешимости и количестве решений

этих уравнений, а также излагаются методы их решения. Нахождение решения продукционно-логического уравнения соответствует обратному логическому выводу на решетке. Терминология, концепция уравнений и механизмы их решения лежат в основе методов *полного* релевантного LP-вывода, составляющих основной предмет настоящего диссертационного исследования. Изложенные в данном разделе результаты были ранее получены в работе [81] для более частной LP-структуры. Их доказательства в нашем случае существенно не изменились, поэтому здесь не приводятся. Однако, в отличие от работы [81], внесено изменение в пошаговую схему решения уравнения. В настоящей работе разбиение исходного отношения на слои производится сразу, до упрощения правой части уравнения. В результате из процесса решения исключается избыточный этап, связанный с идентификацией приближенных решений.

Ранее интересные классы логических уравнений рассматривались в работах [58-59, 66, 79, 101], однако исследуемые в них уравнения имеют отличную от систем продукций природу. На нечетких бинарных отношениях основаны реляционные уравнения, рассматривавшиеся в [10] и ряде других работ. Основные трудности исследования здесь порождаются нечеткостью отношений, и процесс решения соответствует всего лишь единственному шагу обратного вывода.

Пусть дано некоторое бинарное отношение R на решетке $\mathbb{F}$ и справедливо $A \xrightarrow{R} B$. Тогда, как известно [74], B называется образом A, а A – прообразом B при отношении $\xrightarrow{R}$. Каждый элемент из $\mathbb{F}$ может иметь много образов и прообразов. Более того, в рассматриваемой ситуации в силу определения логического отношения любой элемент $B_1 \subset B$ будет образом A и каждый $A_1 \supset A$ окажется прообразом B. По этой причине, для исключения неоднозначности, при изучении образов и прообразов логических отношений необходимо уточнение рассматриваемых понятий.

Для данного элемента $B \in \mathbb{F}$ минимальным прообразом при отношении $\xrightarrow{R}$ называется такой элемент $A \in \mathbb{F}$, что $A \xrightarrow{R} B$ и A является

минимальным, то есть не содержит никакого другого $A_1 \in \mathbb{F}$, для которого $A_1 \xrightarrow{R} B$.

В оставшейся части этого раздела рассматриваются лишь атомно-порожденные решетки.

Определение 1.6.1. Атом x решетки $\mathbb{F}$ называется начальным при отношении R, если в R нет ни одной такой пары (A,B), что x содержится в B и не содержится в A. Элемент X решетки $\mathbb{F}$ называется начальным, если все его атомы являются начальными при отношении R. Подмножество $\mathbb{F}_0(R)$ (будем обозначать $\mathbb{F}_0$, если это не вызовет неоднозначностей) решетки $\mathbb{F}$, состоящее из всех начальных элементов $\mathbb{F}$, называется начальным множеством решетки $\mathbb{F}$ (при отношении R).

Очевидно, начальное множество $\mathbb{F}_0$ образует подрешетку в $\mathbb{F}$.

Обозначим R^L – логическое замыкание отношения R и рассмотрим уравнение

$$R^L(X) = B, \qquad (1)$$

где $B \in \mathbb{F}$ – заданный элемент решетки, $X \in \mathbb{F}$ – неизвестный.

Определение 1.6.2. Частным решением X уравнения (1) называется любой минимальный прообраз элемента B, содержащийся в $\mathbb{F}_0$. Приближенным (частным) решением X уравнения (1) называется любой прообраз элемента B, содержащийся в $\mathbb{F}_0$. Общим решением уравнения (1) называется совокупность всех его частных решений $\{X_s\}$, $s \in S$.

По определению точное решение будет и приближенным. Приближенное решение всегда содержит хотя бы одно точное решение [81].

Уравнения вида (1) будем называть продукционно-логическими (или просто логическими) уравнениями на решетках.

В рамках настоящей диссертационной работы рассматриваются в основном практические аспекты решения логических уравнений, влияющие на эффективность обратного логического вывода в реальных компьютерных системах. Поэтому далее в этом разделе предполагается, что R – конечное

каноническое отношение на атомно-порожденной решетке $\mathbb{F}$, не содержащее пар отношения $\supseteq_R$, а правая часть B уравнения (1) представляет собой конечное объединение атомов.

Введем понятие структурного расслоения исходного отношения R на виртуальные слои (частичные отношения) $\{R_t \mid t \in T\}$. Оно упрощает изучение свойств отношения $\xrightarrow{R}$, а также облегчает построение и исследование ряда алгоритмов, связанных с решением соответствующих логических уравнений. Кроме того, концепция слоев лежит в основе распараллеливания процесса нахождения решений.

С указанной целью вначале разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же атомом x_p в качестве правой части. Такое разбиение имеет смысл благодаря тому, что R является каноническим. Обозначим эти подмножества R^p соответственно их элементу $x_p, p \in P$.

Определение 1.6.3. Слоем R_t в отношении R называется подмножество R, образованное упорядоченными парами, взятыми по одной из каждого непустого $R^p, p \in P$. Два слоя, отличающиеся хотя бы одной парой, считаются различными.

Замечание 1.6.1. Каждый слой содержит максимально возможное подмножество пар из R с уникальными правыми частями. Добавление к слою еще одной пары нарушило бы это условие.

Замечание 1.6.2. Любое подмножество пар в R с уникальными правыми частями принадлежит некоторому слою.

Замечание 1.6.3. В общем случае слои имеют непустые пересечения. Объединение всех слоев равно R.

Замечание 1.6.4. Нетрудно заметить, что общее количество слоев N определяется равенством $N = \prod_{p \in P} N_p$, где N_p – мощность подмножества пар отношения R с правой частью x_p.

Будем констатировать, что решение X уравнения (1) (точное или приближенное) порождается в R некоторым слоем R_t, если $X \xrightarrow{R_t} B$.

Замечание 1.6.5. Аналогично [81] можно показать, что любое частное решение уравнения (1) порождается в R некоторым слоем R_t.

Замечание 1.6.6. Для нахождения решения уравнения (1) в некотором слое R_t достаточно вместо (1) решить аналогичное уравнение с отношением R_t.

Последние замечания не гарантируют, что два различных слоя не могут порождать одного и того же решения. Кроме того, могут существовать слои, дающие частное решение для R_t, но приближенное для R. Некоторые слои могут вообще не давать решений. Однако, справедливо утверждение о том, что один слой не может порождать более одного точного решения (см. [81]).

Лемма 1.6.1. Ни один слой R_t не может содержать двух различных частных решений уравнения (1).

Объединяя перечисленные выше результаты, можно сформулировать следующее утверждение.

Теорема 1.6.1. Для нахождения общего решения уравнения (1) достаточно найти частное решение X_t в каждом слое R_t, если оно существует. Далее из полученного множества решений необходимо исключить элементы, содержащие другие элементы этого же множества.

Следствие 1.6.1. Количество частных решений уравнения (1) оценивается сверху выражением $N = \prod_{p \in P} N_p$, где N_p – мощность подмножества пар отношения R с правой частью x_p, а P определяет все такие подмножества (см. замечание 1.6.4).

Теорема 1.6.1 и замечание 1.6.6 позволяют свести вопрос о решении уравнения (1) к задаче нахождения частного решения уравнения

$$R_t^L(X) = B, \qquad (2)$$

где B – неначальный элемент решетки $\mathbb{F}$, R_t – произвольный слой в R.

Согласно лемме 1.6.1, обратное к $\xrightarrow{R_t}$ отношение является однозначным отображением $f_t : \mathbb{F} \to \mathbb{F}$ с некоторой областью определения $D(f_t) \subseteq \mathbb{F}$. Для него выполнены некоторые важные свойства.

Лемма 1.6.2. Отображение f_t является $\bigcup$-гомоморфизмом [55], то есть $f_t(B_1 \bigcup B_2) = f_t(B_1) \bigcup f_t(B_2), \forall B_1, B_2 \in D(f_t)$.

Следствие 1.6.2. Отображение f_t является изотонным [55]. Это означает, что если $B_1 \supseteq B_2$, то $f_t(B_1) \supseteq f_t(B_2)$.

Таким образом, основываясь на лемме 1.6.2, для решения уравнения (2) достаточно решить уравнение с каждым атомом элемента B в качестве правой части. Принимая во внимание это обстоятельство, рассмотрим задачу нахождения частного решения следующего уравнения:

$$R_t^L(X) = b, \tag{3}$$

где b – неначальный атом решетки $\mathbb{F}$, R_t – произвольный слой в R.

Рассмотрим методы решения этой задачи. В [81] для соответствующей LP-структуры показано, что она эквивалентна задаче на ориентированном графе перечисления всех входных вершин, из которых достижима данная вершина.

Построим такой граф G_{R_t}. Каждому атому решетки $\mathbb{F}$, участвующему в отношении R, сопоставим вершину графа. Далее для каждой пары (A, a) рассматриваемого слоя R_t построим дуги, ведущие из всех вершин, соответствующих атомам A, в вершину, соответствующую данной a. Для краткости изложения будем отождествлять атомы $\mathbb{F}$ и соответствующие им вершины в графе.

В полученном графе G_{R_t} выберем вершину b. Рассмотрим подграф $G_{R_t,b} \subseteq G_{R_t}$, содержащий все вершины, из которых достижима вершина b (включая саму b) и все дуги, соединяющие такие вершины.

Следующий факт завершает описание пошагового процесса решения исходного логического уравнения (1).

Теорема 1.6.2. Уравнение (3) имеет не более одного решения. Если граф $G_{R_t,b}$ не содержит циклов, то единственное решение уравнения состоит из всех точек, соответствующих входным вершинам графа. Если $G_{R_t,b}$ содержит хотя бы один цикл, то уравнение решений не имеет.

Итак, в настоящей главе были изложены основные элементы общей теории LP-структур. Эта теория создает эффективную алгебраическую модель для продукционно-логических систем. Последующие главы работы основываются на ее положениях, применяя их к задачам, связанным с разработкой, исследованием и компьютерной реализацией нового метода релевантного LP-вывода.

Следующая глава посвящена методологии LP-вывода. Обоснованы и апробированы различные способы выбора параметров релевантности. Дано описание алгоритмов решения логических уравнений в слоях, а также алгоритмов исследования начальных прообразов на предмет истинности. Описаны алгоритмы параллельного решения указанных задач.

Глава 2. Релевантный обратный вывод

Настоящая глава посвящена моделированию и исследованию процессов обратного вывода в продукционно-логических системах на основе специального класса алгебраических систем – LP-структур. Как уже отмечалось, понятие LP-структуры представляет абстрактное описание теоретико-множественной модели различных классов продукционных систем. В качестве конкретной предметной области здесь рассматривается известный тип продукционных экспертных систем [44]. Он выбран в силу своей простоты и широкой известности, что позволяет непосредственно сосредоточиться на изложении основных идей и методов релевантного вывода. Однако рассматриваемые подходы могут быть применены к системам с более сложной логикой, в том числе неклассическим [82].

2.1. Продукционная система и ее представление LP-структурой

Последующие разделы главы посвящены возможным применениям LP-структур к моделированию логического вывода в системах продукционного типа. С этой целью введем терминологию, связанную с экспертными продукционными системами. Рассматриваемая ниже модель и ее практическая реализация описаны в [44]. Данная книга носит в большей мере учебный характер, однако ее идеи оказались удачными и в дальнейшем использовались и развивались в ряде работ по экспертным системам, в первую очередь – обучающим (например, [15, 48]).

Продукционные экспертные системы манипулируют множествами фактов и правил (продукций). *Факт* представляет собой единицу декларативной информации – некоторое суждение о внешнем мире или состоянии системы. Стандартным представлением факта является триплет вида «объект.атрибут = значение» [11] (например, «термометр.температура = высокая»). В книге

[44] триплет редуцируется к паре «параметр = значение». Это означает, что объект и атрибут интегрируются в единый параметр. В этой связи «термометр.температура» и «термометр.изготовитель» считаются разными параметрами.

В настоящей диссертационной работе принят еще более абстрактный подход [86], когда в единый элемент интегрируется параметр и его значение, при этом факт считается независимым элементом общего множества фактов. Указанное упрощение делается с тем, чтобы больше сосредоточиться на основных идеях применения LP-структур. В дальнейшем можно реализовать и более сложную конструкцию фактов, скорректировав соответствующим образом описание LP-структуры.

Продукционная система содержит так называемую *рабочую память*. Это некоторое подмножество фактов, которые на текущий момент считаются выполненными. Иногда будем называть такое множество *базой данных* продукционной системы.

Правило (*продукция*) состоит из предпосылки и заключения. *Предпосылка* обычно представляет собой выражение над фактами (например, их конъюнкцию или дизъюнкцию). Предпосылка может быть выполненной (истинной) или невыполненной (ложной) при текущем состоянии рабочей памяти. Если предпосылка верна, то правило может быть применено. *Заключение* – это действие, которое можно осуществить, если верна предпосылка (например, добавить к рабочей памяти некоторый новый факт). *Применение правила* состоит в выполнении действия заключения. В контексте настоящей работы рассматриваются приложения, в которых заключение, как и предпосылка, является выражением над фактами, и соответствующее заключению действие интерпретируется как «считать истинным». Таким образом, в данном случае применение правила означает некоторую модификацию рабочей памяти, обычно – запись в нее тех фактов, справедливость которых вытекает из истинности выражения в заключении правила. Совокупность правил называется *базой знаний*.

Прямым выводом в продукционной системе называется процесс циклического применения правил к содержимому рабочей памяти (его исходное состояние задано в начале работы), и соответственно получение в результате новых фактов, которые считаются справедливыми. Прямой вывод может производиться до тех пор, когда получение новых фактов станет невозможным (при текущем содержимом рабочей памяти не окажется ни одной истинной предпосылки правила, заключение которого способно изменить рабочую память). *Обратный вывод* – это противоположный процесс. В нем по некоторому набору результирующих фактов – *гипотезе*, путем анализа правил в направлении от заключения к предпосылке, подтверждается или опровергается справедливость гипотезы при заданном исходном содержимом рабочей памяти. В процессе обратного вывода содержимое рабочей памяти также меняется.

В работе [82] в качестве возможных приложений теории LP-структур рассматриваются несколько видов продукционных систем, различающихся используемой структурой фактов и правил, точнее – выражения для их предпосылок и заключений, но в качестве базовой используется описанная выше структура.

Алгебраический метод исследования продукционной системы сводится к изучению бинарного отношения R на множестве фактов F. В силу особенностей продукционной системы, R является рефлексивным и транзитивным. Действительно, для любого $a \in F$ можно констатировать, что «если верно a, то верно a». Также при наличии в R двух правил $a \to b$ и $b \to c$ при выводе фактически действует и правило $a \to c$. Например, по правилам «**если** температура высокая, **то** заболел» и «**если** заболел, **то** на работу не ходить» можно сформировать правило «**если** температура высокая, **то** на работу не ходить». Соответствующую данной модели алгебраическую систему можно считать вырожденным случаем LP-структуры с пустым базовым отношением частичного порядка.

Однако нельзя назвать исчерпывающим знанием правило «**если** заболел, **то** на работу не ходить». Ближе к истине были бы следующие высказывания:

«**если** заболел, сегодня рабочий день, время утреннее, **то** на работу не ходить, лечиться», «**если** заболел, сегодня рабочий день, время вечернее, **то** лечиться», «**если** заболел, сегодня выходной день, **то** лечиться».

Поэтому возникает более сложная, называемая стандартной, модель базы знаний [80], правила которой в качестве предпосылки и заключения могут содержать наборы элементарных фактов ($\{a\}$, $\{a,b\}$, $\{a,b,c\}$, …). Общий вид продукции в данном случае таков: $\{a_1,...,a_n\} \to \{b_1,...,b_m\}$, где a_i и b_j – атомарные факты. Смысл такого правила состоит в следующем: если верны все a_i, $i = 1,...,n$, то верны и все b_j, $j = 1,...,m$.

В этой модели объектами бинарного отношения $\to$ являются не элементарные факты, а конечные подмножества их исходного множества F. Состоящая из таких объектов математическая структура представляет собой частный случай решетки $\mathbb{F} = \lambda(F)$.

В данной главе с учетом происхождения рассматриваемой модели используются обозначения, принятые в теории множеств, а не в теории решеток. Чтобы не путать с объектами простейшей логики – элементарными фактами, элементы $\mathbb{F}$ будем, как правило, обозначать большими латинскими буквами. Для любых A,B в $\mathbb{F}$ определены (теоретико-множественные) операции пересечения ($A \bigcap B$) и объединения ($A \bigcup B$). Кроме того, в $\mathbb{F}$ задан частичный порядок – отношение включения $\subseteq$.

Таким образом, на множестве $\mathbb{F}$ имеем базовое отношение $\subseteq$, задающее структуру решетки, и дополнительное отношение $\to$, алгебраически отражающее логические связи знаний конкретной предметной области. Ввиду усложнения модели, логические отношения в данной LP-структуре, кроме упомянутых выше свойств рефлексивности и транзитивности, должны также обладать дополнительными свойствами. Свойство рефлексивности является теперь частным случаем свойства тавтологии включения: при $A \supseteq B$ имеет место $A \to B$. Действительно, если справедливо некоторое множество фактов A, то верно и любое его подмножество B. Кроме того, любую совокупность фактов можно выводить по частям: если $A \to B$ и $A \to C$, то

$A \to B \bigcup C$. Это свойство логических отношений называется дистрибутивностью. Из него формально следует монотонность такой логики: поскольку $A \to A$, то из $A \to B$ по свойству дистрибутивности получаем $A \to A \bigcup B$. Это свойство означает монотонное накопление знаний при применении правил.

2.2. Обратный вывод на основе решения уравнений

Рассмотрим возможности, которые предоставляет аппарат логических уравнений (п. 1.4) для организации обратного вывода в продукционной системе.

Как известно [77], при исследовании алгоритмов принято анализировать их вычислительную сложность путем получения оценок относительно мощности множества исходных данных. Однако в случае продукционных систем (и логического вывода вообще) процесс нахождения подобных оценок обычно приводит к неудовлетворительным результатам. Действительно, имея множество N возможных атомарных фактов, можно сформировать до 2^N предпосылок и заключений потенциальных правил, что соответствующим образом скажется на оценках сложности алгоритмов машины вывода. Таким образом, важное значение приобретают альтернативные способы повышения эффективности системы.

В частности, существенную роль играет место хранения начальных фактов. В то время как множество правил обычно может целиком располагаться в основной памяти компьютера, для начальных фактов такое хранение возможно далеко не всегда. В то же время единственная операция чтения с диска или, более того, получения информации от интерактивного пользователя, по времени выполнения способна перекрыть сотни и тысячи операций обращения к основной памяти. Таким образом, возникает идея организации такого процесса продукционно-логического вывода, при

котором потребуется как можно меньше запросов к внешнему источнику информации [86].

При стандартной организации обратного логического вывода в продукционной системе [44] после выбора отправной гипотезы машина вывода осуществляет просмотр содержащих гипотезу заключений правил, переходя к предпосылкам и в свою очередь рекурсивно проверяя их истинность. Когда в данном процессе в качестве очередной гипотезы встречается начальный факт (он не присутствует в правых частях правил), для проверки его истинности необходимо обращение к базе данных или пользователю. При этом не все проверяемые начальные факты в результате оказываются необходимыми для осуществления логического вывода, то есть порождения выбранной гипотезы.

Предлагаемый метод вывода на основе решения уравнений начинается с построения всех минимальных начальных прообразов в LP-структуре для атомов, соответствующих значениям объекта экспертизы. Далее в построенном множестве достаточно найти тот прообраз, который содержит лишь истинные факты, после чего сразу можно сделать заключение о соответствующем значении объекта экспертизы. Наиболее простой путь в этом плане – просматривать прообразы последовательно, задавая пользователю вопросы о соответствующих начальных фактах (или обращаясь к базе данных за этими фактами). Такой способ уже дает определенные преимущества – исследуются лишь *минимальные* прообразы. Также предварительно можно исключить из процесса «противоречивые» прообразы, то есть содержащие одновременно альтернативные значения одного и того же объекта.

Однако существует более эффективный способ – приоритетный просмотр прообразов, содержащих значения наиболее «релевантных» объектов, то есть объектов, соответствующих выбранной стратегии. Таковыми в частности могут считаться объекты, чьи значения присутствуют в максимальном количестве построенных прообразов. Тогда единственный отрицательный ответ пользователя на заданный вопрос исключает из рассмотрения сразу

большое количество прообразов, что соответственно ускоряет исследование. Второй признак релевантности объекта – присутствие его значений в прообразах минимальной мощности. Таким образом, предпочтение отдается тем прообразам, проверка истинности которых потребует меньшего количества вопросов пользователю (или обращений к базе данных). Сочетая указанные два показателя релевантности, можно достичь результатов, по эффективности существенно превышающих возможности обычной машины вывода.

Поясним метод тривиальным примером, не претендующим на глубокий жизненный смысл. Пусть имеется следующая база знаний (из трех правил) для принятия решения о том, взять ли с собой зонтик при выходе на улицу:

1) **если** тучи **и** идти_пешком **и** выходишь_надолго **то** взять_зонтик;

2) **если** прогноз_плохой **и** идти_пешком **и** выходишь_надолго **то** взять_зонтик;

3) **если** идет_дождик **и** идти_пешком **то** взять_зонтик.

Для применения LP-вывода требуется вычислить все начальные прообразы гипотезы *взять_зонтик* (минимальные подмножества не выводимых из базы знаний фактов, которые порождают факт *взять_зонтик*). В рассматриваемом примере это элементарно, поскольку они совпадают с предпосылками правил. Итак, гипотеза *взять_зонтик* имеет три начальных прообраза:

1) {тучи, идти_пешком, выходишь_надолго};

2) {прогноз_плохой, идти_пешком, выходишь_надолго};

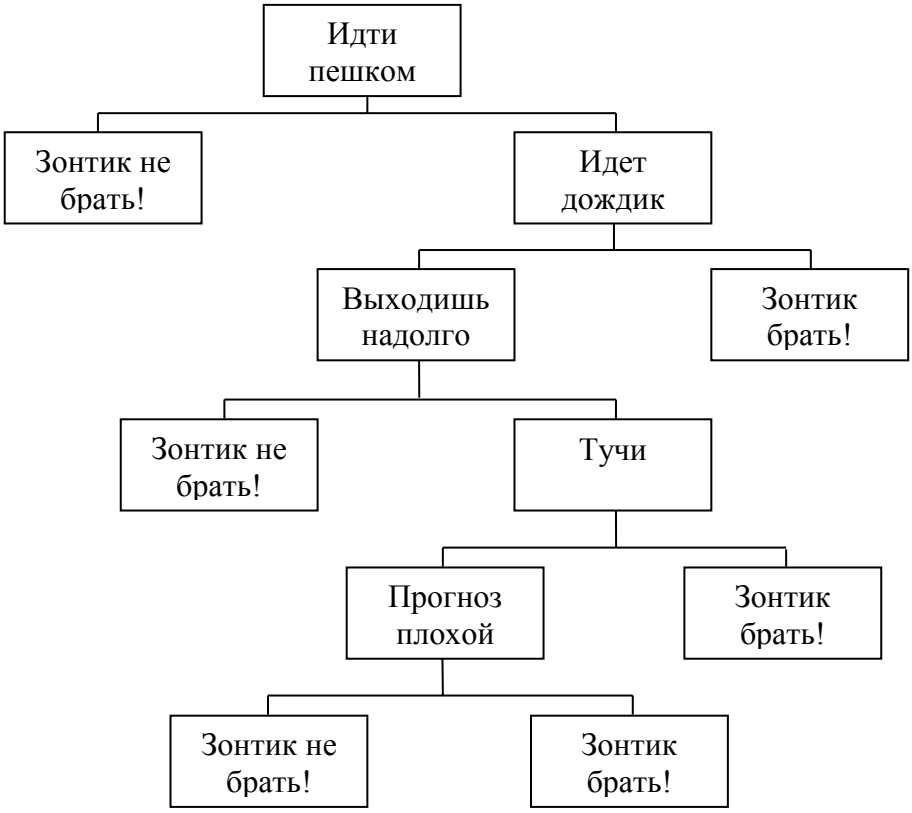
3) {идет_дождик, идти_пешком}.

Далее среди них достаточно найти хотя бы один истинный, то есть состоящий целиком из выполненных фактов, тогда гипотеза окажется верной. Для тестирования каждого факта приходится обращаться к внешнему источнику информации, поэтому число таких проверок необходимо снижать. С этой целью, прежде чем выяснять истинность какого-либо факта, найдем наиболее релевантный из них.

Присвоив всем фактам нулевой приоритет, будем затем повышать его на 1, если факт содержится в максимальном (по сравнению с другими фактами) количестве прообразов. Поскольку факт *идти_пешком* встречается везде, его приоритет станет равным 1. Далее учитываем присутствие фактов в прообразах минимальной мощности (в данном случае – в прообразе №3). В результате повысятся приоритеты фактов *идет_дождик* (до 1) и *идти_пешком* (до 2). Таким образом, релевантным признается факт *идти_пешком*. Именно его истинность целесообразно проверить в первую очередь, обращаясь к внешнему источнику.

При отрицательном ответе на запрос о факте необходимо исключить из рассмотрения все содержащие его прообразы. В нашем случае это сразу решает «проблему зонтика» – все имеющиеся прообразы окажутся ложными («зонтик не брать!»). Если же ответ положителен, вычисляем следующий релевантный факт из еще непроверенных. Им окажется *идет_дождик.* Положительный ответ на запрос о данном факте также решает поставленную задачу – истинным будет прообраз №3 («зонтик брать!»), в противном случае процесс продолжается.

Как показывает рассмотренный пример, следование стратегии релевантности дает шансы решить задачу на ранних запросах, независимо от порядка вычисления прообразов. При этом стандартный обратный вывод соответствовал бы некоторому линейному просмотру прообразов по мере их построения.



Далее отметим, что на основе LP-структур и решения продукционно-логических уравнений может быть также организована верификация знаний. Во-первых, с помощью теоремы о логической редукции продукционная база знаний может быть минимизирована. Таким образом, в ней могут быть выявлены все избыточные правила. Во-вторых, множество правил можно исследовать на противоречивость. Сформулируем семантически противоречивую гипотезу, например, составленную из пары альтернативных значений объекта экспертизы. Решим далее продукционно-логическое уравнение с данной гипотезой в качестве правой части. Если в результате будет найдено решение, состоящее из непротиворечивых (совместимых) фактов, можно сделать вывод о некорректности рассматриваемой базы знаний. Данный процесс можно автоматизировать, запрограммировав процесс перебора пар противоречивых значений.

Заметим также, что пропозициональное исчисление не всегда способно идентифицировать противоречия, определяемые семантикой предметной области. Кроме того, общая математическая логика часто имеет неэффективные реализации.

2.3. Алгоритмы релевантного вывода

Переходя от экспертной системы к терминологии LP-структур, сформулируем соответствующую релевантному выводу следующую задачу нахождения истинного начального прообраза. Даны конечная атомно-порожденная решетка $\mathbb{F}$ (атомы изображают элементарные факты) и на ней бинарное отношение R (представляет совокупность продукционных правил). Не ограничивая общности, можно считать (см. главу 1), что R является каноническим отношением на атомно-порожденной решетке, а правая часть уравнения (1) представляет собой конечное объединение атомов. Тогда, в силу теорем п.1.4, вместо (1) достаточно рассмотреть уравнения с атомами в правой части:

$$R^L(X) = b \qquad (4)$$

Пусть выбран атом $b \in \mathbb{F}$, которому соответствует общее решение уравнения (4) – множество $\{X\}$ всех его минимальных начальных прообразов. На множестве атомов решетки введем частично определенную булеву функцию *True* (функцию «истинности»), которую можно доопределять путем обращения к внешнему источнику информации. В моделируемой продукционной системе интерпретация данной функции такова:

- $True(x) = 1$, если соответствующий факт x содержится в рабочей памяти;
- $True(x) = 0$, если достоверно известно, что x не может содержаться в рабочей памяти;
- $True(x) = null$, если проверка x еще не производилась.

Введем обозначения $T = \bigcup x_k \ (True(x_k) = 1); \ F = \bigcup x_j \ (True(x_j) = 0).$ Необходимо найти такой элемент $X^0 \in \{X\}$, что $X^0 \subseteq T$ (если он существует). В процессе решения задачи требуется также обойтись как можно меньшим количеством доопределений функции *True*.

Последнее требование лежит в основе метода релевантного вывода. Метод состоит из двух стадий: 1) решение уравнения – вычисление множества начальных прообразов и 2) нахождение среди них истинного прообраза. Выполнение указанных стадий может быть организовано в виде конвейера – независимо и параллельно. Работа на каждой из стадий также может быть распределена между параллельными вычислителями.

Для иллюстрации вначале дадим упрощенные описания нерекурсивных алгоритмов обычного обратного вывода и релевантного LP-вывода.

// Обычный обратный вывод

$X^0 = null$

$X = getCurrPreImage\ (b)$

while $X^0 = null$ **and** X != *null* **do**

  is_True = *true*

  foreach $x \subseteq X$ **do**

    if not $x \subseteq T \bigcup F$ **then** is_True = $Ask\ (x)$

    else is_True = $x \subseteq T$

    if not is_True **then break**

  end

  if is_True **then** $X^0 = X$

  else $X = getCurrPreImage\ (b)$

end

Используемая здесь функция $Ask\ (x)$ запрашивает внешний источник об истинности атома x и в соответствии с ответом модифицирует множества T и F (то есть доопределяет функцию $True$). Функция $getCurrPreImage\ (b)$ находит очередной начальный прообраз для гипотезы b. Этот прообраз не обязательно минимален, поскольку не сравнивается с другими существующими начальными прообразами. Заметим также, что при обычном обратном выводе проверка истинности каждого факта может производиться непосредственно перед его добавлением к формируемому прообразу,

поэтому в представленном выше алгоритме не предусмотрены тесты вида $X \subseteq T$.

В следующем алгоритме, напротив, проверке на истинность подвергаются сразу целые прообразы.

// Релевантный LP-вывод

$X^0 = null$

$\{X\} = getPreImages\ (b)$

while $X^0 = null$ **and** $\{X\} \neq \varnothing$ **do**

$k = getRelevantIndex\ (\{X\}, T)$

$Ask\ (x_k)$

foreach $X_j \in \{X\}$ **do**

if $X_j \subseteq T$ **then**

$X^0 = X_j$

break

end

if $X_j \bigcap F \neq \varnothing$ **then** $\{X\} = \{X\} \setminus X_j$

end

end

В последнем алгоритме функция $getPreImages\ (b)$ решает уравнение (4) с правой частью b, то есть строит множество $\{X\}$ всех *минимальных* начальных прообразов для атома b.

Согласно п.1.6, для решения уравнения исходное каноническое отношение R представляется в виде слоев, каждый из которых порождает не более одного решения. Слой содержит максимально возможный набор пар исходного отношения с уникальными правыми частями, два слоя различаются хотя бы одной парой. Непосредственная реализация слоев привела бы к избыточному хранению пар, так как слои могут иметь большие пересечения. Для решения вопроса представления слоев поступим следующим образом.

Разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же атомарным элементом (x_p) в качестве правой части. Обозначим эти подмножества R^p соответственно их элементу $x_p, p \in P$. При реализации канонического отношения для каждого x_p достаточно хранить лишь совокупность левых частей пар с правой частью x_p. Каждому из подмножеств R^p сопоставим итератор – индексную переменную j_p, способную перебирать собственное подмножество, останавливаясь на каждой паре ровно один раз. Таким образом, любой слой R_t в отношении R получается соответствующим набором значений итераторов. Нахождение решения в отдельном слое сводится к получению списка начальных вершин графа, из которых достижима данная вершина (она соответствует атому правой части уравнения).

Эксперименты показали, что при больших объемах БЗ процесс построения *всех* минимальных прообразов может потребовать чрезмерного объема ресурсов компьютера. В связи с данным обстоятельством метод был модифицирован. *Кластерно-релевантный* LP-вывод предполагает последовательное построение кластеров начальных прообразов (подмножеств ограниченной мощности) с их динамическим релевантным исследованием. Модифицированный LP-вывод дает достаточно эффективные (сопоставимые с LP-выводом) результаты для промышленных БЗ больших размеров. Приведем его обобщенный алгоритм.

// Кластерно-релевантный LP-вывод

$X^0 = null$

while $X^0 = null$ **do**

  $\{X\} = getPreImagesCluster\ (b)$

  while $X^0 = null$ **and** $\{X\} \neq \varnothing$ **do**

    $k = getRelevantIndex\ (\{X\}, T)$

    $Ask\ (x_k)$

foreach $X_j \in \{X\}$ **do**

if $X_j \subseteq T$ **then**

$X^0 = X_j$

break

end

if $X_j \bigcap F \neq \varnothing$ **then** $\{X\} = \{X\} \setminus X_j$

end

end

Функция *getPreImagesCluster* (b) выдает фиксированный (ограниченный по количеству элементов или времени вычислений) набор минимальных начальных прообразов атома b, которые на момент вызова функции не пересекаются с множеством F.

Применение этого метода оправдано в случае, если база знаний содержит достаточно большое число фактов и правил (более 200 фактов или 500 правил). В этом случае решение может быть получено на более ранних этапах исследования, чем при обычном обратном выводе. Однако при работе со сравнительно небольшими объемами данных релевантный вывод дает более высокие результаты по сравнению с кластерно-релевантным.

Результаты использования обоих алгоритмов применительно к тестовым базам знаний и анализ их эффективности будут рассмотрены далее.

2.4. Стратегии подсчета релевантности

Использованная в алгоритмах предыдущего раздела функция *getRelevantIndex* ($\{X\}, T$) должна находить индекс k любого из наиболее релевантных и ранее не проверенных на истинность атомов, содержащихся в элементах заданного множества начальных прообразов $\{X\}$. Стратегия релевантности – общее снижение количества вызовов функции $Ask\,(x_k)$. С этой целью функция *getRelevantIndex* может оперировать двумя

упомянутыми в п.2.2 показателями релевантности начальных атомов. К этим показателям относятся следующие параметры:

- присутствие в максимальном количестве построенных прообразов;
- присутствие в прообразах минимальной мощности.

Один из возможных вариантов функции *getRelevantIndex* представлен ниже. Она реализует подсчет суммарной релевантности, который может быть назван *линейным*.

int *getRelevantIndex* ($\{X\}$, T)

// Инициализация приоритетов начальных атомов решетки

foreach $x_k \in \mathbb{F}_0$ **do** Priority [k] = 0

// Нахождение минимальной мощности прообразов

int nMin = *null*

foreach $X_j \in \{X\}$ **do**

if nMin = *null* **or** *Length* (X_j) < nMin **then** nMin = *Length* (X_j)

end

// Вычисление приоритетов начальных атомов

foreach $x_k \in \mathbb{F}_0$ **do**

if not $x_k \subseteq T \bigcup F$ **then** // Рассматриваем лишь непроверенные атомы

foreach $X_j \in \{X\}$ **do**

if $x_k \subseteq X_j$ **then**

Priority[k]++

if *Length* (X_j) = nMin **then** Priority[k]++

end

end

end

end

// Нахождение индекса наиболее релевантного атома

int nRel = *null*

foreach $x_k \in \mathbb{F}_0$ **do**

```
    if nRel = null or Priority[k] > Priority[nRel] then nRel = k
  end
  return  nRel
end
```

Рассмотрим альтернативные способы вычисления релевантности объектов. В частности, возьмем за основу пропорциональный подсчет. Согласно данному подходу показатель релевантности объекта увеличивается не линейно, а обратно пропорционально мощности прообраза, в котором он содержится. Объект, находящийся в прообразе с меньшей мощностью, получит больший коэффициент релевантности, чем объект, находящийся в прообразе с большей мощностью. При этом, если значения объекта присутствуют в максимальном количестве прообразов, релевантность таких объектов еще увеличивается на 1.

Очевидно, что такой способ вычисления может дать неплохие результаты в случае, когда объекты, присутствующие в прообразе минимальной мощности, не присутствуют в других прообразах. Такая ситуация может сложиться при работе с базами знаний с существенно различающейся глубиной правил или с базами знаний, правила в которых имеют глубокую структуру вложенности. В этом случае более эффективным может быть исследование в первую очередь объекта из минимального прообраза.

Рассмотрим пример четырех прообразов для гипотезы «взять зонтик».

{тучи, идти_пешком, птицы_летают_низко, давление_падает};

{влажность_повышенная, душно};

{прогноз_плохой, идти_пешком, выходишь_надолго};

{идет_дождик}.

В случае простого (см. начало настоящего раздела) подсчета релевантности объектов «идет_дождик» и «идти пешком» результаты совпадают и равны 1. В этом случае порядок определения значений указанных объектов однозначно не определен и может быть произвольным. Однако видно, что в случае положительного ответа на вопрос о том, идет ли дождик, можно найти решение за 1 шаг. Пропорциональный подсчет

релевантности позволяет повысить рейтинг объекта «идет_дождик» сразу на 4, в то время как рейтинг объекта «идти_пешком» будет равен 3 (+1, так как объект присутствует в максимальном числе прообразов и +2 за мощность прообраза).

Соответствующая пропорциональному подсчету модифицированная функция *getRelevantIndex* представлена ниже.

```
int getRelevantIndex ({X}, T)
  // Инициализация приоритетов начальных атомов решетки
  foreach x_k ∈ 𝔽_0 do Priority[k] = 0
  int Rating = 1 // Стартовое значение рейтинга
  while {X} ≠ ∅ do
    // Нахождение максимальной мощности прообразов
    int nMax = null
    foreach X_j ∈ {X} do
      if nMax = null or Length (X_j) > nMax then
        nMax = Length (X_j)
      end
    end
    // Вычисление приоритетов начальных атомов
    foreach x_k ∈ 𝔽_0 do
      if not x_k ⊆ T ⋃ F then // Рассматриваем непроверенные факты
        foreach X_j ∈ {X} do
          if x_k ∈ X_j then
            Priority[k]++
            if Length (X_j) = nMax then Priority[k] += Rating
          end
        end
      end
    end
```

foreach $X_j \in \{X\}$ **do** // Исключить рассмотренные прообразы

if $Length(X_j)$ = nMax **then** $\{X\} = \{X\} \setminus X_j$

end

Rating++ // При уменьшении мощности прообразов увеличиваем рейтинг

end

// Нахождение индекса наиболее релевантного факта

int nRel = *null*

foreach $x_k \in \mathbb{F}_0$ **do**

if nRel = *null* **or** Priority[k] > Priority[nRel] **then** nRel = k

end

return nRel

end

Для некоторых баз знаний специальной структуры, как было подчеркнуто выше, этот метод может дать неплохой результат. Эксперименты показывают, что число внешних запросов снижается в среднем на 25%. Однако при тестировании метода на базах знаний с правилами случайного вида он несколько уступает алгоритму, основанному на линейном подсчете релевантности.

Рассмотрим еще один способ вычисления релевантности объектов – «бимаксимальный». В его основе лежит тот очевидный факт, что рассмотрение прообраза мощностью на единицу большей, чем минимальная, может тоже дать положительный результат. Другой фактор релевантности («значения объекта присутствуют в максимальном количестве прообразов») также следует учитывать.

Этот способ похож на описанный выше в п.2.3, с той лишь разницей, что предпочтение отдается минимальному прообразу и прообразу с мощностью на 1 большей минимальной. Этот подход рассчитан на ситуации, когда решение получается на ранних этапах, и истинным является прообраз с минимальной мощностью или же близкой к ней. Данный метод эффективен,

если база знаний содержит большое число правил с глубокой степенью вложенности.

Следуя указанному принципу, можно модифицировать функцию нахождения индекса наиболее релевантного объекта следующим образом.

int *getRelevantIndex* ($\{X\}$, T)

// Инициализация приоритетов начальных атомов решетки

foreach $x_k \in \mathbb{F}_0$ **do** Priority [k] = 0

// Нахождение минимальной мощности прообразов

int nMin = *null*

foreach $X_j \in \{X\}$ **do**

if nMin = *null* **or** *Length* (X_j) < nMin **then** nMin = *Length* (X_j)

end

// Нахождение второй по минимальности мощности прообразов

int nMin2 = *null*

foreach $X_j \in \{X\}$ **do**

if nMin2 = *null* **or** *Length* (X_j) < nMin2 **and** nMin2 ≠ nMin **then**

nMin2 = *Length* (X_j)

end

end

// Вычисление приоритетов начальных атомов

foreach $x_k \in \mathbb{F}_0$ **do**

if not $x_k \subseteq T \bigcup F$ **then** // Рассматриваем лишь непроверенные атомы

foreach $X_j \in \{X\}$ **do**

if $x_k \subseteq X_j$ **then**

Priority[k]++

if *Length* (X_j) = nMin **then** Priority[k]++

if *Length* (X_j) = nMin2 **then** Priority[k]++

end

```
      end
    end
  end
  //  Нахождение индекса наиболее релевантного атома
  int nRel = null
  foreach x_k ∈ 𝔽_0 do
    if nRel = null or Priority[k] > Priority[nRel] then nRel = k
  end
  return  nRel
end
```

Согласно экспериментам, этот способ, как и предыдущий, позволяет уменьшить количество внешних запросов в среднем на 25%.

Кроме представленных выше модифицированных алгоритмов подсчета релевантности, рассмотрим варианты, учитывающие только один из двух факторов релевантности.

Экспериментально установлено, что при небольших базах знаний, правила которых имеют неглубокую структуру вложенности, исключение из рассмотрения показателя релевантности, связанного с минимальной мощностью прообраза, не приводит к снижению эффективности. Это обстоятельства связано с тем, что получаемое количество прообразов оказывается невелико, а их мощности достаточно близки друг к другу.

Примером таких прообразов могут случить следующие множества фактов:

{тучи, идти_пешком, птицы_летают_низко};

{влажность_повышенная, душно, тучи};

{прогноз_плохой, идти_пешком, выходишь_надолго};

{идет_дождик, давление_падает, тучи}.

Однако на практике существование небольших баз знаний с неглубокой вложенностью правил маловероятно. Объясняется это тем, что для получения достаточно достоверного результата необходимо учитывать влияние многих факторов, и правила в базе знаний должны отражать эти факторы. Кроме

того, существует некоторая вероятность, что при большом числе правил одни и те же объекты могут находиться во многих прообразах. Поэтому в случае положительного ответа на запрос понадобится исследование большого числа релевантных объектов. В то же время использование дополнительного показателя релевантности позволило бы отдать предпочтение исследованию прообраза небольшой мощности.

Аналогично при небольших базах знаний с небольшой степенью вложенности правил исключение из рассмотрения показателя релевантности, связанного с наличием объекта в максимальном числе прообразов, не снижает эффективности алгоритма. Этот результат связан с большим разнообразием объектов в прообразах и невозможностью выделить объект, существенно чаще встречающийся в максимальном числе прообразов.

Примером могут служить следующие прообразы:

{тучи, прогноз_неблагоприятный};

{влажность_повышенная, душно};

{выходишь_надолго};

{идет_дождик}.

Однако при работе с большими базами знаний с глубокой вложенностью правил есть высокая вероятность получения схожих по составу прообразов. При этом последний метод не позволяет на ранних этапах исследования истинности отбросить много прообразов в случае отрицательного ответа пользователя на вопрос об объекте, который встречается во многих прообразах. Следовательно, такой способ также не позволяет получить прирост эффективности при работе с большими и сложными базами знаний.

Таким образом, теоретически обосновано и экспериментально установлено, что только сочетание двух показателей релевантности дает наибольшее повышение эффективности вывода, существенно снижая число внешних запросов.

2.5. Использование параллельных вычислений

При больших объемах баз знаний и их достаточно «глубокой» структуре необходимо использовать все имеющиеся в наличии возможности повышения эффективности логического вывода. Одна из подобных возможностей – применение параллельных вычислений. Подобным методам для продукционных систем было посвящено немало работ. К наиболее известным из них относится фундаментальный труд [16]. Параллельным методам в продукционных системах посвящены также работы [35, 41, 53, 57]. Эти методы могут быть использованы на более абстрактном уровне при компьютерной реализации LP-структур.

Однако настоящий раздел диссертационного исследования посвящен вопросам распараллеливания алгоритмов LP-вывода, стратегия которого направлена на снижение числа обращений к внешним устройствам. В этом состоит основная особенность настоящей работы по отношению к другим параллельным реализациям продукционных систем, в которых минимизация по указанному параметру не применяется.

Метод релевантного LP-вывода был модифицирован с использованием параллельных вычислений. Параллельный релевантный LP-вывод предполагает одновременное построение набора начальных прообразов с их дальнейшим исследованием в разных потоках. Здесь и далее под «потоками» подразумеваются threads – потоки выполнения [96].

Как уже было отмечено в п.2.3, для решения уравнения исходное каноническое отношение R представляется в виде слоев. Работа с различными слоями может быть организована независимо и параллельно. Ниже приводятся основные положения такой организации.

Первичный поток приложения создает новые потоки (они ограничены параметром $MaxThreads$ – максимальное количество потоков), передавая им пакет данных. Созданный поток находит решение продукционно-логического уравнения в отдельном слое. Как только вычисляется достаточное число

прообразов, которое ограничено соответствующим параметром (или все прообразы) модуль LP-вывода начинает (релевантно) их исследовать на предмет истинности, обращаясь при необходимости за фактами к внешнему источнику. Если при обработке очередного кластера прообразов истинный прообраз выявить не удается, то процесс вычисления прообразов продолжается. При этом запоминается информация об установленных на предыдущем шаге ложных фактах. На ее основе перед очередным процессом вычисления прообразов сужается множество актуальных правил. Это обстоятельство также существенно ускоряет работу. По окончании работы потоки завершаются.

В случае, когда рабочих потоков бывает слишком много, в соответствии с законом Амдала, эффективность параллельных вычислений начинает снижаться [51]. Частое создание и завершение потоков с малым временем работы, а также большое количество переключений их контекстов, увеличивают объем ресурсов. Поэтому при реализации параллельного LP-вывода используется заранее создаваемый пул потоков [96]. Максимальный размер пула ограничивается параметром. Его значение должно быть большим, чем число процессоров, чтобы добиться максимальной степени одновременности.

Главный поток выбирает поток из пула и передает ему необходимые данные для обработки. Когда количество активных потоков достигает максимума, запрос помещается в очередь. Если число активных потоков меньше максимального, создается новый поток, который получает пакет данных на обработку. Если же количество активных потоков равно максимальному, пакет ставится в очередь и ждет освобождения одного из потоков. В качестве механизма синхронизации потоков могут быть использованы критические секции, которые инициируются в процессе активизации.

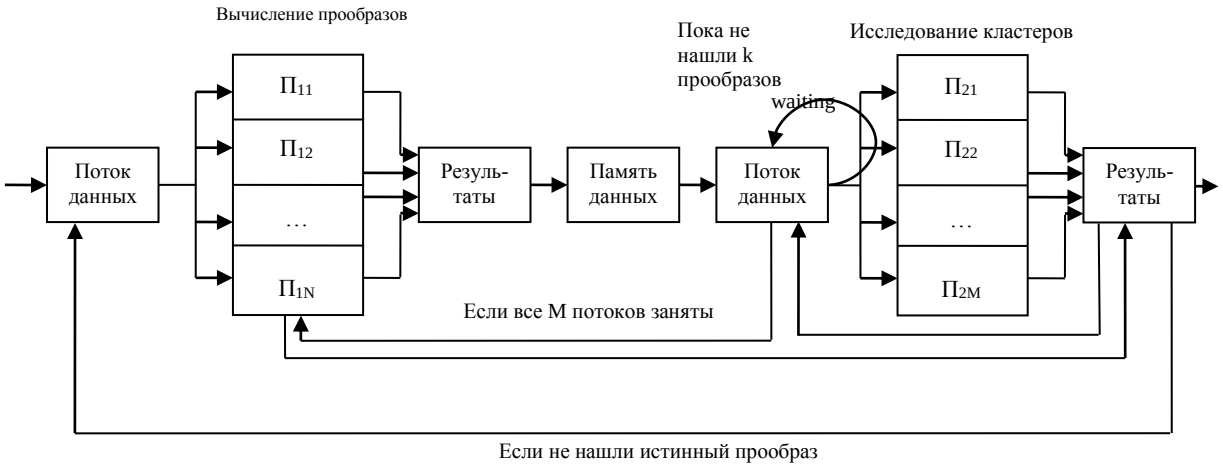
С целью повышения производительности процессы вычисления прообразов и их дальнейшего исследования выполняются асинхронно. Пока решение не найдено, полученный на очередном шаге кластер прообразов

помещается в динамическую структуру данных Q, организованную в виде очереди.

Главный поток извлекает кластеры прообразов из очереди и, при наличии свободных вторичных потоков выполнения, инициирует параллельное исследование их элементов на релевантность.

Найденные на очередном этапе наиболее релевантные элементы помещаются в очередь с приоритетом P, реализованную на основе двоичной кучи [77]. Максимальный размер такой очереди задается параметром $maxRelevanceQueueSize$. Подобная организация данных позволяет отсортировать исследованные элементы в соответствии с их релевантностью, при необходимости изменять их порядок следования при добавлении новых элементов и за константное время получать доступ к элементу с максимальным показателем. Процесс определения релевантных объектов построенных кластеров продолжается до тех пор, пока решение не будет найдено.

Общая структура алгоритма параллельного релевантного вывода может быть представлена в виде следующей схемы.



Ниже приведен алгоритм параллельного релевантного вывода.

// Параллельный релевантный вывод

$X^0 = null$

while $X^0 = null$ **do**

asynchronous call $\{X\} = getPreImagesCluster(b)$

if $\{X\} \neq \varnothing$ **then**

addClusterToQueue(Q, $\{X\}$*)* // Добавляем кластер в очередь *Q*

end

while not *clustersQueueIsEmpty(Q)* **do in parallel** // Если очередь не пуста

$\{Y\}$ = *extractClusterFromQueue(Q)* // Извлекаем кластер из очереди

foreach $(Y_j \in \{Y\})$ **do**

if $Y_j \bigcap F \neq \varnothing$ **then** $\{Y\} = \{Y\} \setminus Y_j$

end

while $X^0 = null$ **do**

asynchronous if $(relevanceQueueSize(P) < maxRelevanceQueueSize)$

then in parallel // Если в P есть свободное место

// Индекс k релевантного объекта и его показатель *relevance*

foreach $(Y_j \in \{Y\})$ **do**

if $Y_j \bigcap F \neq \varnothing$ **then** $\{Y\} = \{Y\} \setminus Y_j$

end

$k = getRelevantIndex\ (\{Y\}, T, relevance = 0)$

$InsertToQueue(P, k, relevance)$ // Объект в очередь с приоритетом

else

$Waiting()$ // Ждем, пока не освободится место в очереди

end

asynchronous while $(relevanceQueueSize(P) != 0)$ **do**

$k = ExtractMaxFromQueue(P)$ // Эл-т с макс. релевантностью

$Ask\ (y_k)$ // Определяем истинность k-го факта

foreach $Y_j \in \{Y\}$ **do**

if $Y_j \subseteq T$ **then**

$X^0 = Y_j$

break

```
            end
            if Y_j ⋂ F ≠ ∅ then
               foreach ( y_m ∈ {Y_j}) do
                  // Эл-т в очереди P - понижаем релевантность
                  if elemIsMemberOfQueue(y_m, P) then changeRelevance(y_m, P)
               end
            end
         end
      end
   end
end
```

В функции *getPreImagesCluster(b)* инициируется процесс вычисления фиксированного набора минимальных начальных прообразов атома b. Как уже было сказано, этот процесс осуществляется в отдельных потоках для обеспечения наибольшей эффективности.

Функция *getRelevantIndex* $(\{Y\}, T, relevance = 0)$ в отдельном потоке для каждого кластера находит релевантный элемент из нерассмотренных на данный момент. Она возвращает его показатель релевантности в переменной *relevance*. Далее элемент с показателем релевантности помещается в очередь с приоритетом. Если записываемый в очередь элемент уже находится в ней, то показатели релевантности складываются, и очередь перестраивается.

Параллельно с описанными действиями функция *Ask* (y_k) запрашивает внешний источник об истинности наиболее релевантного факта, извлеченного из очереди с приоритетом. При получении отрицательного ответа на запрос прообразы, содержащие элемент y_k, исключаются из рассмотрения, а релевантности уже исследованных элементов модифицируются.

При разработке алгоритмов решения сложных задач с использованием параллельных вычислений важно верно оценить эффективность их применения, то есть получаемое ускорение работы. С этой целью можно использовать модель вычислений в виде ациклического графа $G = (V, R)$, в котором множество операций, выполняемых в исследуемом алгоритме решения задачи, представляются, как множество вершин $V = \{1,..,|V|\}$, а информационные зависимости между операциями – в виде множества дуг R [61]. При этом дуга $r = (i, j)$ означает, что операция j использует результат выполнения другой операции i, а те операции алгоритма, между которыми нет пути, могут быть распараллелены.

Возможное описание параллельного выполнения алгоритма нахождения решения продукционно-логического уравнения в отдельных слоях с помощью графа изображен на следующем рисунке.

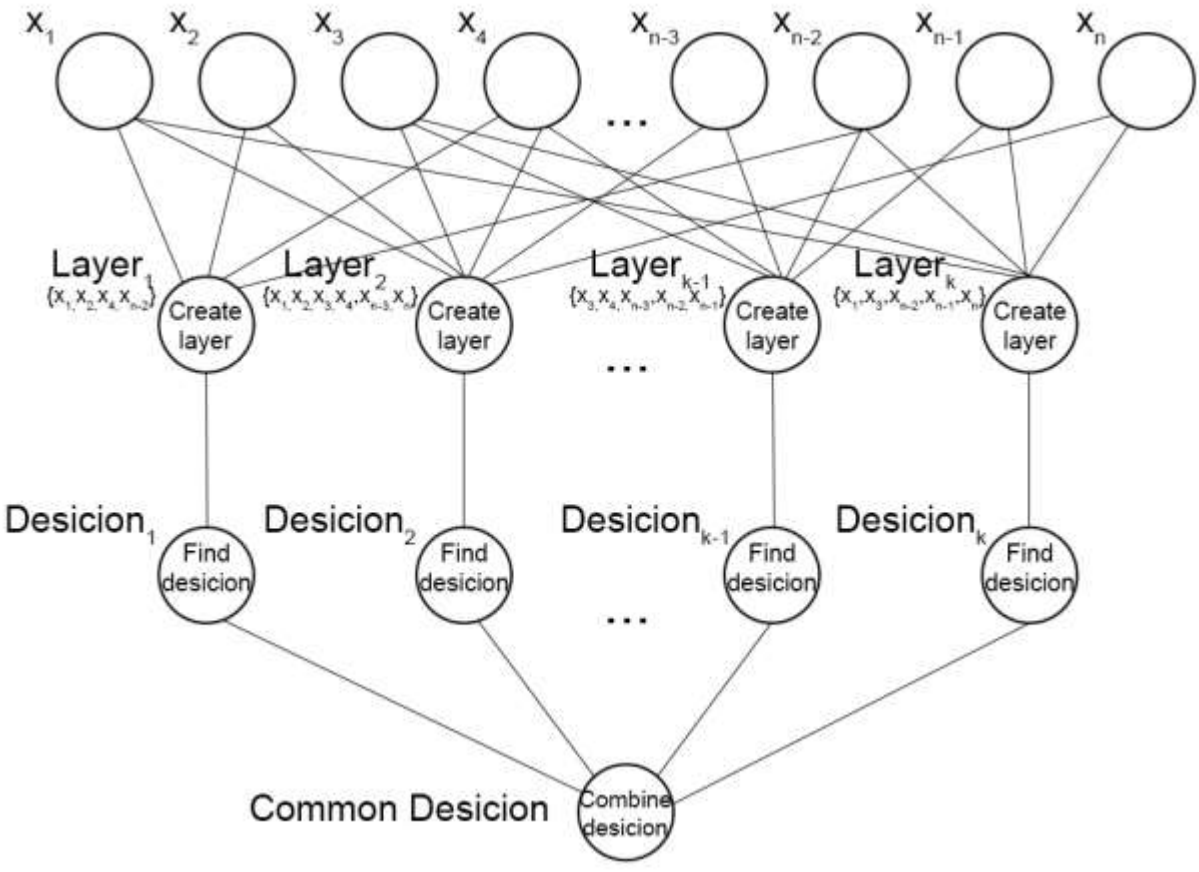


Рис. 1. Граф вычислений

Ниже приведен алгоритм соответствующей функции $getPreImagesCluster(b)$. Обозначим множество слоев $\{R_i\}$ через R'. Как уже говорилось, решение в каждом слое вычисляется отдельным потоком. Количество активных в данный момент потоков хранится в переменной $countUsedThreads$. В случае, если в пуле есть свободный поток, он запускается и в нем вызывается функция $FindEquationSolution(R_i)$, которая находит решение уравнения в очередном слое R_i.

```
// Нахождение решения уравнения на каждом слое в отдельном потоке
foreach R_i ∈ R' do in parallel
  if countUsedThreads < MaxThreads then
    BeginThread() // начать выполнение потока
    countUsedThreads++
    FindEquationSolution(R_i)
    ExitThread() // завершить поток
    countUsedThreads--
  else
    Waiting() // ждем освобождения потока
  end
end.
```

Функция $getRelevantIndex(\{Y\}, T, relevance)$ находит индекс k любого из наиболее релевантных и ранее не проверенных на истинность атомов, содержащихся в элементах текущего кластера прообразов $\{Y\}$ и его показатель релевантности $relevance$. При имеющемся обычно большом количестве прообразов процесс выявления релевантных объектов весьма ресурсозатратен, поэтому он также распараллеливается. Для очередного кластера прообразов в отдельном потоке (количество потоков аналогично ограничено параметром $MaxThreadsCount$) запускается процесс их релевантного исследования на предмет истинности. Количество активных в данный момент потоков хранится в переменной $usedThreadsNumber$. Здесь

возможны различные способы подсчета релевантности, в частности, описанные в п.2.4. Для синхронизации потоков используется механизм критических секций. Приведем граф вычислений для алгоритма вычисления показателя релевантности (рис. 2).

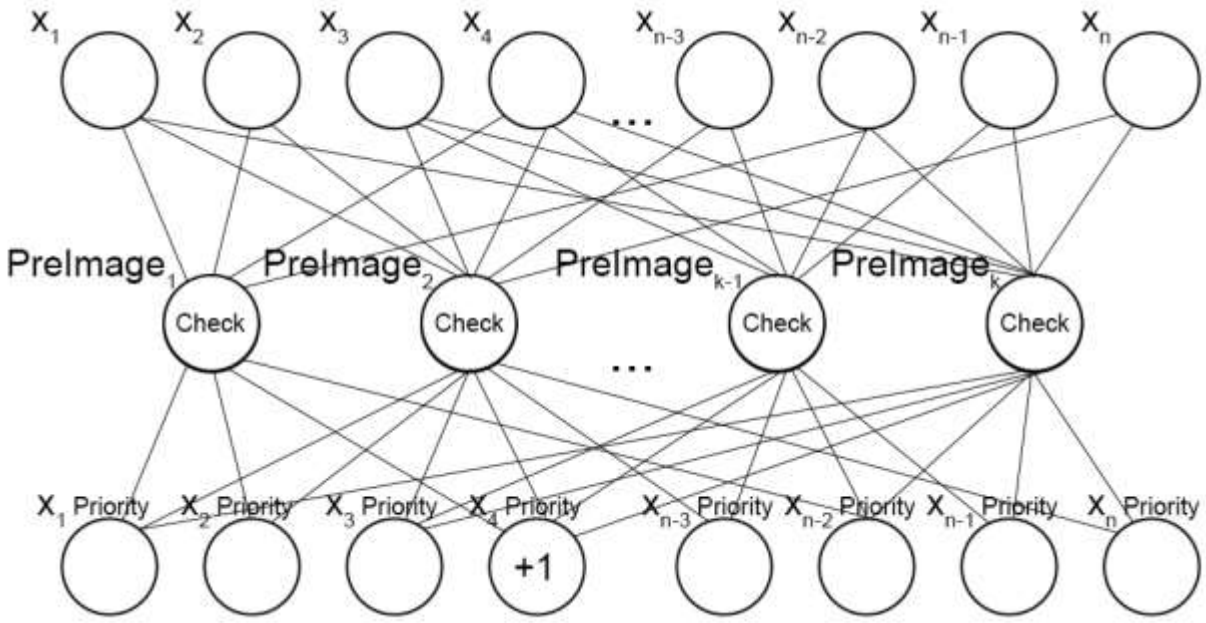


Рис. 2. Граф вычислений

Ниже представлен один из возможных параллельных вариантов функции *getRelevantIndex*.

```
// Параллельный подсчет релевантности
int getRelevantIndex ({Y}, T, relevance)
if (usedThreadsNumber < MaxThreadsCount) or
  (countUsedThreads < MaxThreads) then
   BeginThread() // начать выполнение потока
   if (usedThreadsNumber < MaxThreadsCount) then
     usedThreadsNumber++
     usedSelfThreads = true
   else
     countUsedThreads++
     usedSelfThreads = false
   end
```

$k = MostRelevantFind(relevance)$ // Индекс релевантного атома

$ExitThread()$ // завершить поток

if *usedSelfThreads* **then**

$usedThreadsNumber--$

else

$countUsedThreads--$

end

return k

else

$Waiting()$ // Ждем, пока поток не освободится

end

Используемая в этом алгоритме функция $MostRelevantFind(relevance)$ позволяет найти индекс k наиболее релевантного объекта и запомнить его показатель релевантности в переменной $relevance$.

Эксперименты показывают, что параллельный LP-вывод дает достаточно эффективные результаты для обработки баз знаний больших размеров.

Описанные выше идеи и приемы релевантного вывода приводят к уменьшению количества вызовов функции $Ask(y_k)$ – обращений за информацией к внешнему источнику, хотя при этом может производиться значительно больше вычислений по сравнению с обычным обратным выводом. Формальное обоснование данного утверждения представляет собой математически недоопределенную задачу, существенно зависящую от исходных данных. Далее в главе 4 эффективность LP-вывода будет обоснована результатами массированных экспериментов и их статистической обработкой: число внешних запросов в среднем снижается на 15–20%. Таким образом, на сегодня алгоритм LP-вывода можно назвать эвристическим в том смысле, что его преимущества показаны лишь экспериментально.

При LP-выводе может производиться значительно больше вычислений по сравнению с обычным обратным выводом, однако уменьшается количество обращений за информацией к внешнему источнику.

Этот факт перекликается с архитектурой современных процессоров (Intel и др.), поддерживающей спекулятивное выполнение команд и загрузку данных. Применяется, например, вынесение команд загрузки далеко вперед инструкций, использующих эти данные [103]. Таким образом, вместе с чтением необходимых данных происходит и загрузка тех, которые будут использоваться после текущих, а могут оказаться и вообще невостребованными. Некоторые вычислительные команды также выполняются заблаговременно, и впоследствии их результаты могут даже не потребоваться. Такая, на первый взгляд неэффективная, работа дает общее преимущество в скорости выполнения за счет уменьшения числа обращений процессора к памяти.

В нашей задаче снижается количество обращений машины вывода к внешнему источнику за счет увеличения числа операций, выполняемых во внутренней памяти компьютера – вычисления *всех прообразов*, с осознанием, что *все прообразы* скорее всего не понадобятся.

В следующей главе описывается компьютерная реализация LP-структур и связанного с ними метода релевантного обратного вывода, включая представленные ранее различные стратегии релевантности и параллельные вычисления. Успешность такой реализации подтверждает работоспособность и эффективность теоретических положений настоящей диссертационной работы.

Глава 3. Компьютерная реализация

При разработке теоретических положений не всегда уделяется должное внимание прикладным аспектам. В результате оказывается возможной ситуация, когда для практического применения этих положений окажется недостаточно интеллектуальных или технических ресурсов. Одна из основных целей настоящей главы – продемонстрировать возможность применения метода релевантного LP-вывода на практике. С этой целью описывается созданная автором новая версия интегрированной среды разработки и эксплуатации продукционных систем LPExpert, включающая библиотеку ParallelLPStructure, а также реализация и возможности применение LP-структур для верификации знаний и ускорения обратного вывода.

Указанный пакет существенно доработан по сравнению с его версией, описанной в работе [84]. Основные расширения возможностей LPExpert таковы: различные стратегии подсчета релевантности, параллельный LP-вывод, параллельное исследование прообразов, более гибкая и эффективная структура представления решеток с возможностью использования 64-разрядной архитектуры компьютера.

Рассматриваются вопросы кодирования LP-структур, опирающиеся на известные методы представления решеток битовыми векторами [18, 42]. Обсуждаются особенности и интерфейс объектно-ориентированного класса ParallelLPStructure, разработанного автором на языке C++ с использованием библиотеки STL. Класс ParallelLPStructure инкапсулирует свойства и методы описанной в главах 1–2 теории LP-структур, включая нахождение логической редукции и решение продукционно-логических уравнений.

3.1. Общие принципы реализации

Важный вопрос, возникающий при попытке компьютерной реализации LP-структур, состоит в обосновании формата представления решеток и заданных на них бинарных отношений [84]. Для булеана при количестве его атомов n общее число элементов составляет 2^n. Таким образом, в общем случае хранение отношения на решетке в виде статической матрицы (смежности или инциденций) не подходит. Небольшой учебный пример из [44] содержит около 80 элементарных значений объектов, которые при моделировании LP-структурой трансформируются в соответствующее количество атомов решетки. В данной ситуации использовались бы огромные разреженные матрицы, полное хранение которых в памяти компьютера было бы неэффективным и на практике вряд ли возможным. Поэтому бинарное отношение на решетке практически можно хранить лишь в виде динамического множества пар ее элементов. Следующие ниже замечания также подчеркивают основную особенность реализации LP-структур, а именно – потребность снижения расхода памяти, иногда в ущерб быстродействию.

Далее с точки зрения программной реализации рассмотрим задачу нахождения логической редукции LP-структуры. В приложении к экспертным продукционным системам ее решение заключается в получении минимальной базы знаний, эквивалентной исходной.

Как описано в п. 1.5, для бинарного отношения R на решетке $\mathbb{F}$ логическая редукция $\underset{\sim}{R}^0$ может быть получена последовательным выполнением следующих шагов:

1) добавить к R все пары вида (A,A), где $A \in \mathbb{F}_R$ (рефлексивные пары), и обозначить новое отношение R_1;
2) добавить к R_1 всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i, B_i) $(i = 1,...,n)$ принадлежат R_1;

3) объединить полученное отношение с отношением включения $\supset_R$ и обозначить новое отношение $\tilde{R}$;
4) построить транзитивную редукцию R^0 отношения $\tilde{R}$;
5) исключить из $\tilde{R}$ содержащиеся в нем пары вида $A \supset_R B$ и обозначить новое отношение R_{-1};
6) исключить из R_{-1} всевозможные пары (A,B) с элементами вида $A=\bigcup_i A_i$, $B=\bigcup_i B_i$, где все (A_i,B_i) $(i=1,...,n)$ принадлежат R_{-1} и не совпадают с (A,B);
7) исключить из полученного отношения все рефлексивные пары.

Из соображений экономии памяти физическое добавление к множеству новых пар следует производить лишь в случае необходимости. В частности, нецелесообразно хранить рефлексивные (шаг 1) или подчиненные (шаг 3) пары, если способ кодирования решетки будет допускать эффективное вычисление частичного порядка.

Непосредственная реализация шага 2 также потребовала бы чрезмерного расхода памяти, соизмеримого с построением булеана на универсуме R. Кроме того, в дальнейших вычислениях из огромного количества добавленных таким образом пар практически использовалась бы лишь незначительная часть. Как следствие, приходим к целесообразности динамического построения необходимых пар множества $\tilde{R}$.

Обсудим также вопрос построения транзитивной редукции (шаг 4) отношения $\tilde{R}$. Как показано в [2], эта задача вычислительно эквивалентна задаче нахождения транзитивного замыкания, что, например, при применении известного алгоритма Уоршолла [47] составило бы $O(N^3)$ операций. Существуют и улучшенные версии алгоритма построения транзитивного замыкания [34, 46, 52]. Однако в рассматриваемой ситуации число N заменяется выражением 2^N и, таким образом, нахождение транзитивной редукции посредством замыкания не представляется эффективным. В качестве практически реализуемого способа можно лишь

выбрать исследование множества пар на транзитивную избыточность. Таким образом, в описываемой реализации транзитивная редукция отношения $\tilde{R}$ вычисляется путем исключения пар, связанных транзитивными последовательностями.

Что касается процессов решения продукционно-логических уравнений в LP-структурах, то связанные с ними вопросы были подробно освещены в главе 2, вплоть до алгоритмов на псевдокоде, так что повторно возвращаться к ним здесь нецелесообразно.

3.2. Кодирование LP-структур

Способ представления в памяти LP-структур очевидным образом непосредственно связан с методами кодирования решеток. Данному вопросу посвящены работы [18, 42, 78], а также раздел монографии [71]. Как было отмечено выше, рассматриваемая реализация LP-структур должна экономить память и предоставлять быстрые «решеточные» операции (объединение, пересечение, частичный порядок). Данным требованиям во многом удовлетворяет кодирование решеток битовыми векторами.

Как указано, например, в [42], конечная булева решетка может быть представлена множеством битовых векторов (векторов с двоичными значениями компонент) одинаковой размерности N, где N – число атомов решетки. Каждому элементу a решетки соответствует единственный битовый вектор $BitVect(a)$, при этом каждому атому – вектор с одной единицей в соответствующей позиции и нулями – в остальных. Вычисление операций на решетке сводится к вычислению побитовых логических операций сложения ($|$) и умножения ($\&$) над компонентами векторов:

$a \vee b = BitVect(a) \,|\, BitVect(b)$;

$a \wedge b = BitVect(a) \,\&\, BitVect(b)$;

$a \le b \Leftrightarrow BitVect(a) \,\&\, BitVect(b) = BitVect(a)$;

$a \le b \Leftrightarrow BitVect(a) \,|\, BitVect(b) = BitVect(b)$.

Как известно (например, [103]), побитовые логические операции относятся к низкоуровневым и, соответственно, наиболее быстрым двуместным операциям в современных компьютерах. Теоретически с точки зрения анализа сложности алгоритмов нельзя утверждать, что операция над двумя векторами будет выполняться за константное время, поскольку разрядность компьютера (например, 32 или 64) может оказаться недостаточной для представления битового вектора единственным словом памяти. Однако величина сложности операции $N/32$ или $N/64$ вполне приемлема при общем количестве элементов решетки 2^N.

Поскольку представленная в данной главе реализация стандартных LP-структур использует конечные булевы решетки, то заявленные ранее потребности исчерпываются указанным выше способом кодирования решеток. Однако, если необходимо использование более общих видов решеток, можно обратиться, например, к исследованию [18], где описывается технология кодирования битовыми векторами произвольных решеток.

Еще один вопрос реализации LP-структур связан со способом хранения вторичных бинарных отношений на решетках. В соответствии с замечаниями п. 3.1, принято представление бинарного отношения в виде множества пар. Следует иметь в виду, что эффективность операций доступа к множеству (поиск, включение и исключение элементов) на практике обеспечивается, например, возможностью его хранения в виде сбалансированного бинарного дерева (так реализуются множества в STL – стандартной библиотеке C++ [64]). Такой подход требует сопоставления элементам хранимого множества уникальных вполне упорядоченных ключей. Ключ по возможности должен размещаться в слове или двойном слове компьютера. Тогда при доступе к множеству операция сравнения ключей будет выполняться за константное время. Таким образом, возникает идея хранения пары элементов решетки как пары ссылок на соответствующие им битовые векторы. Эти ссылки будут содержаться в смежных частях двойного целого числа, и данное число может использоваться в качестве упомянутого ключа.

Для хранения канонических отношений на решетке при реализации ряда алгоритмов будет применяться еще одно представление – в виде вектора множеств. Напомним, что каноническое отношение на атомно-порожденной решетке (п. 1.3) состоит из пар «хорновского» типа. Правая часть такой пары представляет собой атом решетки, а левая – объединение атомов. Для такого отношения можно построить массив, элементами которого являются подмножества элементов решетки. Индексом такого массива служит номер атома решетки (например, номер единственной ненулевой компоненты в соответствующем атому битовом векторе), а содержимым элемента массива – множество элементов решетки, составляющих с данным атомом пары в качестве их левых частей.

Описанное представление эффективно как с точки зрения использования памяти, так как хранятся лишь левые части пар, так и быстродействия, поскольку доступ к ним осуществляется по индексу в массиве. Этот способ применяется для реализации в LP-структуре алгоритмов параллельного обратного вывода при неизменном множестве правил.

3.3. Архитектура класса ParallelLPStructure

Рассмотрим особенности построения и основные возможности класса ParallelLPStructure, который реализует LP-структуры, описанные в главах 1–2. Разработка выполнена на языке C++ с привлечением библиотеки STL – Standard Template Library в среде программирования MS Visual Studio.

Как уже отмечалось в п.3.2, для хранения в памяти элементов решетки применяются битовые векторы. В классе ParallelLPStructure используется гибкая структура битового вектора – его длина может настраиваться вызывающей программой. Размерность такого вектора задается двумя параметрами – длиной в байтах одного кластера (*eItemSize*) и количеством этих кластеров (*eLengthItems*) [84].

Величина *eItemSize* определяет размер кластера – такой части вектора, которая обрабатывается одной логической операцией языка C++. Это значение хранится в виде статического константного поля класса и может быть изменено с перекомпиляцией. Выбором *eItemSize* = 1 можно структурировать битовый вектор в виде последовательности байтов, что приводит к экономии памяти и делает структуру вектора более прозрачной с точки зрения понимания деталей алгоритмов. Если же положить *eItemSize* = 8, то можно существенно повысить быстродействие операций над векторами, так как многие циклы в программе станут в 8 раз короче, а 64-битовые операции окажутся более эффективными для ЭВМ с соответствующей разрядностью памяти. По сравнению с описанной реализацией решеток в [84], настоящая работа допускает в качестве кластеров 64-разрядные числа, в то время как в [84] они были максимум 32-разрядными. В связи с данным обстоятельством, значительное число методов класса LPStructure реализовано в ParallelLPStructure с соответствующими изменениями.

Количество кластеров битового вектора хранится в целочисленном поле *eLengthItems*. Конструктор класса получает в качестве параметра соответствующую целую величину для настройки данного поля. Этот подход предоставляет возможность «клиентской» программе при создании LP-структуры динамически настраивать длину битовых векторов, исходя из размера решетки и соответствующего объема базы знаний.

В классе ParallelLPStructure определены типы *Elem* и *Pair* путем переименования соответственно типов обычного и двойного целых чисел. Тип *Elem* предполагает хранение адреса элемента решетки, а именно – соответствующего элементу битового вектора. Тип *Pair* содержит смежную пару величин *Elem*, то есть предназначен для хранения пары элементов (точнее, их адресов) из некоторого бинарного отношения на решетке. Конечно, типы *Elem* и *Pair* также можно было бы реализовать в виде классов с перегруженными операторами (|, &, <=), соответствующими операциям на

решетке. Однако решение использовать простейшие типы оказывается более последовательным в плане принятой стратегии экономии памяти.

При использовании для реализации LP-структур библиотеки STL могут широко применяться ее эффективные параметризованные контейнерные классы *vector* и *set*, а также их комбинации. В частности, для представления множеств элементов решетки полезным оказывается тип *ElemContainer*, определяемый как *set*<*Elem*>. Для хранения задаваемых на решетке бинарных отношений общего вида можно определяется тип *PairContainer* как *set*<*Pair*>. Некоторыми алгоритмами в качестве промежуточных данных используются векторы пар – *PairVector* (*vector*<*Pair*>). Каноническое бинарное отношение, в соответствии с изложенными выше соображениями, представимо вектором множеств *ElemContainerVector* (*vector*<*ElemContainer*>).

Величина *aMaxThread* задает максимально возможное количество рабочих потоков выполнения. Она хранится в виде статического константного поля объектно-ориентированного класса ParallelLPStructure.

Таким образом, перечислены основные типы, используемые в классе ParallelLPStructure.

3.4. Основная функциональность LP-структуры

Класс ParallelLPStructure инкапсулирует перечисленные ниже основные методы. Конструктор ParallelLPStructure (const int aLen) в качестве параметра принимает значение целочисленного поля eLengthItems для настройки длины битовых векторов.

Ряд методов предназначен для обеспечения стандартных операций над множеством пар LP-структуры. К ним относятся следующие функции, назначение которых следует из их названий:

BOOL insert (const Pair pair);

BOOL insert (const Elem left, const Elem right);

```
Pair extract ();
BOOL erase (const Pair pair);
BOOL erase (const Elem left, const Elem right);
void clear ();
ULONGLONG count () const;
```

Несколько методов реализуют базовую функциональность класса. Они имеют два общих параметра

```
const OnLPEvent onEvent = NULL,
const DWORD dwUser = 0.
```

Эти параметры обеспечивают возможность сообщения «клиенту» детальной информации о событиях, происходящих во время работы. Параметр onEvent (если задан) содержит адрес функции обратного вызова – обработчика LP-событий. Параметр dwUser, как это принято во многих функциях WinAPI [110], содержит сопутствующую информацию «пользователя». Обработчик LP-событий обязан иметь следующий прототип:

```
typedef void (__cdecl *OnLPEvent) (const Pair aPair,
                                   const int nEventType,
                                   const DWORD dwUser);
```

Здесь параметр aPair содержит пару элементов решетки, связанную с произошедшим событием. Параметр nEventType передает код события, dwUser возвращает информацию «пользователя», которая может быть использована им по собственному усмотрению, например, для идентификации источников событий. Предусмотрены следующие виды событий (возможные значения параметра nEventType):

```
static const int etRedundant = 0;  // Обнаружено избыточное правило
static const int etInference = 1;    // Найден элемент прямой логической связи
static const int etPreImageCount = 2; // Получено количество прообразов
static const int etPreImage = 3;      // Найден очередной начальный прообраз
```

Следующие два метода предназначены для построения логической редукции LP-структуры на основе анализа логических связей пар элементов.

Функция

BOOL isLConnected (const Pair aPair,

const OnLPEvent onEvent = NULL,

const DWORD dwUser = 0);

определяет наличие логической связи для пары элементов, задаваемой параметрами aPair. Остальные параметры описаны выше. Они позволяют «клиенту» получить структуру логической связи, если она будет обнаружена.

Функция

ULONGLONG lReductionLC (const OnLPEvent onEvent = NULL,

const DWORD dwUser = 0);

непосредственно строит логическую редукцию LP-структуры, обращаясь в процессе своей работы к функции isLConnected. Ее параметры позволяют при необходимости сообщить «клиенту» об обнаруженных избыточных парах. В качестве результата она возвращает количество пар редуцированной LP-структуры.

Метод

void maxImage (const Elem eSource,

const Elem eRes,

const OnLPEvent onEvent = NULL,

const DWORD dwUser = 0);

для заданного элемента решетки eSource вычисляет в LP-структуре его наибольший образ eRes. Параметры onEvent и dqUser позволяют «клиенту» получать информацию о парах, которые вносят вклад в прямой логический вывод.

Метод

BOOL minPreImages (const Elem eDest,

const Elem aNegative = 0,

const INT aMaxCount = 0,

const OnLPEvent onEvent = NULL,

const DWORD dwUser = 0,

const int nOptimize = otMemory);

предназначен для нахождения решений продукционно-логических уравнений. Для заданного элемента решетки eDest он вычисляет минимальные прообразы, о которых сообщает вызывающей программе посредством параметров onEvent и dwUser. Параметр nOptimize задает один из двух возможных режимов оптимизации работы – по памяти или по времени.

Два специальных параметра функции minPreImages носят более прикладной характер, чем остальные. Параметр aNegative, если он задан, содержит объединение начальных атомов решетки, которые должны быть принудительно исключены из процесса поиска прообразов. Указанные атомы могут, например, соответствовать заведомо ложным фактам базы знаний. Учет их наличия позволяет сократить время работы алгоритма вычисления прообразов. Параметр aMaxCount, если не равен нулю, содержит целое число значение, ограничивающее число искомых прообразов. Когда этот параметр положителен, при достижении указанного им количества найденных прообразов процесс прекращается, и найденные прообразы немедленно возвращаются «клиенту». Отрицательное значение параметра aMaxCount, точнее, его абсолютная величина, интерпретируется как период времени – количество миллисекунд, отведенных для вычисления прообразов.

Параметры aNegative и aMaxCount позволяют пошагово решать продукционно-логическое уравнение, получая и обрабатывая решения ограниченными порциями. Данный способ используется алгоритмом кластерно-релевантного LP-вывода.

Итак, представлены основные публичные методы класса ParallelLPStructure. Для их реализации используется также внутренняя функциональность, основное содержание которой описывается ниже.

Функция

void getLConnected (const Elem eDest, Elem eRes);

вычисляет объединение всех элементов решетки eRes, которые содержатся в прообразах заданного элемента eDest. При решении уравнения

предварительный вызов данной функции позволяет существенно сократить множество обрабатываемых элементов в слоях исходного отношения.

Следующие два перегруженных (в смысле объектно-ориентированного программирования [56]) метода

```
void minPreImagesOne (const Elem eDest, ElemContainer *ecRes);
void minPreImagesOne (const int nRItem,
                      const int nRBit,
                      ElemContainer *ecRes);
```

предназначены для нерекурсивного нахождения множества ecRes всех минимальных начальных прообразов заданного атома решетки в LP-структуре. В первом варианте функции исходный атом задается как элемент решетки eDest, во втором – номерами кластера nRItem и его разряда nRBit в соответствующем атому битовом векторе. Эти функции используются при решении продукционно-логических уравнений в режиме экономии памяти.

Функция

```
BOOL ecPursue (const int nItem, const int nBit);
```

рекурсивно находит множество всех минимальных начальных прообразов заданного атома решетки (nRItem и nRBit), а также всех промежуточных атомов, участвующих в данном рекурсивном алгоритме. Результаты формируются в векторе множеств – соответствующем поле класса, имеющем тип ElemContainerVector.

Предыдущие функции в своей работе обращаются к двум методам, позволяющим соответственно суммировать прообразы (объединять их как элементы решетки) и пополнять множество прообразов. При этом контролируется минимальность прообразов, то есть из множества исключаются приближенные решения – прообразы, содержащие другие прообразы.

```
BOOL addPreImageMin (ElemContainer *ecSum, ElemContainer *ecOne);
BOOL insPreImageMin (ElemContainer *ecDst, ElemContainer *ecSrc);
```

Ряд перечисленных далее методов обеспечивает выполнение основных решеточных операций. Она описаны в секции private, однако могут быть

перенесены в protected или public при расширении функциональности класса ParallelLPStructure:

```
BOOL EQ (const Elem ls, const Elem rs); // (ls) == (rs) в решетке
BOOL LE (const Elem ls, const Elem rs); // (ls) <= (rs) в решетке
BOOL LT (const Elem ls, const Elem rs); // (ls) < (rs) в решетке
BOOL EZ (const Elem ls); // (ls) == 0 в решетке
void lJoin (const Elem ls, const Elem rs);  // (ls) |= (rs) в решетке
BOOL lDiff (const Elem ls, const Elem rs);  // (ls) –= (rs) в решетке
```

Функция

```
void  minPreImageInParallel (const Elem eDest,
                             const ElemContainer * layer,
                             const CallbackEvent callback);
```

предназначена для нахождения минимального начального прообраза в слое layer, с использованием заданного атома решетки eDest. Результат работы возвращается обратно в главный поток путем вызова функции (указатель на которую передается в аргументе callback), имеющей следующий прототип:

```
typedef void (* CallbackEvent) (const ElemContainer result);
```

Для параллельного нахождения всех минимальных начальных прообразов используется функция

```
void threadedMinPreImages (const Elem eDest,
                           const Elem aNegative = 0,
                           const INT aMaxCount = 0,
                           const OnLPEvent onEvent,
                           const DWORD dwUser = 0),
```

которая для каждого из доступных слоев, получившихся в результате разбиения, запускает процедуру нахождения минимального начального прообраза в отдельном потоке, передавая указатель на соответствующий список аргументов для этой функции.

Параметр aMaxCount содержит значение, ограничивающее количество искомых прообразов. При достижении указанного количества найденных прообразов запускается процесс исследования их на истинность, в то время

как продолжается вычисляться следующая порция прообразов. Если значение aMaxCount равно нулю, то вычисляются все прообразы. После того как вычислена достаточная порция прообразов, функция сообщает о них вызывающей программе посредством параметров onEvent и dwUser.

Для работы с пулом потоков используются функции WinAPI. К ним относятся функции создания пула и установления максимального и минимального размера пула.

```
TP_POOL WINAPI CreateThreadpool (PVOID reserved);
BOOL WINAPI SetThreadpoolThreadMinimum (PTP_POOL ptpp,
                                        DWORD cthrdMin);

void WINAPI SetThreadpoolThreadMaximum (PTP_POOL ptpp,
                                        DWORD cthrdMost);
```

Функция инициализации среды ответного вызова:

```
void InitializeThreadpoolEnvironment (PTP_CALLBACK_ENVIRON pcbe);
```

Функция для связывания пула потоков со средой ответного вызова:

```
void SetThreadpoolCallbackPool (PTP_CALLBACK_ENVIRON pcbe,
                                PTP_POOL ptpp);
```

Функция для создания рабочих объектов и для работы с ними:

```
PTP_WORK WINAPI CreateThreadpoolWork (
                                PTP_WORK_CALLBACK pfnwk,
                                PVOID Context,
                                PTP_CALLBACK_ENVIRON pcbe);
```

Используется и ряд других функций WinAPI, роль которых менее значима для понимания архитектуры класса ParallelLPStructure.

Несколько функций разработаны для параллельного исследования истинности начальных прообразов в кластерах. Функция

```
void  findMaxCountOfPreimages (ElemContairerVector *ecRes,
                               const CallbackEvent callback);
```

предназначена для определения наиболее релевантного элемента в переданном кластере прообразов. Результат работы возвращается в главный

поток путем вызова функции, указатель которой передается в аргументе callback.

Функция

```
void threadedFindMaxCountOfPreimages (
                                    const OnParallelLPEvent  onParallelEvent,
                                  const DWORD dwUser = 0);
```

запускает описанную выше процедуру нахождения релевантного элемента кластера прообразов в отдельном потоке, передавая указатель на необходимый для функции findMaxCountOfPreimages список аргументов.

Параметр onParallelEvent (если задан) обеспечивает возможность сообщения «клиенту» детальной информации о результате работы функции. Он содержит адрес функции обратного вызова, которая должна иметь следующий прототип:

```
typedef void (__cdecl *OnParallelLPEvent) (const int nParallelEventType,
                                           const DWORD dwUser);
```

Параметр dwUser возвращает информацию пользователя, которая может быть использована им по собственному усмотрению.

Максимальное число потоков, одновременно выполняющих подсчет прообразов, ограничено полем maxThreadsCountForRelevance.

Переменная CRITICAL_SECTION section, используемая внутри методов работы с пулом, предназначена для синхронизации потоков. При вызове метода EnterCriticalSection() происходит блокировка секции, и последующие вызовы этого метода не возвратят управление вызывающему потоку до тех пор, пока секция не будет освобождена. При захвате критической секции используется поле LockCount, значение которого увеличивается на единицу при вызове метода EnterCriticalSection() и уменьшается на единицу, если вызван метод LeaveCriticalSection(). В случае, когда LockCount = 0, поток может получить данные, охраняемые критической секцией [96].

Таким образом, описана основная функциональность класса ParallelLPStructure, что также позволяет судить о принципах его построения. Для обеспечения возможности использования данного класса в других

системах его интерфейс оформлен в виде динамической C-библиотеки LPStructure.dll. Названия и параметры ее функций соответствуют представленным выше публичным методам класса ParallelLPStructure, поэтому нет необходимости в их более подробном описании. Заметим лишь, что почти каждая из этих функций имеет дополнительный параметр

const HANDLE lps;

который фактически представляет адрес экземпляра ParallelLPStructure. Полный список функций библиотеки LPStructure.dll выглядит следующем образом:

```
typedef  LPStructure :: Pair Pair;
typedef  LPStructure :: OnLPEvent OnLPEvent;
typedef  LPStructure :: OnParallelLPEvent OnParallelLPEvent;
typedef  LPStructure :: Elem Elem;
extern "C"  // Чтобы имена в DLL были предсказуемыми
{
  DLL_API  HANDLE lpsCreate (const int nLen);
  DLL_API  void lpsDelete (const HANDLE lps);
  DLL_API  BOOL lpsInsert (const HANDLE lps, const LPStructure::Pair aPair);
  // Получить пару из множества (и исключить ее)
  DLL_API  LPStructure::Pair lpsExtract (const HANDLE lps);
  DLL_API  void lpsGetBegLConnected (const HANDLE lps,
                                     const LPStructure::Elem eDest,
                                     LPStructure::Elem eRes,
                                        const LPStructure::Elem aSought = 0);
  DLL_API  ULONGLONG lpsLReductionLC (
                                  const HANDLE lps,
                                   const LPStructure::OnLPEvent onEvent = NULL,
                                  const DWORD dwUser = 0);
  DLL_API  BOOL lpsConnected (
                                   const HANDLE lps,
                                   const LPStructure::Pair aPair,
```

```
                                    const LPStructure::OnLPEvent onEvent = NULL,
                                   const DWORD dwUser = 0);
  DLL_API  void lpsMaxImage (
                                   const HANDLE lps,
                                   const LPStructure::Elem eSource,
                                   const LPStructure::Elem eRes,
                                      const LPStructure::OnLPEvent onEvent = NULL,
                                   const DWORD dwUser = 0);
  DLL_API BOOL lpsMinPreImages (
                                   const HANDLE lps,
                                   const LPStructure::Elem eDest,
                                   const LPStructure::Elem aNegative = 0,
                                   const INT nMaxCount = 0,
                                     const LPStructure::OnLPEvent onEvent = NULL,
                                   const DWORD dwUser = 0,
                                   const int nOptimize = LPStructure::otMemory);
   DLL_API void lpsThreadedMinPreImages (
                                          const HANDLE lps,
                                          const LPStructure::Elem eDest,
                                          const LPStructure::Elem aNegative = 0,
                                          const INT nMaxCount = 0,
                                            const LPStructure::OnLPEvent onEvent,
                                          const DWORD dwUser = 0);
   DLL_API void lpsThreadedFindMaxCountOfPreimages (
                              const HANDLE lps,
                                const LPStructure::OnParallelLPEvent onParallelEvent,
                              const DWORD dwUser = 0);
   DLL_API void lpsFreeLPCode (const HANDLE lps,
                                    const LPStructure::Elem eDest);
}
```

3.5. LPExpert – новая версия IDE

В настоящей диссертационной работе предусмотрено проведение массированных вычислительных экспериментов и подробного исследования полученных результатов методами математической статистики. Для реализации экспериментов необходим соответствующий программный инструмент. В качестве такого инструмента может быть выбран пакет программ LPExpert, разработанный С.Д. Махортовым и описанный в [84]. Он представляет собой интегрированную среду для создания и эксплуатации продукционных экспертных систем с использованием основных положений и результатов теории LP-структур.

Однако пакет LPExpert содержал лишь минимальные возможности применения методологии LP-вывода, которая на момент создания этого пакета еще не была достаточно разработанной и исследованной. Такая задача решается в настоящем диссертационном исследовании. Отсюда достаточно логичным выглядел бы переход к новой версии LPExpert, дополненной возможностями, которые открываются благодаря полученным в диссертации результатам. Таким образом, в задачи настоящего исследования вошла разработка новой версии указанного программного пакета, включающей возможности настройки и выбора подходящих параметров релевантности, параллельные вычисления истинных прообразов, а также взаимодействующей с описанным выше классом ParallelLPStructure.

Поскольку полное описание пакета LPExpert опубликовано в [84], в настоящем разделе более подробно остановимся лишь на изменениях, связанных с обновлением его версии.

Пакет оснащен многодокументным оконным интерфейсом пользователя и состоит из следующих основных блоков: редактор правил, компилятор правил, машина вывода. К его функциональности добавлена dll-библиотека – интерфейс описанного в п.3.3 класса ParallelLPStructure, что значительно расширяет возможности пакета в плане эффективности создания с его

помощью баз знаний и применения улучшенных алгоритмов обратного вывода, а также параллельных вычислений.

Расширения функций пакета LPExpert основаны на представлении наборов фактов и правил базы знаний LP-структурой (см. п.2.1). Возможности пакета в плане разработки и исследования баз знаний обеспечивает модуль интерфейса класса ParallelLPStructure. Эти средства включаются лишь при обнаружении на компьютере доступного файла LPStructure.dll. В этом случае данная библиотека динамически загружается, а модуль настраивает связи с ее функциями.

По окончании компиляции базы знаний, при наличии загруженной библиотеки LPStructure.dll, для каждого продукционного правила в памяти компьютера создается пара соответствующих правилу двоичных векторов (предпосылка и заключение). Эти векторы (точнее, их адреса) в дальнейшем используются при формировании LP-структуры. Общий размер векторов вычисляется на основе результатов компиляции. Атомами соответствующей решетки становятся пары «*объект = значение*». Для создания LP-структуры модуль последовательными вызовами функции *lpsInsert* формирует набор пар логического отношения. Далее можно использовать все возможности класса ParallelLPStructure, а именно – выявлять избыточные правила, исследовать образы и прообразы заданных подмножеств фактов базы знаний. Можно также выполнять обратный LP-вывод, предварительно задавая параметры релевантности и при этом использовать параллельные вычисления. Имеется функция проверки множества фактов на непротиворечивость. Дополнительные возможности рассчитаны на опытного разработчика баз знаний.

Поскольку пользователь пакета LPExpert работает в терминах базы знаний, для обмена информацией с классом ParallelLPStructure модуль содержит ряд интерфейсных функций. Прототипы основных из них (на языке Object Pascal) приводятся ниже (LP-кодом называется соответствующий подмножеству фактов двоичный вектор).

type

```
  TElem = ULONG; // Элемент LP-структуры
  TPair = Int64; // Пара отношения на LP-структуре
  TlpsCodeItem = BYTE; // Тип кластера LP-кода (можно увеличить)
  PlpsCode = ^TlpsCodeItem; // Тип указателя LP-кода
  TValuesArray = array of WORD; // Массив значений объектов

  // Обработчик LP-события
  TlpsEventHandler = procedure (const aPair: TPair;
                                const nEventType: Integer;
                                const dwUser: DWORD); cdecl;

  // Обработчик результатов функции подсчета релевантности
  TlpsParallelEventHandler = procedure (const aElem: TElem;
                                        const dwUser: DWORD); cdecl;
// Создать LP-структуру
function lpsMake(const bNew: Boolean = True): Boolean;
// Найти правило по LP-паре
function RuleByLP(const aPair: TPair): PRule;
// Добавить/удалить значение объекта в LP-коде
procedure addLegalToLPCode(const lpsCode: PlpsCode;
                           const nValue: WORD;
                           const bNegate: Boolean = False);
// Прочитать значение объекта из LP-кода
function getValuesFromLPCode(const lpsCode: PlpsCode;
                             aValues: TValuesArray): WORD;
// Сравнить два LP-кода
function cmpLPCode(const lpsLCode, lpsRCode: PlpsCode): Integer;
// Проверить LP-код на непротиворечивость
function tstLPCode(const lpsCode: PlpsCode): Boolean;
// Проверить LP-код на непротиворечивость с заданным значением
function tstLPCodeLegal(const lpsCode: PlpsCode;
```

```
                                const nValue: WORD): Boolean;
// Вычислить прообразы
procedure findPreImages (const aElem: TElem;
                         const aNeg: TElem,
                         const aCount: integer;
                         const aEvent:TlpsEventHandler;
                         const dwUser: DWORD);
// Посчитать релевантность
procedure PreImagesMaxCount (const aEvent: TlpsParallelEventHandler);
// Выдать LP-код в виде строки нулей и единиц
function showLPCode(const lpsCode: PlpsCode): string;
```

В LPExpert реализованы еще несколько новых функций, поддерживающих стратегии релевантности и параллельное исследование истинности прообразов. Функция

```
function checkMinPreImages (): Boolean;
```

инициирует проверку прообразов на предмет истинности. Далее с этой целью применяются описанные в Главе 2 методы релевантного обратного вывода. Функция

```
function getRelevant (typeRelevance: WORD, firstPoint: Boolean,
                      secondPoint: Boolean): WORD;
```

исследует множество полученных прообразов на релевантность.

Если параметр typeRelevance равен 0, то вызывается процедура

```
procedure getRelevantWithLinearAlgorithm ();
```

Она реализует метод LP-вывода с линейным подсчетом релевантности, сочетающим в себе два показателя.

В целях вычисления релевантности каждого элемента множества полученных прообразов вызывается функция

```
function getRelevanceInMaxCount () : integer;
```

Она инициирует проверку на принадлежность элементов прообразам для нахождения максимального количества тех прообразов, которые содержат в себе эти элементы.

Внутри данного метода происходит обращение к функции threadedFindMaxCountOfPreimages, которая определяет элемент, принадлежащий максимальному числу исследованных прообразов. В результате релевантность такого объекта увеличивается на 1.

Если typeRelevance равен 1, то вызывается процедура

procedure getRelevantWithProportionalLinearAlgorithm ();

Она инициирует пропорциональный подсчет релевантности, в процессе которого создается временный упорядоченный массив мощностей прообразов lengthArray. Его использование позволяет увеличивать релевантность объекта в зависимости от мощностей содержащих его прообразов.

Если typeRelevance меньше 0, то вызывается процедура

procedure getRelevantModifiedAlgorithms (firstPoint: Boolean,
secondPoint: Boolean);

Она инициирует процесс подсчета релевантности, предусматривающий использование только одного из двух показателей. Параметр firstPoint, установленный в *True*, означает увеличение рейтинга на 1 в случае, если элемент содержится в максимальном числе прообразов. Параметр secondPoint, если равен *True*, указывает, что при подсчете релевантности используется второй фактор, учитывающий присутствие элемента в прообразе минимальной мощности. Один из параметров функции должен иметь значение *False*, которое показывает, что соответствующий фактор релевантности необходимо исключить.

Интерфейс пользователя интегрированной среды разработки и эксплуатации продукционных систем LPExpert построен в многодокументном стиле (MDI-приложение). В новой версии он дополнен несколькими опциями.

Для задания способа расчета параметров релевантности при выборе меню *Работа* | *Режимы* реализован соответствующий диалог: вкладка *Способы подсчета релевантности* содержит компонент RadioGroup, состоящий из 4 RadioButton:

- линейный подсчет релевантности;
- пропорциональный подсчет релевантности;
- модифицированный линейный подсчет релевантности;
- исключение одного из коэффициентов релевантности.

Они позволяют выбрать нужный способ подсчета релевантности.

По умолчанию выбран первый способ. При указании четвертого способа появляются еще 2 RadioButton, позволяющие оставить лишь один из двух используемых показателей релевантности.

Для применения в LP-выводе параллельных вычислений добавлен соответствующий пункт меню: *Работа* | *Parallel LP-вывод*.

Итак, в настоящем разделе были описаны выполненные автором диссертации расширения архитектуры и функциональных возможностей пакета программ LPExpert. Они обеспечивают реализацию нового метода релевантного вывода с выбором способов подсчета релевантности и использованием параллельных вычислений.

В следующей главе описывается ряд массированных экспериментов, проведенных с использованием новой версии пакета LPExpert и программной библиотеки ParallelLPStructure. Результаты экспериментов и их статистическая обработка показали преимущества методов обратного вывода, предложенных в настоящей диссертационной работе.

Глава 4. Статистический анализ

При выполнении алгоритмов LP-вывода, описанных в главе 2, может производиться значительно больше вычислений по сравнению с обычным обратным выводом. Применяемая в настоящей работе стратегия направлена на уменьшение количества обращений за информацией к *внешнему источнику*, а также и общего времени работы за счет *распараллеливания задач*. Формальное обоснование уменьшения числа запросов представляет математически недоопределенную задачу, существенно зависящую от исходных данных. Поэтому на сегодня алгоритмы LP-вывода можно считать эвристическими в том смысле, что их преимущества подтверждаются лишь экспериментально. С учетом данного обстоятельства приобретает особое значение формальное сравнительное исследование результатов проводимых экспериментов. Аппаратом такого исследования может служить математическая статистика.

В настоящей главе приводятся и подробно описываются результаты статистической обработки данных, полученных в результате выполнения множества тестов для рассматриваемых в работе алгоритмов логического вывода. В разделе 4.1 представлены основная терминология и последовательность применения некоторых статистических методов обработки данных. Далее в главе приводятся результаты использования таких методов. Сравниваются количества внешних запросов при использовании обычного обратного вывода, релевантного вывода и его модификаций, а также кластерно-релевантного вывода.

С целью обоснования эффективности предложенных в работе алгоритмов LP-вывода было проведено порядка 500 автоматизированных тестов с «формальными» базами знаний, генерируемыми случайным образом с возможностью контроля «глубины», количества генерируемых правил и объектов. Описанные эксперименты позволяют определить количество выполняемых запросов к внешнему источнику информации в зависимости от

объема исходных фактов и правил. Показано, что число внешних запросов в среднем снижается на 15–20%.

Кроме этого, представлены итоги сравнения времени выполнения релевантного вывода с использованием параллельных вычислений и без них. Все полученные результаты сопровождаются иллюстрациями, сгенерированными при обработке данных в пакете Statistica [92], необходимыми диаграммами и графиками, по которым также можно судить о преимуществах рассматриваемых алгоритмов.

В завершение главы полученные при статистических исследованиях результаты систематизируются.

4.1. Основные теоретические сведения

В настоящей работе результаты проводимых экспериментов обрабатываются с помощью программного пакета Statistica 6. Этот пакет предназначен для всестороннего статистического анализа информации. Приведем базовые теоретические сведения, необходимые для дальнейшего изложения.

Статистическая проверка статистических гипотез

Статистической называют гипотезу о виде неизвестного распределения или о параметрах известных распределений. *Нулевой (основной)* называют выдвинутую гипотезу H_0. *Конкурирующей* (альтернативной) называют гипотезу H_1, которая противоречит нулевой. Различают гипотезы, которые содержат одно и более одного предположений. Если гипотеза однозначно фиксирует распределение наблюдений, то ее называют *простой*, в противном случае – *сложной* [69].

В итоге проверки гипотезы могут быть допущены ошибки двух родов.

Ошибка первого рода состоит в том, что будет отвергнута правильная нулевая гипотеза. Вероятность ошибки первого рода называют *уровнем*

значимости и обозначают α. *Ошибка второго рода* состоит в том, что будет принята неправильная нулевая гипотеза. Вероятность ошибки второго рода обозначают [69].

Статистическим критерием (или просто *критерием)* называют случайную величину K, которая служит для проверки гипотезы. *Наблюдаемым (эмпирическим)* значением $K_{набл}$ называют то значение критерия, которое вычислено по выборкам. *Критической областью* называют совокупность значений критерия, при которых нулевую гипотезу отвергают. *Областью принятия гипотезы (областью допустимых значений)* называют совокупность значений критерия, при которых нулевую гипотезу принимают [63].

Основной принцип проверки статистических гипотез: если наблюдаемое значение критерия принадлежит критической области, то нулевую гипотезу отвергают; если наблюдаемое значение критерия принадлежит области принятия гипотезы, то гипотезу принимают [63].

Критическими точками (границами) $k_{кр}$ называют точки, отделяющие критическую область от области принятия гипотезы.

Правосторонней называют критическую область, определяемую неравенством $K > k_{кр}$, где – положительное число.

Левосторонней называют критическую область, определяемую неравенством $K < k_{кр}$, где $k_{кр}$ – отрицательное число.

Двусторонней называют критическую область, определяемую неравенством $K < k_1, K > k_2$, где $k_2 > k_1$. В частности, если критические точки симметричны относительно нуля, то двусторонняя критическая область определяется неравенствами (в предположении, что $k_{кр} > 0$)

$K < -k_{кр}, K > k_{кр}$ или равносильным неравенством $|\mathrm{K}| > k_{кр}$.

Для отыскания критической области задаются уровнем значимости и ищут критические точки, исходя из следующих соотношений:

а) для правосторонней критической области $P(K > k_{кр}) = \alpha(k_{кр} > 0)$;

б) для левосторонней критической области $P(K < k_{кр}) = \alpha (k_{кр} < 0)$;

в) для двусторонней симметричной области $P(K > k_{кр}) = (\frac{\alpha}{2})(k_{кр} > 0)$,

$P(K < -k_{кр}) = \alpha / 2$.

Методы, которые для каждой выборки формально точно определяют, удовлетворяют ли выборочные данные нулевой гипотезе или нет, называются критериями значимости [69].

Критерии значимости подразделяются на следующие три типа.

1. Критерии значимости, служащие для проверки гипотез о параметрах распределений генеральной совокупности (чаще всего нормального распределения). Эти критерии называются параметрическими.
2. Критерии, которые для проверки гипотез не используют предположений о распределении генеральной совокупности. Эти критерии не требуют знаний параметров распределения, поэтому называются непараметрическими.
3. Особую группу критериев составляют критерии согласия, служащие для проверки гипотез о согласии распределения генеральной совокупности, из которой получена выборка, с ранее принятой теоретической моделью (чаще всего нормальным распределением).

Мощностью критерия называют вероятность попадания критерия в критическую область при условии, что справедлива конкурирующая гипотеза. Другими словами, мощность критерия – это вероятность не допустить ошибку второго рода (отклонить нулевую гипотезу) [69].

Сравнение двух дисперсий нормальных генеральных совокупностей

На практике задача сравнения дисперсий возникает, если требуется сравнить точность приборов, инструментов, сами методы измерений и т.д. Очевидно, предпочтительнее тот прибор, инструмент и метод, который обеспечивает наименьшее рассеяние результатов измерений, т.е. наименьшую дисперсию.

Пусть необходимо проверить гипотезу о том, что две независимые выборки получены из генеральных совокупностей и с одинаковыми дисперсиями σ_X^2 и σ_Y^2. Для этой цели используется F-критерий Фишера.

Порядок применения F-критерия следующий.

1. Принимают предположение о нормальности распределения генеральных совокупностей. При заданном уровне значимости α формулируется нулевая гипотеза $H_0 : \sigma_X^2 = \sigma_Y^2$ о равенстве дисперсий нормальных генеральных совокупностей при конкурирующей гипотезе $H_1 : \sigma_X^2 > \sigma_Y^2$.
2. Получают две независимые выборки из совокупностей и объемом n_X и n_Y соответственно.
3. Рассчитывают значения исправленных выборочных дисперсий S_X^2 и S_Y^2. Большую из дисперсий обозначают S_1^2, меньшую – S_2^2.
4. Вычисляют значение F-критерия по формуле $F_{набл} = S_1^2 / S_2^2$.
5. По таблице критических точек распределения Фишера-Снедекора, по заданному уровню значимости и числу степеней свободы $v_1 = n_1 - 1, v_2 = n_2 - 1$ (v_1 – число степеней свободы большей исправленной дисперсии), находится критическая точка $F_{кр}(\alpha, v_1, v_2)$.

   Если применяется двусторонний критерий ($H_1 : \sigma_X^2 \neq \sigma_Y^2$), критическую точку $F_{кр}(\alpha/2, n_1, n_2)$ ищут по уровню значимости $\alpha/2$ (вдвое меньшему заданного) и числам степеней свободы n_1 и n_2 (n_1 – число степеней свободы большей дисперсии).
6. Делается вывод: если вычисленное значение F-критерия больше или равно критическому ($F_{набл} \geq F_{кр}$), то дисперсии различаются значимо на заданном уровне значимости. В противном случае ($F_{набл} < F_{кр}$) нет оснований для отклонения нулевой гипотезы о равенстве двух дисперсий [63].

Сравнение двух средних нормальных генеральных совокупностей

В статистических исследованиях часто возникает задача сравнения средних значений двух генеральных совокупностей, представленных выборками. Для решения этой задачи в случае распределений, близких к нормальному, используется t-тест Стьюдента. Приведем алгоритм его использования.

Пусть имеются две выборки объемом n_1 и n_2. Проверяем нулевую гипотезу $H_0 : a_1 = a_2$ о равенстве двух средних нормальных совокупностей X и Y следующим образом.

1. Вначале вычисляются оценки средних $\overline{x}_1, \overline{x}_2$ и несмещенные оценки дисперсий S_1^2, S_2^2.

2. На заданном уровне значимости проверяется гипотеза о равенстве дисперсий $H_0^{\sim} : \sigma_1^2 = \sigma_2^2$ при альтернативной $H_0^{\sim} : \sigma_1^2 \neq \sigma_2^2$.

3. Если $H_0^{\sim}$ принимается, то вычисляется статистика $t = \dfrac{|\overline{x_1} - \overline{x_2}|}{S\sqrt{\dfrac{1}{n_1} + \dfrac{1}{n_2}}}$,

где $S^2 = (\dfrac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2})$. В случае с односторонней областью ($H_1 : a_1 > a_2$ или $H_1 : a_1 < a_2$) t сравнивается с $t_{кр} = t_{1-\alpha}(n_1 + n_2 - 2)$, найденным по таблице критических точек распределения Стьюдента, а в случае с двусторонней областью ($H_1 : a_1 \neq a_2$) – сравнивается с $t_{кр} = t_{1-\alpha/2}(n_1 + n_2 - 2)$. Если при этом $t \leq t_{кр}$, то H_0 принимается.

Если $H_0^{\sim}$ отвергается, то вычисляется статистика $t = \dfrac{|\overline{x_1} - \overline{x_2}|}{\sqrt{\dfrac{S_1^2}{n_1} + \dfrac{S_2^2}{n_2}}}$ и сравнивается с $t_{кр} = t_{1-\alpha}(k)$ (при этом для $H_1 : a_1 > a_2$ или $H_1 : a_1 < a_2$ берется односторонняя область) или с $t_{кр} = t_{1-\alpha/2}(k)$ (при этом для $H_1 : a_1 \neq a_2$ – двусторонняя), найденным по таблице критических точек распределения Стьюдента,

где $k = \dfrac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{\frac{(\frac{s_1^2}{n_1})^2}{n_1 - 1} + \frac{(\frac{s_2^2}{n_2})^2}{n_2 - 1}}$ (округляется до целого). Если при этом $t \le t_{кр}$, то H_0 принимается [63].

Исследования в пакете Statistica 6

Для сравнения арифметических средних двух групп данных может быть использован классический метод *t-критерий Стьюдента*. Основной целью этого метода является выявление значимости различия между средними.

Критерий Стьюдента относится к группе параметрических методов анализа. Его корректное применение требует выполнения следующих трех условий:

1) выборки должны быть *независимыми*, т.е. свойства одной из них никак не должны быть связаны со свойствами другой;
2) выборки должны подчиняться *нормальному закону распределения*;
3) между дисперсиями выборок не должно быть статистически значимой разницы (*однородность дисперсий*).

Теоретически t-критерий может применяться, даже если размеры выборок небольшие, и если переменные нормально распределены, а дисперсии наблюдений в группах не слишком различны. Предположение о нормальности можно проверить, исследуя распределение визуально с помощью гистограммы. В случае с большим объемом выборок t-критерий может применяться, если переменные не распределены нормально [62].

Если значения признака в двух сравниваемых группах распределены не нормально, применение параметрического t-теста для их сравнения может приводить к неверным результатам. В таких случаях следует воспользоваться соответствующим непараметрическим аналогом теста Стьюдента. Для сравнения двух независимых ненормально распределенных выборок используется *U-тест Манна-Уитни* [62].

Равенство дисперсий в двух группах можно проверить с помощью F-критерия. При вычислении различий между средними фактически анализируются выборочные дисперсии. Для выборки объема n выборочная дисперсия вычисляется как сумма квадратов отклонений от выборочного среднего, деленная на $n-1$. Таким образом, при фиксированном объеме выборки n дисперсия представляет собой функцию суммы квадратов [92].

Проверка значимости в таком методе анализа основана на сравнении компонент дисперсий, обусловленных межгрупповым и внутригрупповым разбросом. Если верна нулевая гипотеза (о равенстве средних значений в двух выборках), можно рассчитывать на сравнительно небольшое различие выборочных средних из-за случайной изменчивости. Поэтому при нулевой гипотезе внутригрупповая дисперсия практически будет совпадать с общей дисперсией, посчитанной без учета групповой принадлежности.

Полученные внутригрупповые дисперсии можно сравнить на основе F-критерия, показывающего, действительно ли отношение дисперсий значимо больше 1 [92].

Однако тест Стьюдента предназначен для сравнения исключительно *двух выборок*. В случае с большим числом выборок необходимо использовать дисперсионный анализ (ANOVA) [62].

Дисперсионный анализ позволяет проверить гипотезу об отсутствии различий между сравниваемыми группами в целом. Однако с его помощью невозможно узнать, какие именно группы различаются между собой. Для выяснения этого факта необходимо воспользоваться методами множественных сравнений. Механизм их работы заключается в проведении попарных сравнений средних значений всех групп, включенных в дисперсионный анализ.

Параметрические варианты однофакторного и многофакторного дисперсионного анализа помимо условий о нормальности и однородности групповых дисперсий предполагают также, что сравниваемые группы независимы.

Если же данные не распределены по нормальному закону, а объем выборок слишком мал для того, чтобы вообще сделать какие-либо выводы относительно вида распределения, параметрический дисперсионный анализ неприменим. В этом случае используется непараметрический *дисперсионный анализ Краскела-Уоллиса* [62].

Для визуальной оценки разницы между группами результатов экспериментов можно построить диаграммы размаха. Они представляют собой точку, соответствующую средней арифметической или медиане, которую окружает вертикально расположенный прямоугольник, длина которого соответствует одному из показателей разброса или точности оценки генерального параметра. Дополнительно от этого прямоугольника отходят отрезки, также соответствующие по длине одному из показателей разброса или точности. Таким образом, графики этого типа позволяют дать полную статистическую характеристику для каждой анализируемой выборки [50].

4.2. Исследование алгоритма релевантного LP-вывода

Для исследования эффективности алгоритма обратного вывода возьмем две выборки данных, полученных в результате экспериментов и представляющие собой количестве запросов к внешнему источнику в обычном обратном выводе и релевантном обратном выводе. Объемы выборок равны $n_1 = n_2 = 100$. Для анализа будем использовать результаты экспериментов, представленные в таблице А.1 (см. Приложение А).

4.2.1. Сравнение дисперсий генеральных совокупностей

Для сравнения методов обычного и релевантного обратного вывода возьмем две выборки, представленные в таблице А.1. Необходимо определить, обладают ли эти методы одинаковой эффективностью $H_0 : \sigma_X^2 = \sigma_Y^2$, если принять уровень значимости $\alpha = 0.05$ и в качестве конкурирующей гипотезы $H_1 : \sigma_X^2 > \sigma_Y^2$.

Найдем выборочные дисперсии и исправленные выборочные дисперсии генеральных совокупностей данных.

$$S_u^2 = \frac{\sum u_i^2 - |\sum u_i|^2 / n_1}{n_1 - 1} = 1176.39141 \text{ и } S_v^2 = \frac{\sum v_i^2 - |\sum v_i|^2 / n_2}{n_2 - 1} = 769.5976.$$

Сравним дисперсии. Найдем отношение большей исправленной дисперсии:

$$F_{набл} = \frac{S_X^2}{S_Y^2} = \frac{S_u^2}{S_v^2} = 1.5286.$$

По условию конкурирующая гипотеза имеет вид $\sigma_X^2 > \sigma_Y^2$, поэтому критическая область односторонняя и при отыскании критической точки следует брать уровни значимости равные $\alpha = 0.05$ и числам степеней свободы $v_1 = n_1 - 1 = 99, v_2 = n_2 - 1 = 99$ находим критическую точку $F_{кр}(0.05, 99, 99) = 1.394$.

Так как $F_{набл} > F_{кр}$, то гипотезу о равенстве дисперсий отвергаем – имеет место альтернативная гипотеза. Другими словами, исправленные дисперсии различаются значимо и, следовательно, релевантный метод обратного вывода обеспечивают большую эффективность. Кроме того, метод релевантного вывода обеспечивает наименьшую дисперсию, а, следовательно, является более эффективным алгоритмом по количеству внешних запросов.

4.2.2. Сравнение средних генеральных совокупностей

Для сравнения эффективности методов обычного и релевантного обратного выводов воспользуемся методом сравнения средних двух генеральных совокупностей, представленных выборками из таблицы А.1. При уровне значимости 0,05 установим, значимо или незначимо различаются результаты исследований, в предположении, что они распределены нормально.

Проверяется гипотеза $H_0 : a_1 = a_2$ при альтернативной гипотезе $H_1 : a_1 > a_2$. Вычислим оценки средних и дисперсий: $\bar{x}_1 = 57.85$; $\bar{x}_2 = 39.28$; $S_1^2 = 1176.39141$; $S_2^2 = 769.5976$;

Предварительно проверим гипотезу о равенстве дисперсий $H_0 : \sigma_1^2 = \sigma_2^2$: $\frac{S_1^2}{S_2^2} = 1.5286$; так как $F_{кр}(0.05, 99, 99) = 1.394$, то гипотеза о равенстве дисперсий отклоняется. Для проверки гипотезы о равенстве средних вычислим выборочное значение статистики критерия: $t = 4.2096$. Число степеней свободы $k = 198$.

Так как по таблице критических точек распределения Стьюдента имеем $t_{кр} = 1.97$, гипотеза о равенстве средних отклоняется и принимается альтернативная гипотеза. Полученное эмпирическое значение $t = 4.2096$ находится в зоне значимости. Другими словами, средние результаты измерений различаются значимо. Среднее значение выборки, полученной для релевантного вывода, существенно меньше соответствующего значения в обычном выводе. Полученные результаты свидетельствуют об уменьшении количества запросов в случае релевантного вывода, а следовательно, он обеспечивает большую эффективность.

В представленных выше методах необходимо, чтобы случайные величины были распределены нормально. Предположение о нормальности распределения будет проверено далее. Кроме того, специальные исследования показывают, что предложенные алгоритмы весьма устойчивы (особенно при больших объемах выборок) по отношению к отклонению от нормального распределения [50].

4.2.3. Исследования в пакете Statistica 6

Объем выборки в рассматриваемом случае равен количеству проведенных тестов, то есть ста элементам. Количество фактов созданной БЗ является случайной величиной, распределенной по дискретному равномерному закону. Две независимые группы наблюдений – количество вопросов,

заданных пользователю в обычном обратном выводе и в релевантном LP-выводе – являются функциями от этой случайной величины.

Проверим предположение о нормальности распределения с помощью гистограмм. В программе STATISTICA имеется специальный модуль – *Distribution fitting* (Подгонка распределения), позволяющий проверить соответствие анализируемых данных целому ряду математических распределений, в том числе определить, подчиняются ли данные о количестве внешних запросов нормальному распределению [62].

Получим гистограммы распределения данных и колоколообразную красную кривую, соответствующую ожидаемому нормальному распределению (у этого ожидаемого распределения те же средняя арифметическая и стандартное отклонение, что и в анализируемой совокупности данных) (рис. 4.2.3.1).

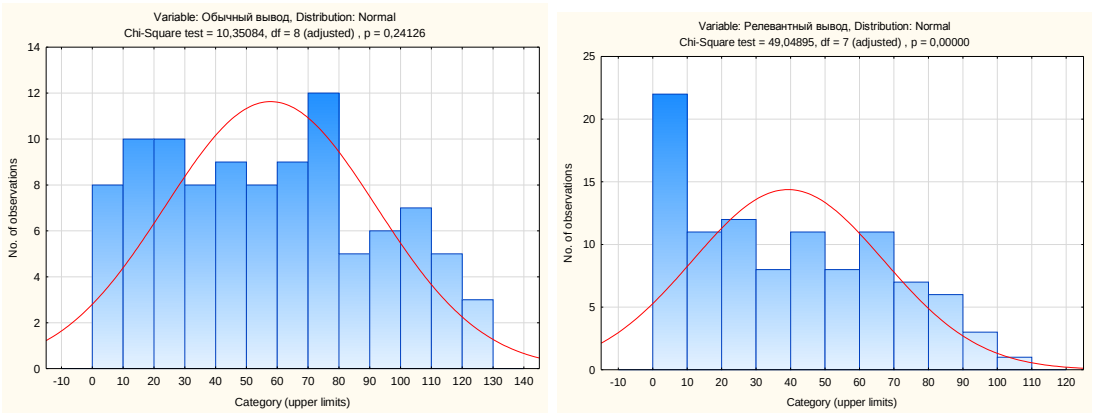


Рис. 4.2.3.1. Гистограммы для наборов данных

Глядя на полученный рисунок, можно сказать, что во втором случае имеются некоторые отклонения от нормального распределения. Это заключение, основанное на визуальном анализе распределения, имеет и более строгое подтверждение в виде результатов теста χ^2. В данном случае этот тест проверяет нулевую гипотезу о том, что наблюдаемое распределение анализируемого признака не отличается от теоретически ожидаемого нормального распределения. Поскольку вероятность ошибиться отклонив эту гипотезу оказалась намного меньше 0,05 (*для второго случая* P = 0,00000), принимаем, что гипотеза действительно неверна, а значит, распределение

значений количества запросов статистически отличается от нормального распределения.

Следует отметить, что мощность теста χ^2 при проверке нормальности распределения анализируемых данных относительно невысока, его применение достаточно часто приводит к ошибочному выводу о нормальности распределения. Поэтому лучше воспользоваться другими тестами, например, тестами Колмогорова-Смирнова и Лиллифорса на нормальность и W-тестом Шапиро-Уилка. Они проверяют нулевую гипотезу об отсутствии различий между наблюдаемым распределением признака и теоретическим ожидаемым нормальным распределением. Наиболее предпочтительным, особенно при небольших выборках является использование *W*-критерия Шапиро-Уилка, поскольку он обладает наибольшей мощностью в сравнении со всеми перечисленными критериями (т.е. чаще выявляет различия между распределениями в тех случаях, когда они действительно есть).

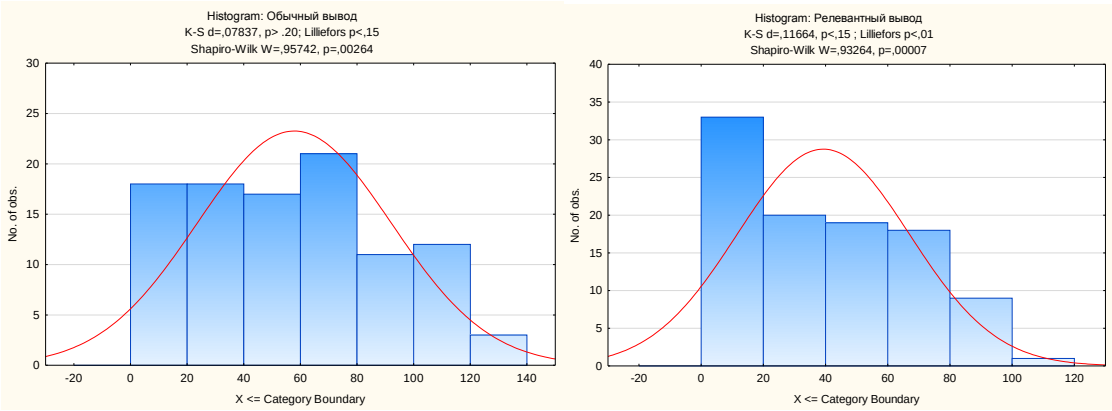


Рис. 4.2.3.2. Результаты применения тестов Колмогорова-Смирнова и Лиллифорса на нормальность и W-теста Шапиро-Уилка

Результаты выбранных тестов на нормальность автоматически располагаются в заголовке графика. При $P > 0{,}05$ можно заключить, что анализируемое распределение *не отличается* от нормального. В нашем примере полученные значения *P* не позволяют подтвердить предположение о нормальности распределения данных.

Кроме того, реализована еще одна проверка данных на нормальность распределения – с использованием графика нормальных вероятностей. Он

изображает зависимость ожидаемых нормальных частот значений признака от их реальных частот. Если между наблюдаемым и ожидаемым распределениями нет никакой разницы, точки на этом графике выстроятся строго вдоль прямой. Иначе они образуют фигуру, отличную от прямой.

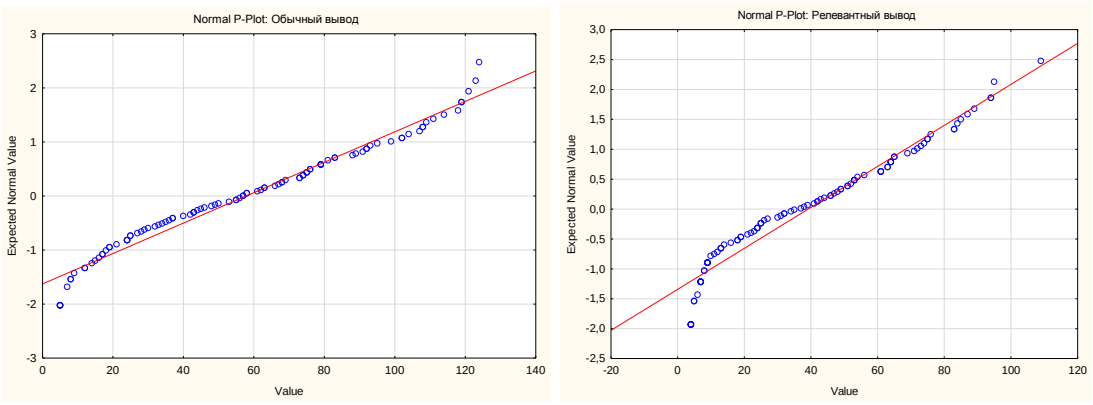


Рис. 4.2.3.3. Графики нормальных вероятностей для результатов обычного и релевантного вывода

Во втором случае наблюдается некоторый разброс точек относительно теоретически ожидаемой прямой, что еще раз доказывает, что распределение не является нормальным. Однако большие объемы выборок позволяют использовать t-критерий даже в случае ненормальности определения [62]. Применим *t*-тест для сравнения средних значений этих двух независимых выборок.

В результате применения t-критерия получена результирующая таблица (рис. 4.2.3.2.).

T-test for Independent Samples (Spreadsheet3)
Note: Variables were treated as independent samples

Group 1 vs. Group 2	**Mean Group 1**	Mean Group 2	t-value	df	p	Valid N Group 1	Valid N Group 2	Std.Dev. Group 1	Std.Dev. Group 2	F-ratio Variances	p Variances
Обычный вывод vs. Релевантный вывод	57,85000	39,28000	4,209609	198	0,000039	100	100	34,29856	27,74162	1,528580	0,035911

Рис. 4.2.3.4. Результат анализа в пакете Statistica 6 с применением t-критерия

В рассматриваемом примере p = 0.000039 << 0.05, на основании чего можно сделать вывод о наличии статистически значимых различий между сравниваемыми средними значениями.

Значение *p Variances=0.035911* < 0.05, откуда следует, что дисперсии сравниваемых выборок различаются (то есть условие однородности дисперсий не выполняется).

Поскольку по гистограммам наблюдалось некоторое отклонение от нормального распределения, можно воспользоваться соответствующим непараметрическим аналогом теста Стьюдента. Для сравнения двух независимых не нормально распределенных выборок используется *U-тест Манна-Уитни*.

Mann-Whitney U Test (Spreadsheet3)
By variable Вывод
Marked tests are significant at p <,05000

variable	Rank Sum обычный	Rank Sum релевантный	U	Z	p-value	Z adjusted	p-value	Valid N обычный	Valid N релевантный
Запросы	11625,00	8475,000	3425,000	3,847116	0,000120	3,847634	0,000119	100	100

Рис. 4.2.3.5. Результат теста U-таста Манна-Уитни

Так как $P << 0{,}05$ (5ый столбец), делается вывод о наличии статистически значимой разницы между сравниваемыми выборками. В отличие от *t*-теста, тест Манна-Уитни сравнивает не средние значения выборок, а суммы рангов по каждой из них. Ранг – это положение определенного значения изучаемого признака в упорядоченном по убыванию или возрастанию ряду.

Анализ данных с помощью t-критерия, сравнения средних и меры отклонения от среднего в группах можно производить с помощью диаграмм размаха.

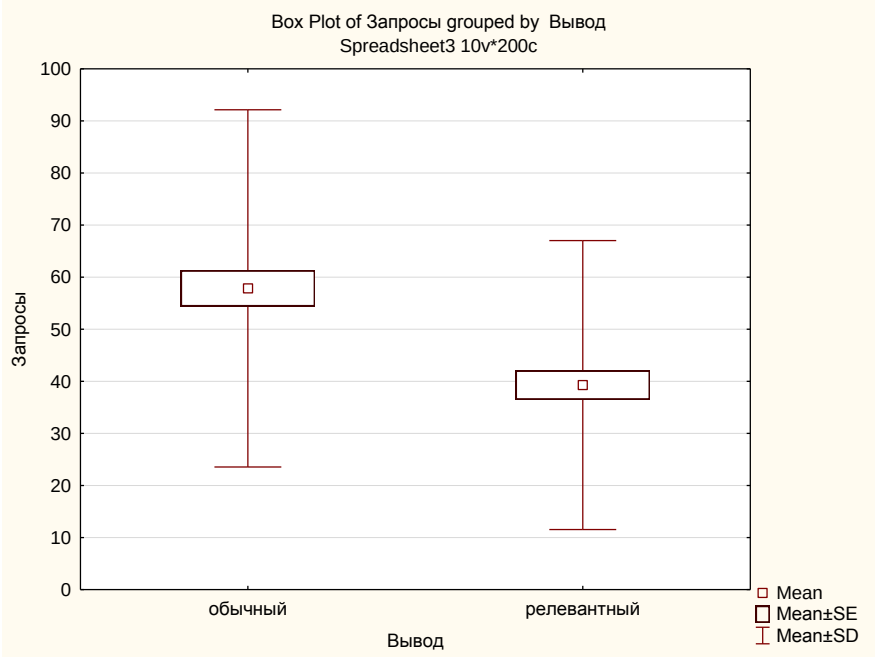


Рис. 4.2.3.6. Диаграммы размахов для результатов обычного и релевантного вывода

На графиках хорошо видно, что количество запросов при обычном логическом выводе значительно больше, чем при релевантном. Для полученных результатов были построены графики, представленные на рисунке 4.2.3.7. Они демонстрируют существенную эффективность релевантного LP-вывода.

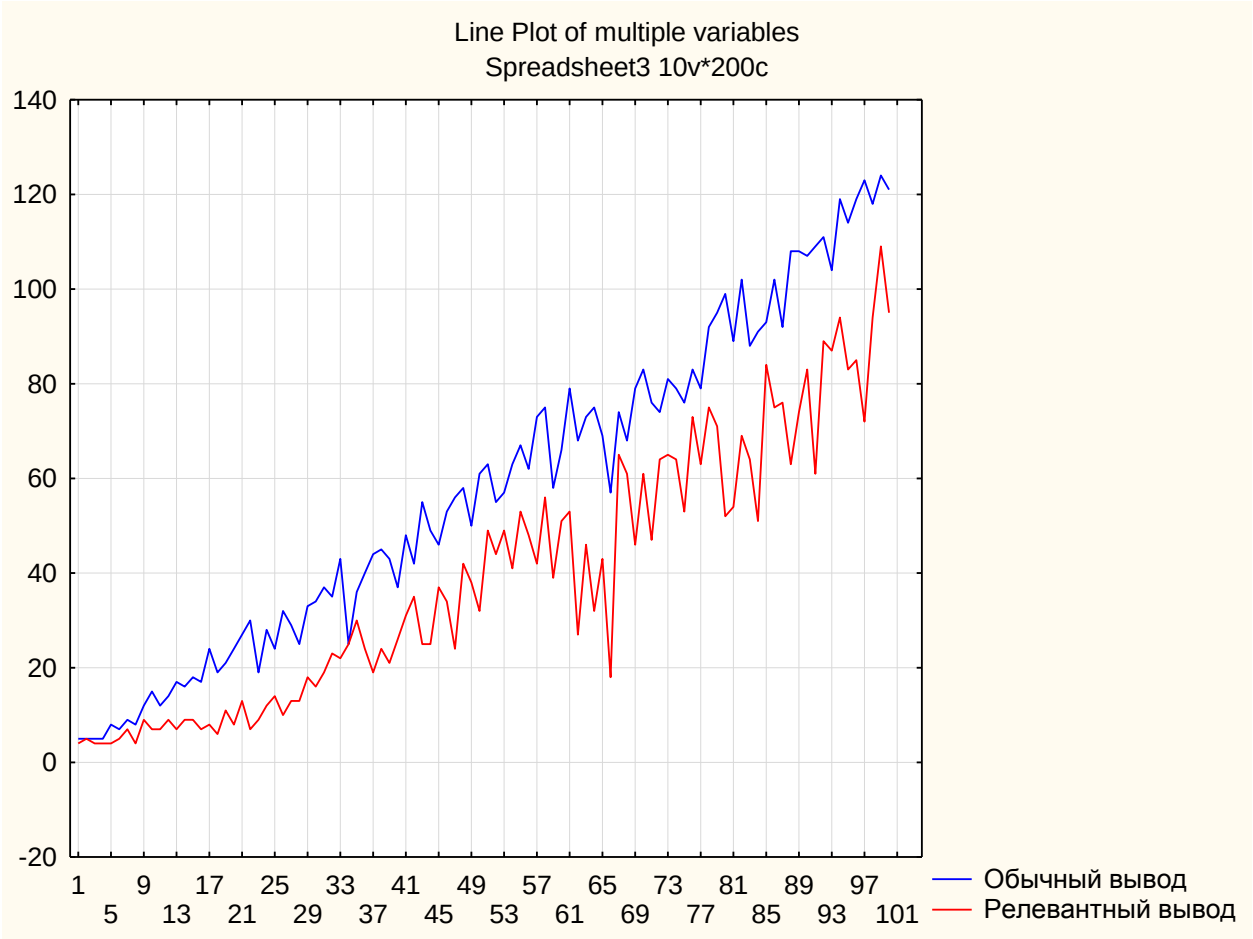


Рис. 4.2.3.7. График зависимости количества запросов от числа фактов

4.3. Исследование кластерно-релевантного LP-вывода

Для исследования эффективности кластерно-релевантного LP-вывода были созданы порядка пятисот баз знаний с большим количеством фактов и правил. В процессе нахождения решения количество прообразов ограничивалось в диапазоне от 1 до 200. В результате работы рассматриваемых алгоритмов получены данные о количестве обращений к внешнему источнику, представленные в таблице А.2 (Приложение А).

Тест Стьюдента и его непараметрические аналоги предназначены для сравнения исключительно *двух выборок,* и его нельзя применять для попарных сравнений более двух выборок [50]. Во избежание данной ошибки целесообразно использовать дисперсионный анализ.

Воспользуемся *однофакторным дисперсионным анализом,* чтобы проверить влияние одного фактора – максимального количества прообразов. Этот анализ параметрический, т.е. предполагает выполнение следующих обязательных условий в отношении данных:

1) в каждой из сравниваемых групп значения анализируемого признака распределяются *нормально*;
2) групповые *дисперсии однородны* (т.е. между ними нет статистически значимой разницы);
3) все сравниваемые выборки должны быть независимыми.

Перед получением результатов анализа следует проверить, выполняются ли указанные условия [62].

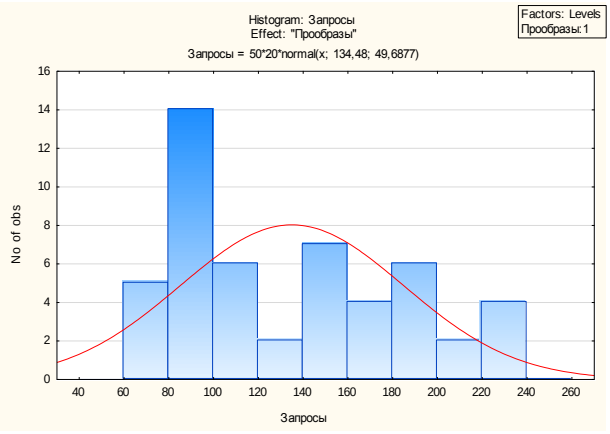
Для проверки однородности групповых дисперсий воспользуемся тестом Левена. Если результат этого теста указывает на отсутствие различий между дисперсиями ($P > 0{,}05$), то применение параметрического варианта дисперсионного анализа обосновано. В нашем примере различий действительно нет ($P = 0{,}975$).

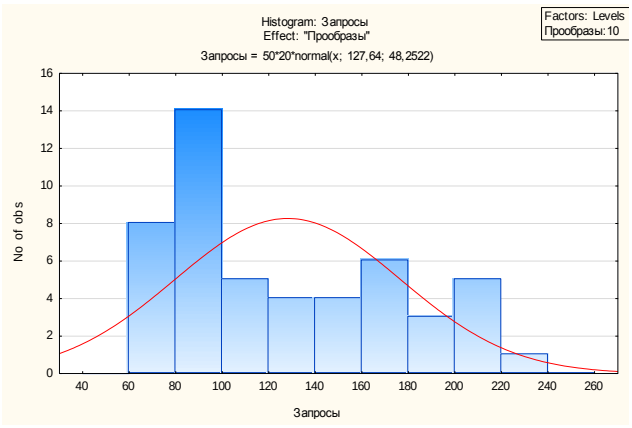
Levene's Test for Homogeneity of Variances (Spreadsheet4.sta)
Effect: "Прообразы"
Degrees of freedom for all F's: 6, 343

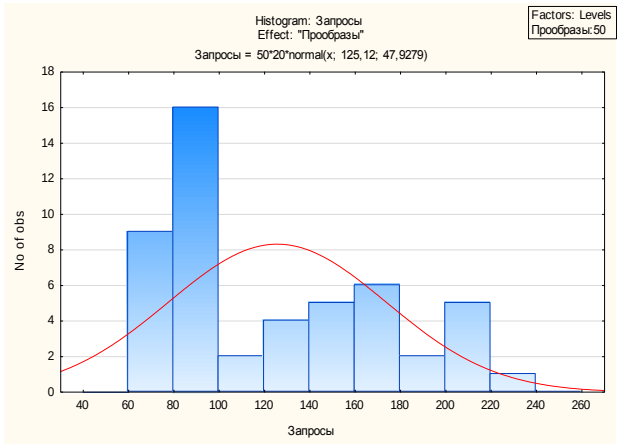
	MS Effect	MS Error	F	p
Запросы	98,03608	478,4081	0,204921	0,975148

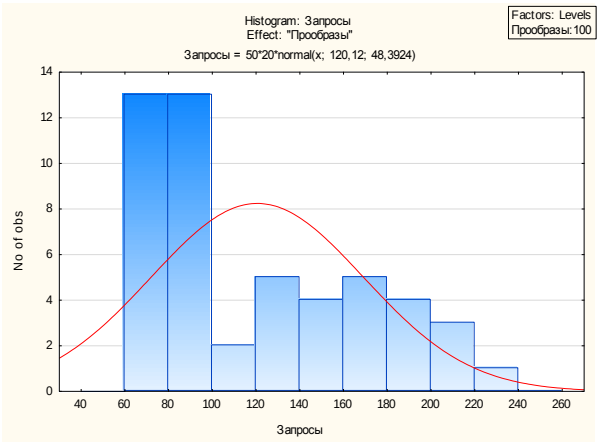
Рис. 4.3.2. Результаты теста Левена

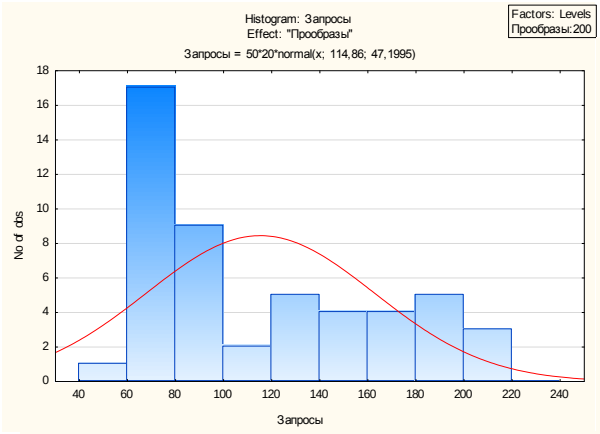
Для проверки нормальности распределения анализируемых данных можно воспользоваться графиком нормальных вероятностей или построить гистограммы. Результаты этих проверок приведены ниже.

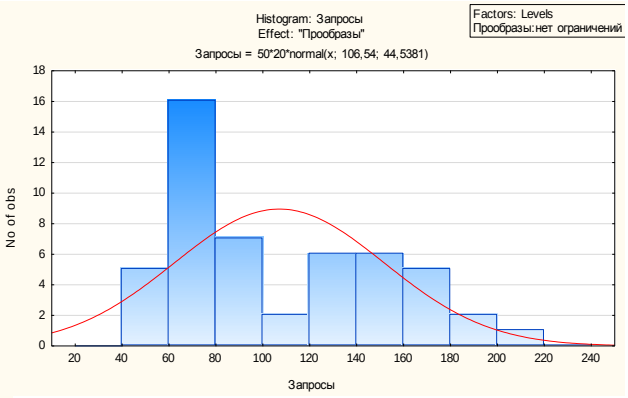

Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 134,48; 49,6877)
Factors: Levels
Прообразы:1
No of obs
Запросы

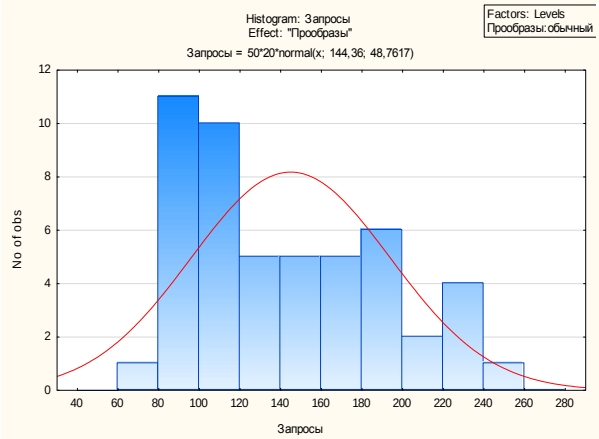

Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 127,64; 48,2522)
Factors: Levels
Прообразы:10
No of obs
Запросы

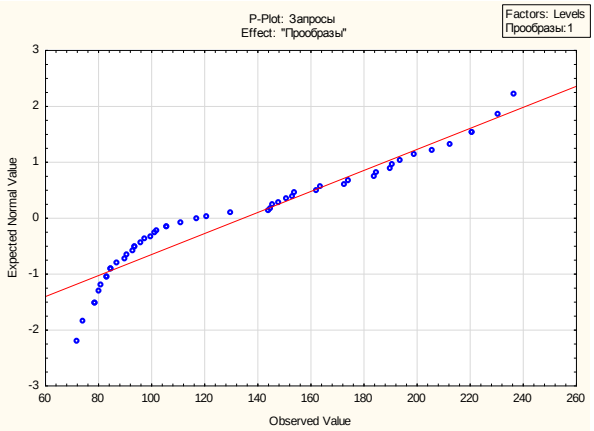

Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 125,12; 47,9279)
Factors: Levels
Прообразы:50
No of obs
Запросы


Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 120,12; 48,3924)
Factors: Levels
Прообразы:100
No of obs
Запросы


Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 114,86; 47,1995)
Factors: Levels
Прообразы:200
No of obs
Запросы


Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 106,54; 44,5381)
Factors: Levels
Прообразы:нет ограничений
No of obs
Запросы


Histogram: Запросы
Effect: "Прообразы"
Запросы = 50*20*normal(x; 144,36; 48,7617)
Factors: Levels
Прообразы:обычный
No of obs
Запросы


P-Plot: Запросы
Effect: "Прообразы"
Factors: Levels
Прообразы:1
Expected Normal Value
Observed Value

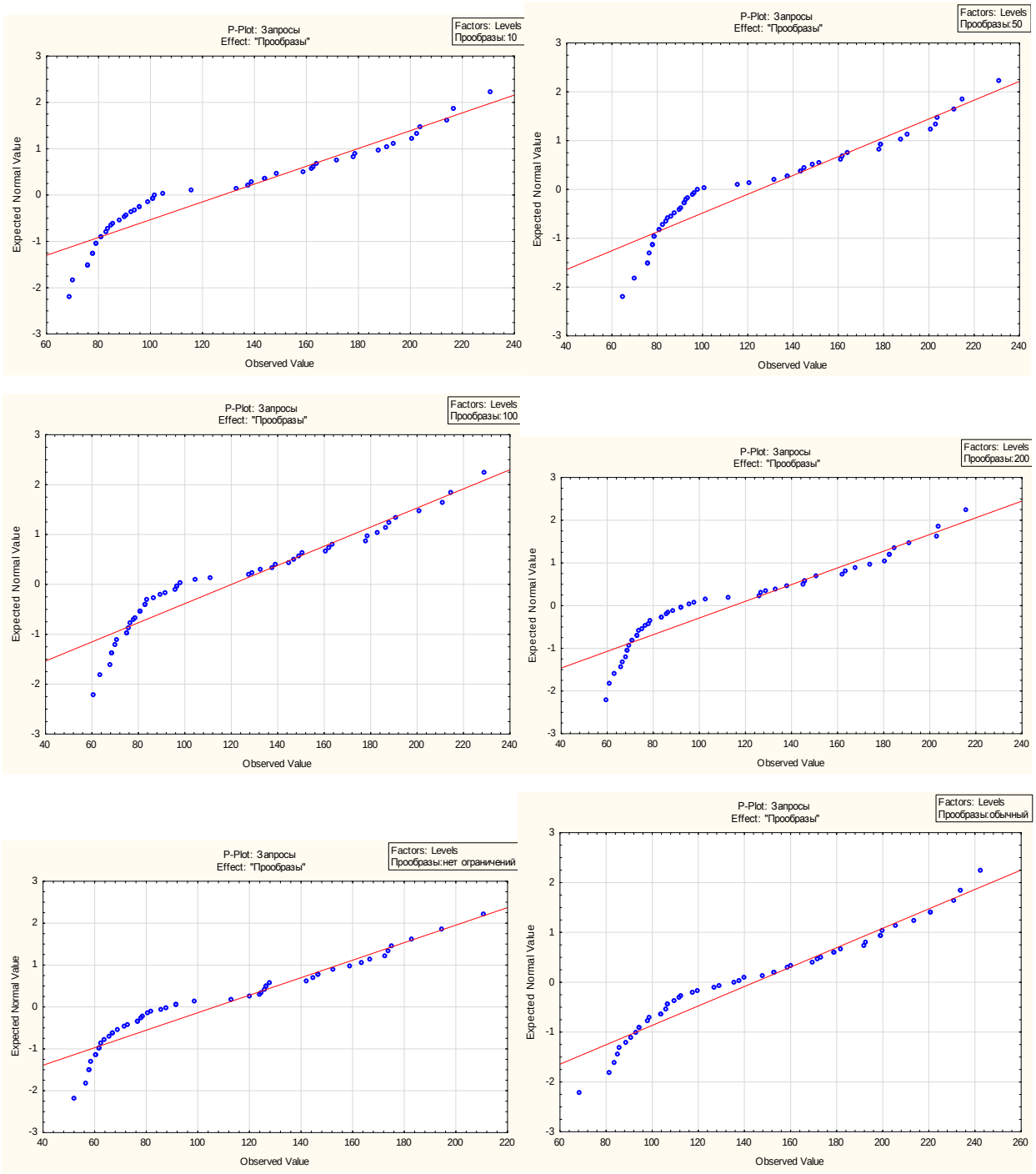


Рис. 4.3.3. Графики нормальных вероятностей и гистограммы для кластерно-релевантного с числом прообразов 1, 10, 50, 100, 200, релевантного вывода и обычного обратного вывода, соответственно.

Однако, следует отметить, что однофакторный дисперсионный анализ в определенной степени все же устойчив к нарушениям условий нормальности, причем чем больше общее число наблюдений, тем выше степень устойчивости.

Осталось проверить, насколько значимо различается количество запросов в зависимости от ограничений прообразов.

Univariate Tests of Significance for Запросы (Spreadsheet4.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	5445275	1	5445275	2378,563	0,000000
Прообразы	46928	6	7821	3,416	0,002741
Error	785235	343	2289		

Рис. 4.3.4. Результат проверки значимости различий количества запросов

В представленной таблице результатов величина ошибки *P* для нулевой гипотезы об отсутствии связи между количеством прообразов и запросами равна 0.002741. Так как $P << 0{,}05$, можно заключить, что среднее количество запросов различается статистически значимо в зависимости от максимального количества прообразов.

Однако дисперсионный анализ позволяет проверить лишь гипотезу об отсутствии различий между сравниваемыми группами в целом. С его помощью невозможно узнать, какие именно группы различаются между собой. Для выяснения этого факта необходимо воспользоваться методами множественных сравнений, являющихся частью апостериорного анализа. Механизм их работы заключается в проведении попарных сравнений средних значений всех групп, включенных в дисперсионный анализ.

Программа STATISTICA предлагает ряд тестов для множественных сравнений. Наиболее используемыми являются тесты Тьюки и Ньюмена-Кейлса. Результат выполнения одного этих тестов приведены на рисунке.

Tukey HSD test; variable Запросы (Spreadsheet4.sta)
Approximate Probabilities for Post Hoc Tests
Error: Between MS = 2289,3, df = 343,00

Cell No.	Прообразы	{1} 144,36	{2} 134,48	{3} 127,64	{4} 125,12	{5} 120,12	{6} 106,54	{7} 114,86
1	0		0,946624	0,584003	0,407781	0,147464	0,001514	0,033565
2	1	0,946624		0,991760	0,958800	0,744640	0,054269	0,382807
3	10	0,584003	0,991760		0,999973	0,986378	0,292642	0,835314
4	50	0,407781	0,958800	0,999973		0,998541	0,452540	0,936309
5	100	0,147464	0,744640	0,986378	0,998541		0,791704	0,998058
6	max	0,001514	0,054269	0,292642	0,452540	0,791704		0,977048
7	200	0,033565	0,382807	0,835314	0,936309	0,998058	0,977048	

Рис. 4.3.5. Результаты теста Тьюки

Из рисунка, в частности, видно, что статистически значимая разница в числе запросов существует между количеством запросов в кластерно-релевантном выводе с ограничением в 200 прообразов и количеством запросов в обычном выводе (на рисунке – 0 прообразов), а также между количеством запросов в обычном и релевантном выводе (на рисунке – max) ($P < 0,05$). Тогда как остальные случаи по эффективности не различаются ($P > 0,05$) (тест Тьюки выполнен для выборок с одинаковыми объемами).

В предыдущем случае не принималось во внимание исходное количество фактов в исследуемых базах знаний, а оно могло сказаться на количестве запросов. Поэтому нужно учесть данное обстоятельство и выполнить двухфакторный дисперсионный анализ (фактор 1 – число прообразов, фактор 2 – количество фактов).

Univariate Tests of Significance for Запросы (Spreadsheet4.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	4797848	1	4797848	46311,13	0,000000
Прообразы	41711	6	6952	67,10	0,000000
Факты	753644	11	68513	661,32	0,000000
Прообразы*Факты	4033	66	61	0,59	0,994240
Error	27558	266	104		

Рис. 4.3.6. Результаты двухфакторного дисперсионного анализа

Основные в этой таблице – вторая и третья строки. В конце этих строк приведены вероятности ошибок для нулевых гипотез об отсутствии влияния количества прообразов и числа фактов на количество запросов. Видно, что в обоих случаях $P < 0,05$. Это говорит о значительном влиянии данного фактора на количество запросов.

Как уже отмечалось, исследуемые данные не были распределены по нормальному закону. Вследствие этого также был применен непараметрический *дисперсионный анализ Краскела-Уоллиса* (или *H*-тест), хотя он и обладает несколько меньшей мощностью в сравнении с параметрическим вариантом. В результате получены результаты, представленные на рисунке.

Median Test, Overall Median = 108,500; Запросы (Spreadsheet4.sta)
Independent (grouping) variable: Прообразы
Chi-Square = 7,360000 df = 6 p = ,2888

Dependent: Запросы	0	1	10	50	100	max	200	Total
<= Median: observed	17,00000	23,00000	26,00000	26,00000	27,00000	28,00000	28,00000	175,0000
expected	25,00000	25,00000	25,00000	25,00000	25,00000	25,00000	25,00000	
obs.-exp.	-8,00000	-2,00000	1,00000	1,00000	2,00000	3,00000	3,00000	
> Median: observed	33,00000	27,00000	24,00000	24,00000	23,00000	22,00000	22,00000	175,0000
expected	25,00000	25,00000	25,00000	25,00000	25,00000	25,00000	25,00000	
obs.-exp.	8,00000	2,00000	-1,00000	-1,00000	-2,00000	-3,00000	-3,00000	
Total: observed	50,00000	50,00000	50,00000	50,00000	50,00000	50,00000	50,00000	350,0000

Рис. 4.3.7. Результаты анализа Краскела-Уоллиса

Величина ошибки $P > 0,05$ для нулевой гипотезы о том, что количество запросов в рассматриваемых случаях не различается, позволяет сделать вывод об отсутствии различий между сравниваемыми группами. Для наглядной демонстрации полученных результатов исследования построена диаграмма размаха и приведен график зависимости числа запросов от количества прообразов.

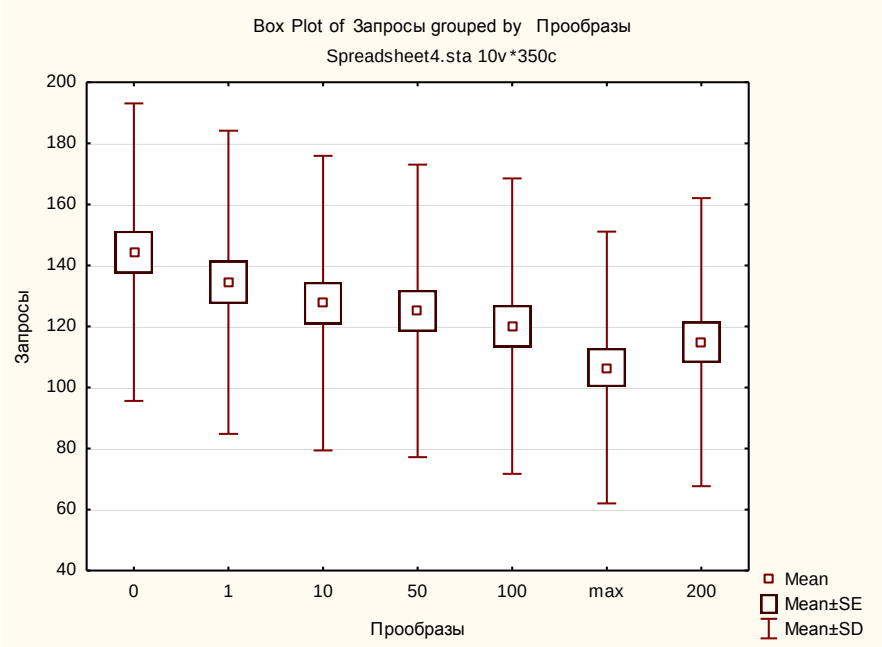


Рис. 4.3.8. Диаграммы размаха

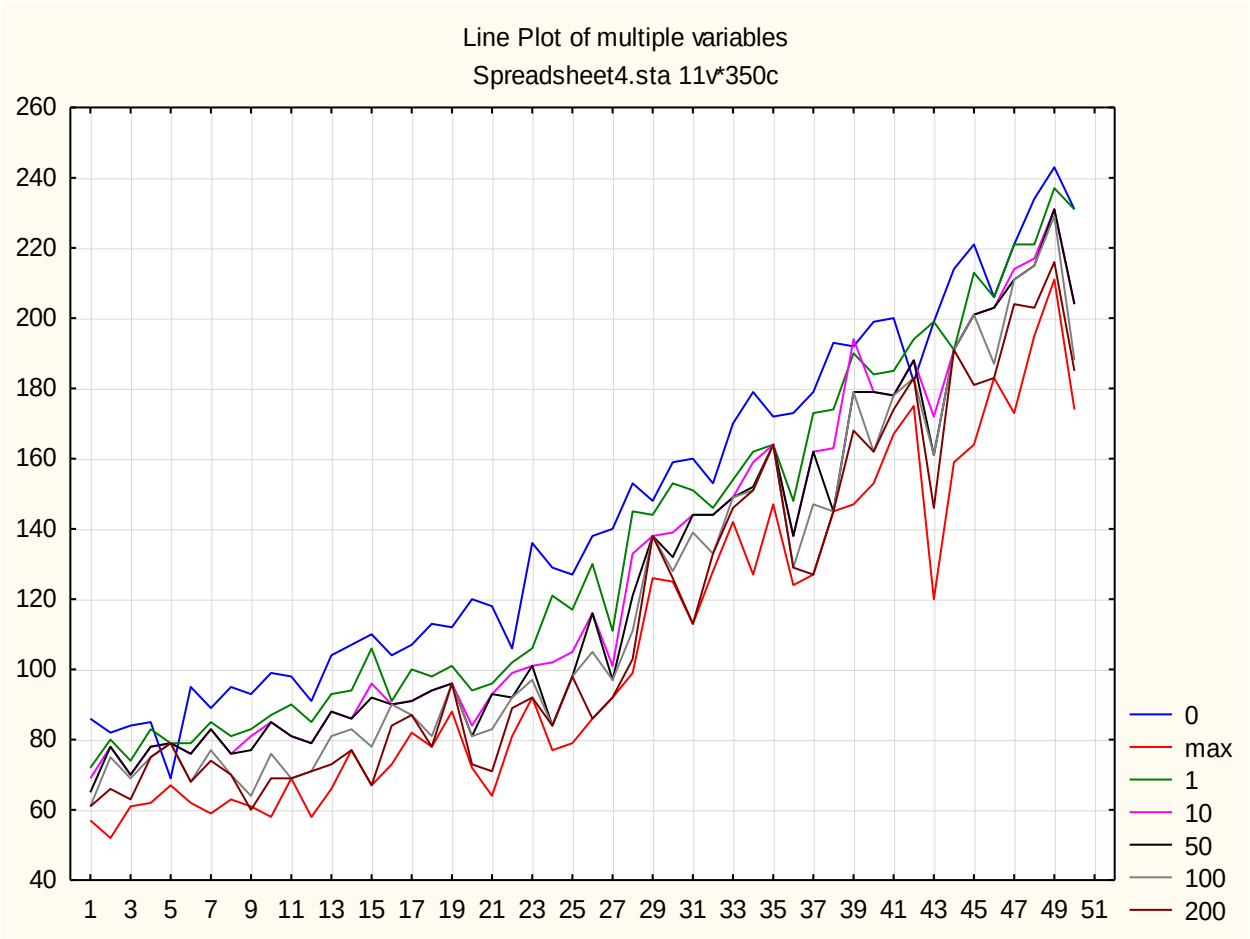


Рис. 4.3.9. График зависимости количества запросов от количества фактов в кластерно-релевантном выводе с числом прообразов 1, 10, 50, 100, 200, в релевантном выводе (красная линия) и обычном обратном выводе (синяя линия).

Представленные результаты приводят к следующему выводу. Кластерно-релевантный LP-вывод незначительно уступает релевантному выводу по числу внешних запросов, однако имеет неоспоримые преимущества перед обычным обратным выводом. Данное обстоятельство особенно актуально при обработке баз знаний больших размеров, когда релевантный LP-вывод неприменим в силу ограниченности ресурсов компьютера.

4.4. Применение пропорционального подсчета релевантности

Одна из потенциально наиболее эффективных модификаций релевантного вывода, предполагающая пропорциональных подсчет релевантности, также была протестирована на ряде примеров баз знаний с большим количеством

фактов и правил. В результате получены данные о количестве обращений к внешнему источнику в трех случаях, представленные в таблице А.3 (Приложение А).

Были проведены однофакторный и двухфакторный дисперсионные анализы, в ходе которых была установлена степень влияния типа метода и количества фактов в базе знаний на число внешних запросов. Для проверки однородности групповых дисперсий был использован тест Левена, результаты которого приведены на рисунке.

Levene's Test for Homogeneity of Variances (Spreadsheet5.sta)
Effect: "Тип метода"
Degrees of freedom for all F's: 2, 147

	MS Effect	MS Error	F	p
Запросы	26,49505	632,3398	0,041900	0,958977

Рис. 4.4.1. Результаты применения теста Левена

Результат этого теста указывает на отсутствие различий между дисперсиями ($P > 0,05$), поэтому применение параметрического варианта дисперсионного анализа обосновано.

Для проверки нормальности распределения анализируемых данных были построены графики нормальных вероятностей и гистограммы, приведенные на рисунке ниже.

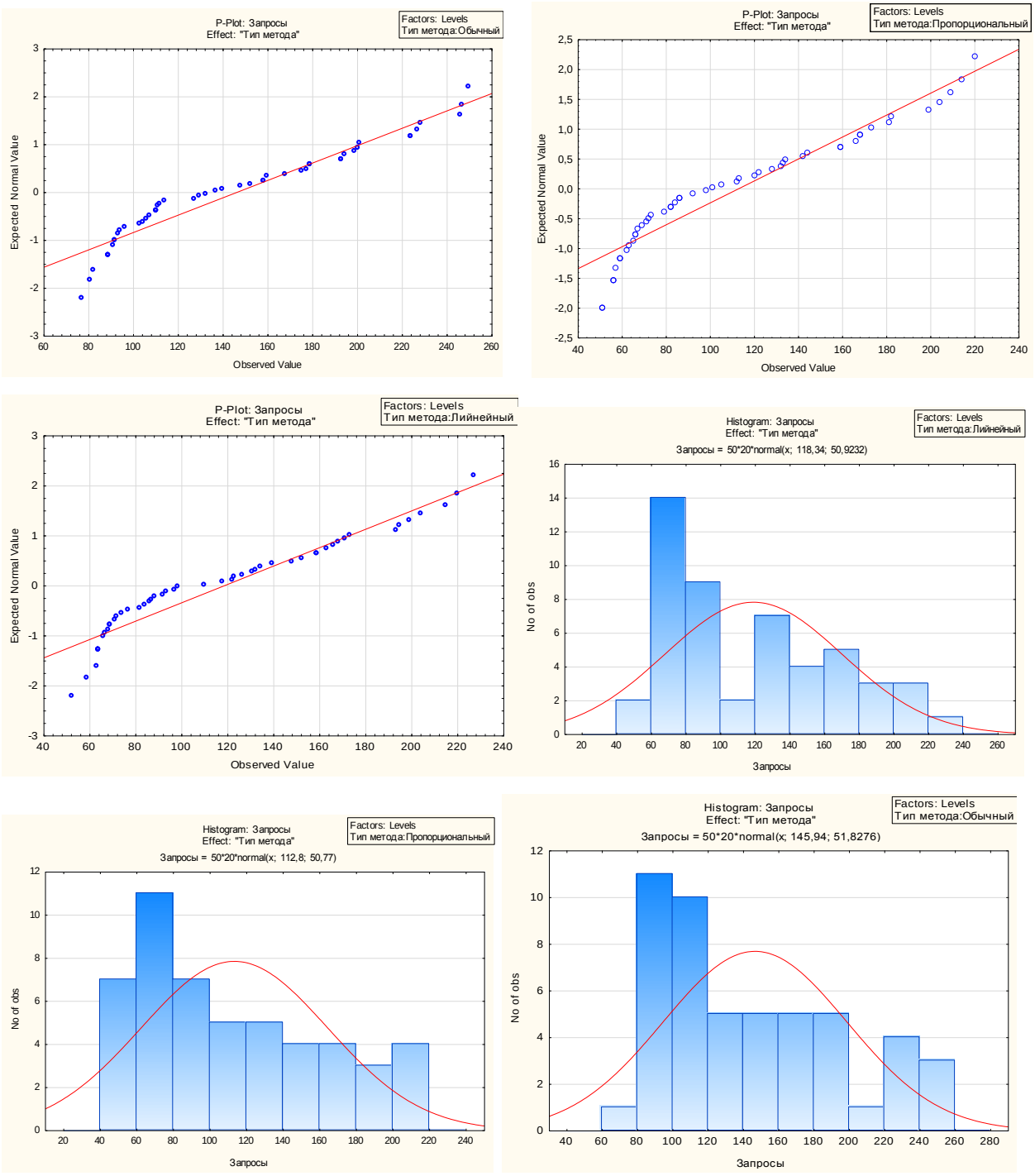


Рис. 4.4.2. Графики нормальных вероятностей и гистограммы для обычного обратного вывода, релевантного вывода и релевантного вывода с пропорциональным подсчетом релевантности, соответственно.

Снова наблюдается некоторое отклонение от нормального распределения, но объем выборок позволяет продолжить проверку. Результат дисперсионного анализа представлен на следующем рисунке.

Effect	Univariate Tests of Significance for Запросы (Spreadsheet5.sta) Sigma-restricted parameterization Effective hypothesis decomposition SS	Degr. of Freedom	MS	F	p
Intercept	2369822	1	2369822	904,8740	0,000000
Тип метода	31512	2	15756	6,0161	0,003081
Error	384986	147	2619		

Рис. 4.4.3. Результат дисперсионного анализа

Поскольку величина ошибки для нулевой гипотезы об отсутствии связи между типом метода и количеством запросов $P << 0,05$, можно заключить, что количество запросов статистически значимо различается в зависимости от типа метода.

Апостериорный анализ с помощью метода Тьюки показал, что статистически значимая разница в количестве запросов существует между парами релевантного метода с линейным подсчетом релевантности и обычного обратного метода, а также обычного обратного метода и релевантного метода с пропорциональным подсчетом релевантности.

Cell No.	Tukey HSD test; variable Запросы (Spreadsheet5.sta) Approximate Probabilities for Post Hoc Tests Error: Between MS = 2619,0, df = 147,00 **Тип метода**	{1} 118,34	{2} 112,80	{3} 145,94
1	Лийнейный		0,850932	0,019207
2	Пропорциональный	0,850932		0,003458
3	Обычный	0,019207	0,003458	

Рис. 4.4.4. Результаты теста Тьюки

Двухфакторный дисперсионный анализ показал значительное влияние выбранного метода и исходного числа фактов на количество запросов ($P << 0,05$).

Effect	Univariate Tests of Significance for Запросы (Spreadsheet5.sta) Sigma-restricted parameterization Effective hypothesis decomposition SS	Degr. of Freedom	MS	F	p
Intercept	2081556	1	2081556	19920,25	0,000000
Факты	372450	11	33859	324,03	0,000000
Тип метода	28296	2	14148	135,39	0,000000
Факты*Тип метода	624	22	28	0,27	0,999554
Error	11912	114	104		

Рис. 4.4.5. Результаты двухфакторного дисперсионного анализа

Вследствие имеющихся отклонений от нормальности распределения применен также непараметрический однофакторный *дисперсионный анализ Краскела-Уоллиса*, который показал наличие статистически значимых различий между сравниваемыми группами (P<<0,05).

Kruskal-Wallis ANOVA by Ranks; Запросы (Spreadsheet5.sta)
Independent (grouping) variable: Тип метода
Kruskal-Wallis test: H (2, N= 150) =12,90668 p =,0016

Depend.: Запросы	Code	Valid N	Sum of Ranks	Mean Rank
Лийнейный	101	50	3460,500	69,21000
Пропорциональный	102	50	3201,000	64,02000
Обычный	103	50	4663,500	93,27000

Рис. 4.4.6. Результаты дисперсионного анализа Краскела-Уоллиса

Для наглядности построены также диаграмма размаха и графики зависимости количества запросов от исходного числа фактов.

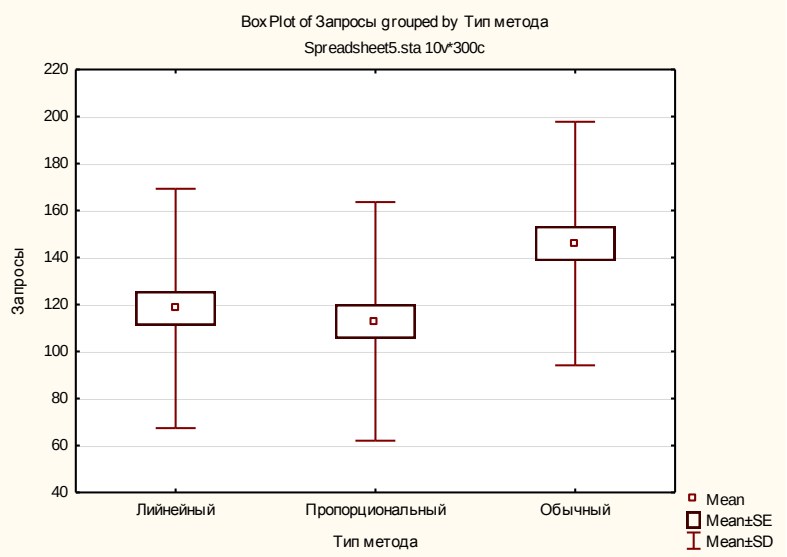


Рис. 4.4.7. Диаграммы размаха

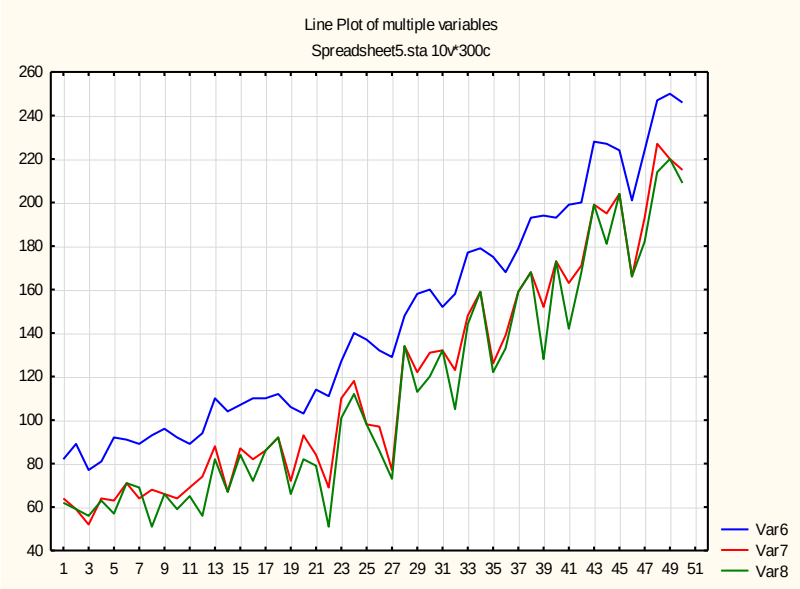


Рис. 4.4.8. График зависимости числа запросов от количества фактов в релевантном выводе с пропорциональным подсчетом релевантности (зеленая линия), в релевантном выводе (красная линия) и обычном выводе (синяя линия).

Все полученные результаты свидетельствуют о том, что релевантный вывод с пропорциональным подсчетом релевантности дает значительные преимущества перед обычным обратным выводом.

4.5. Исследование выполнения параллельных алгоритмов

Для объективной демонстрации возможностей параллельного релевантного LP-вывода было создано порядка пятисот баз знаний, существенно отличающихся по объему множеств правил. При тестировании получены показатели времени выполнения алгоритмов обычного и параллельного релевантного обратного вывода (таблица А.3 в Приложении А). Максимальное количество потоков также ограничивалось в диапазоне от 1 до 20 с шагом 3. Результаты проведенных экспертиз обработаны в пакете Statistica 6.

4.5.1. Сравнение дисперсий генеральных совокупностей

Для сравнения методов релевантного и параллельного релевантного обратного вывода по результатам экспериментов были найдены дисперсии генеральных совокупностей. Более эффективным считается метод, который обеспечивает наименьшую дисперсию.

Проверим предположение о равенстве дисперсий δ_X^2 и δ_Y^2 с помощью F-критерия Фишера.

Для сравнения рассматриваемых алгоритмов возьмем две выборки, объемы которых $n_1 = n_2 = 50$. Выборки представляют собой данные о времени выполнения алгоритмов обычного релевантного вывода и параллельного вывода (показано на примере с 5 потоками). Вначале определим, обладают ли эти методы одинаковой эффективностью $H_0 : \delta_X^2 = \delta_Y^2$ при уровне значимости $\alpha = 0.01$, если в качестве конкурирующей гипотезы принять предположение о том, что эффективность метода параллельного вывода выше, то есть $H_1 : \delta_X^2 > \delta_Y^2$.

Найдем выборочные дисперсии и исправленные выборочные дисперсии генеральных совокупностей данных.

$$S_u^2 = \frac{\sum u_i^2 - \left|\sum u_i\right|^2 / n_1}{n_1 - 1} = 2.47 \quad \text{и} \quad S_v^2 = \frac{\sum v_i^2 - \left|\sum v_i\right|^2 / n_2}{n_2 - 1} = 1.59 \ .$$

Сравним дисперсии, вычислив их отношение по формуле $F_{набл} = \frac{S_X^2}{S_Y^2} = \frac{S_u^2}{S_v^2} = 1.553.$

По условию конкурирующая гипотеза имеет вид $\delta_X^2 > \delta_Y^2$, поэтому критическая область односторонняя, и при отыскании критической точки следует брать уровни значимости равными $\alpha = 0.0$. При числе степеней свободы $v_1 = n_1 - 1 = 49$, $v_2 = n_2 - 1 = 49$ находим критическую точку $F_{кр} = 1.295.$

Так как $F_{набл} > F_{кр}$, то гипотезу о равенстве дисперсий отвергаем – дисперсии различаются значимо. Следовательно, имеет место альтернативная гипотеза о том, что параллельный релевантный метод обратного вывода обеспечивают большую эффективность по времени выполнения.

4.5.2. Сравнение средних генеральных совокупностей

Также был применен метод сравнения средних значений двух нормальных генеральных совокупностей. Для решения этой задачи в случае распределений, близких к нормальному, используется t-тест Стьюдента. При том же уровне значимости 0,01 установим, значимо или незначимо различаются результаты исследований, в предположении, что они распределены нормально.

Проверяется гипотеза $H_0 : a_1 = a_2$ при альтернативной гипотезе $H_1 : a_1 > a_2$. Вычислим оценки средних и дисперсий: $\overline{x_1} = 2.28; \overline{x_2} = 1.85;$ $s_1^2 = 2.47;$ $s_2^2 = 1.59$.

Предварительно проверим гипотезу о равенстве дисперсий $H_0 : \delta_1^2 = \delta_2^2 : \frac{s_1^2}{s_2^2} = 1.553$. Поскольку $F_{кр} = 1.295$, то гипотеза о равенстве дисперсий отклоняется. Вычислим выборочное значение статистики критерия: $t = 1.49$. Число степеней свободы $k = 98$.

Так как по таблице критических точек распределения Стьюдента $t_{кр} = 0.992$, гипотеза о равенстве средних отклоняется и принимается альтернативная гипотеза. Другими словами, средние результаты измерений различаются значимо.

Предложенные алгоритмы также весьма устойчивы (при больших объемах выборок) по отношению к отклонению от нормального распределения [50].

4.5.3. Исследования в пакете Statistica 6

В процессе исследования в пакете Statistica были проведены однофакторный и двухфакторный дисперсионные анализы. Проверка однородности групповых дисперсий выполнена с помощью теста Левена, результаты которого приведены на рисунке.

Levene's Test for Homogeneity of Variances (Spreadsheet7.sta)
Effect: Потоки
Degrees of freedom for all F's: 5, 294

	MS Effect	MS Error	F	p
Время	3,228410	0,609062	5,300625	0,000111

Рис. 4.5.3.1. Применения теста Левена

Результаты показывают, что дисперсии не различаются (P<<0,05). Следовательно, одно из условий применения параметрического варианта дисперсионного анализа не выполнено. Однако этот метод весьма устойчив в случае отклонения от указанного требования, поэтому можно продолжить исследование [62].

Для проверки нормальности распределения анализируемых данных были построены графики нормальных вероятностей и гистограммы, приведенные на рисунке ниже.

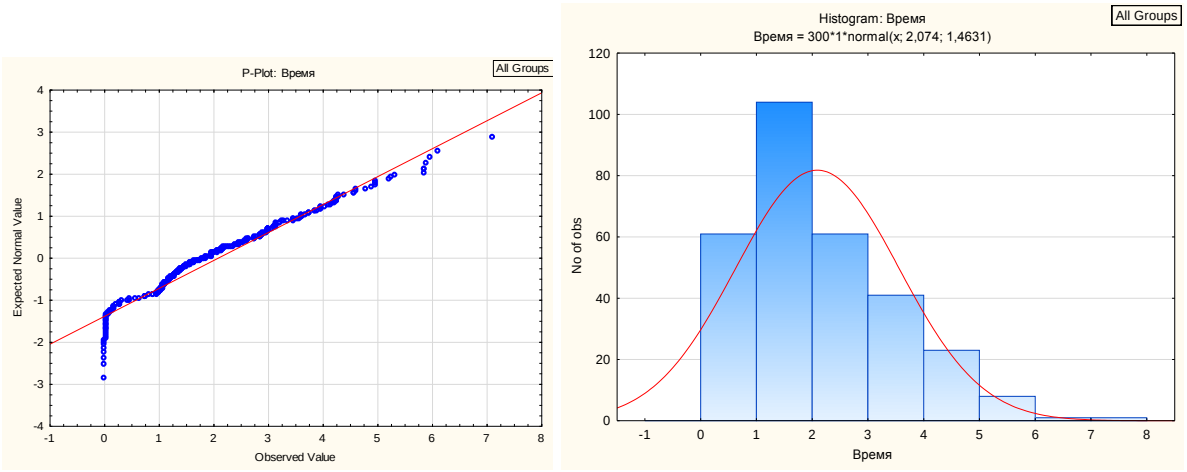


Рис. 4.5.3.2. График нормальных вероятностей и гистограмма

По рисункам видно, что распределение близко к нормальному. Кроме того, объемы выборок позволяют продолжить исследование. Результат дисперсионного анализа представлен на следующем рисунке.

Univariate Tests of Significance for Время (Spreadsheet7.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1290,503	1	1290,503	632,6348	0,000000
Потоки	40,327	5	8,065	3,9538	0,001742
Error	599,727	294	2,040		

Рис. 4.5.3.3. Результат дисперсионного анализа

Поскольку величина ошибки для нулевой гипотезы об отсутствии связи между количеством потоков и временем выполнения $P << 0{,}05$, можно заключить, что время статистически значимо различается в зависимости от числа потоков.

Апостериорный анализ с помощью метода Тьюки показал, что статистически значимая разница во времени выполнения существует между парами параллельного релевантного вывода с 5 потоками и 15 потоками, а также параллельного релевантного вывода с 8 и 15 потоками.

Tukey HSD test; variable Время (Spreadsheet7.sta)
Approximate Probabilities for Post Hoc Tests
Error: Between MS = 2,0399, df = 294,00

Cell No.	Потоки	{1} 2,2824	{2} 2,0739	{3} 1,8466	{4} 1,4901	{5} 2,0687	{6} 2,6826
1	1		0,978298	0,647656	0,061704	0,975810	0,726543
2	2	0,978298		0,968346	0,317385	1,000000	0,271218
3	5	0,647656	0,968346		0,812921	0,971366	0,040124
4	8	0,061704	0,317385	0,812921		0,327529	0,000440
5	10	0,975810	1,000000	0,971366	0,327529		0,262113
6	15	0,726543	0,271218	0,040124	0,000440	0,262113	

Рис. 4.5.3.4. Результаты анализа с помощью метода Тьюки

Двухфакторный дисперсионный анализ выявил значительное влияние числа потоков и количества фактов, а также их взаимное воздействие, на время выполнения ($P << 0{,}05$).

Univariate Tests of Significance for Время (Spreadsheet7.sta)
Sigma-restricted parameterization
Effective hypothesis decomposition

Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1290,503	1	1290,503	12592,53	0,00
Факты	550,367	9	61,152	596,71	0,00
Потоки	40,327	5	8,065	78,70	0,00
Факты*Потоки	24,764	45	0,550	5,37	0,00
Error	24,596	240	0,102		

Рис. 4.5.3.5. Результаты двухфакторного дисперсионного анализа

Непараметрический однофакторный *дисперсионный анализ Краскела-Уоллиса* продемонстрировал показал наличие статистически значимых различий между сравниваемыми группами (P<<0,05).

Kruskal-Wallis ANOVA by Ranks; Время (Spreadsheet7.sta)
Independent (grouping) variable: Потоки
Kruskal-Wallis test: H (5, N= 300) =14,42252 p =,0131

Depend.: Время	Code	Valid N	Sum of Ranks	Mean Rank
1	1	50	8124,500	162,4900
2	2	50	7609,000	152,1800
5	5	50	6967,000	139,3400
8	8	50	5860,500	117,2100
10	10	50	7664,500	153,2900
15	15	50	8924,500	178,4900

Рис. 4.5.3.6. Результаты анализа Краскела-Уоллиса

Для наглядности построены также диаграмма размаха и графики зависимости времени выполнения алгоритма от числа потоков.

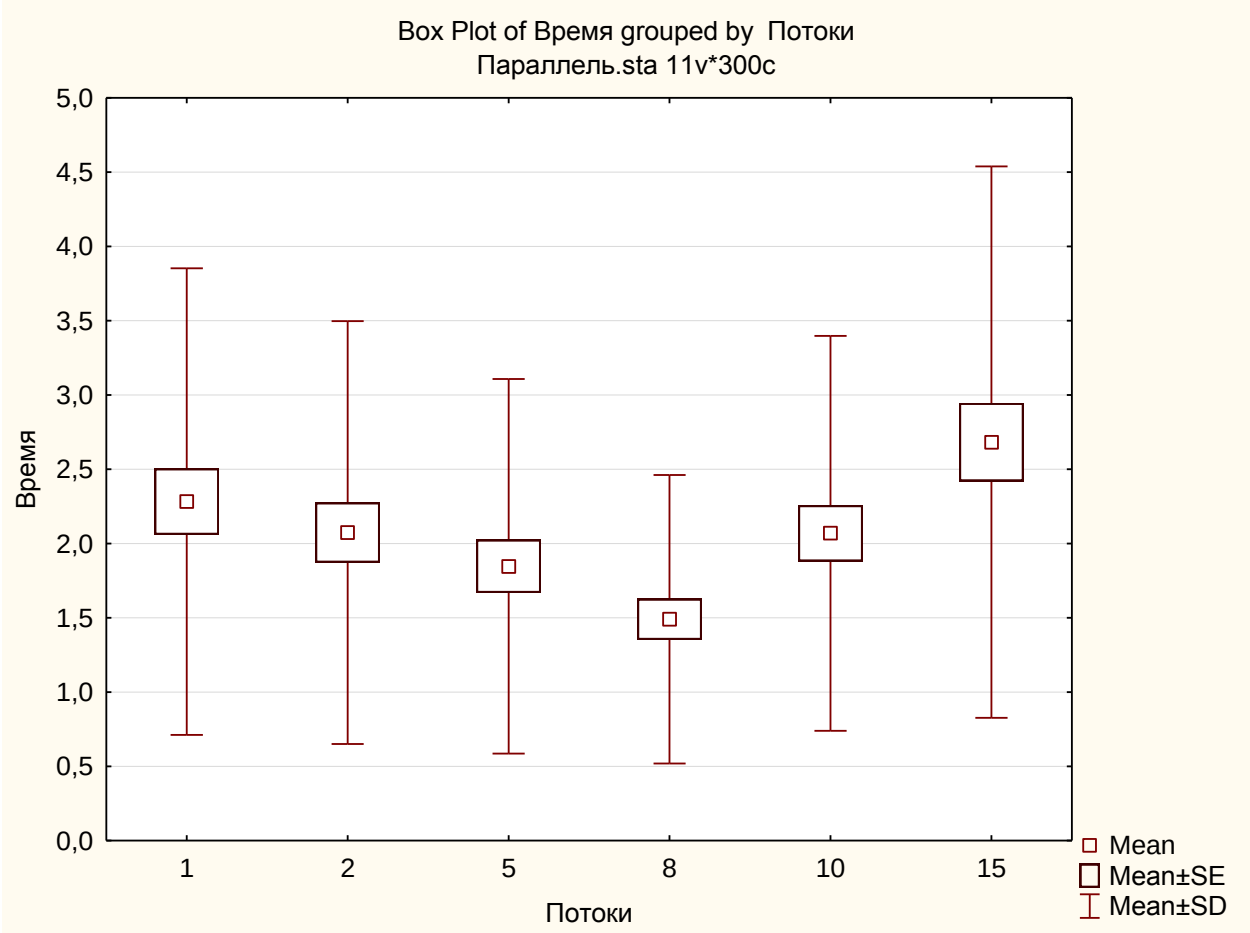


Рис. 4.5.3.7. Диаграммы размаха

По диаграмме размаха видны различия между средними. Можно заключить, что наиболее высокую эффективность по времени дает

использование 8 потоков. Продолжение увеличения числа потоков снижает эффективность, увеличивая время выполнения.

Аналогичный вывод можно сделать и по графику зависимости времени выполнения алгоритма от числа потоков. На оси X отмечено количество правил в базах знаний, на оси Y – время выполнения в секундах, а цветы кривых отражают количество потоков при параллельной реализации алгоритма.

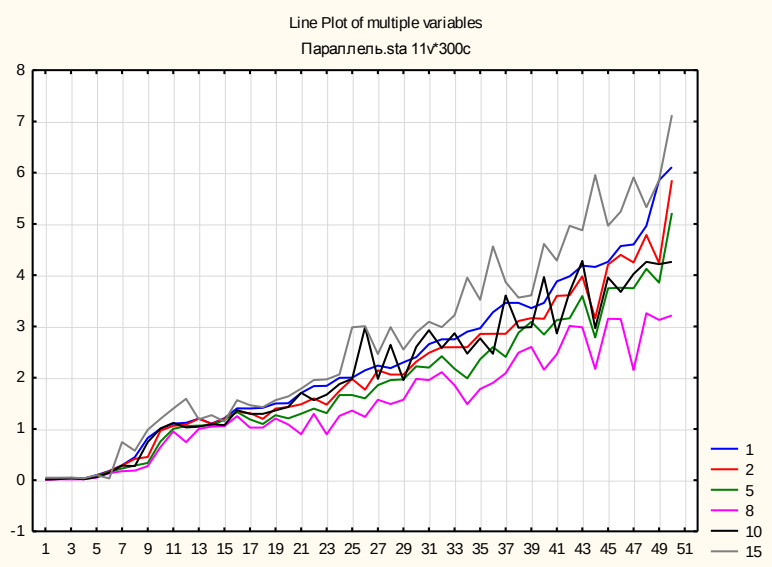


Рис. 4.5.3.8. График зависимости времени вычисления от числа потоков в параллельном релевантном выводе

Для случая параллельного релевантного вывода с 8 потоками и релевантного вывода применен t-критерий, результаты которого говорят о наличии статистически значимых различий между средними (P<<0,05).

T-tests; Grouping: Потоки (Параллель.sta)
Group 1: 1
Group 2: 8

Variable	Mean 1	Mean 8	t-value	df	p	Valid N 1	Valid N 8	Std.Dev. 1	Std.Dev. 8	F-ratio Variances	p Variances
Время	2,282412	1,490060	3,034365	98	0,003086	50	50	1,570445	0,971105	2,615248	0,001014

Рис. 4.5.3.9. Результат применения t-критерия

Все продемонстрированные результаты свидетельствуют об уменьшении времени нахождения решения в случае использования многопоточности на 20–25% в среднем. В данной ситуации речь идет о статистическом показателе, поскольку и при обычном выводе имеется некоторая вероятность случайного достижения лучшего результата «с первого раза».

В результате экспериментов было также установлено максимальное количество потоков выполнения (threads), дающее прирост в эффективности на компьютере имеющейся конфигурации (Intel core i7 − 870 (2.93 GHz 8 Mb L3 cache) 4 ядра, 8 потоков, 12 GB 1333MHz DDR3). Анализ результирующих таблиц показывает, что оптимальное количество потоков примерно равно 8. При превышении этого значения наблюдается снижение эффективности.

4.6. Выводы

С целью обоснования эффективности представленных в работе алгоритмов релевантного вывода проведено порядка пятисот тестов со случайными базами знаний, существенно отличающимися по объему фактов и правил, а также по глубине их вложенности. Результаты тестов обработаны статистическими методами и получены следующие выводы.

- Релевантный вывод сокращает число запросов к внешнему источнику в среднем на 15−20% по сравнению с обычным обратным выводом.
- Модификация релевантного вывода с пропорциональным подсчетом релевантности также обеспечивает прирост в эффективности, заключающийся в дальнейшем уменьшении числа запросов.
- Кластерно-релевантный вывод дает хорошие результаты по сравнению с обычным обратным выводом в случае работы с базами данных больших размеров, для которых релевантный вывод не может быть применен из-за ограниченности ресурсов компьютера.
- Параллельная реализация релевантного LP-вывода позволяет сократить время выполнения рассматриваемых процессов в среднем на 20%. Кроме того, по полученным графикам установлено оптимальное количество потоков, дающее наибольший прирост к эффективности для текущей конфигурации компьютера.

Сделанные выводы наглядно подтверждаются построенными графиками и диаграммами.

Заключение

Настоящая диссертационная работа представляет собой исследование в области формальных знаний – логических систем продукционного типа, составляющих основу многих моделей в информатике. Его основная задача – разработка и изучение метода релевантного LP-вывода, с целью развития его идей до работающей на практике новой технологии ускорения логического вывода и верификации знаний, а также обоснование преимуществ этой технологии в сравнении с другими подходами к аналогичным задачам. В рамках проведенного исследования получены следующие основные результаты.

- Сформулирована и исследована обобщенная модель LP-структуры, расширяющая область применения теории LP-структур.
- Усовершенствована схема процесса решения продукционно-логического уравнения. В результате из этого процесса исключен избыточный этап, связанный с идентификацией приближенных решений.
- Разработан новый метод обратного вывода в логических системах продукционного типа, обеспечивающий снижение числа обращений к внешним устройствам.
- Предложены и апробированы различные способы выбора параметров релевантности для процессов релевантного и кластерно-релевантного LP-вывода, предоставляющие гибкие возможности управления обратным выводом.
- Разработаны параллельные алгоритмы релевантного и кластерно-релевантного LP-вывода, а также параллельные алгоритмы вычисления истинных прообразов в LP-структурах, повышающие быстродействие LP-вывода.
- Создана новая версия интегрированного программного пакета LPExpert, предназначенного для разработки и эксплуатации продукционно-логических систем. Она реализует все перечисленные выше алгоритмы, параллельные

вычисления и стратегии релевантности, а также поддерживает 64-разрядную архитектуру процессора.

- Выполнены массированные вычислительные эксперименты и проведено подробное исследование полученных результатов методами математической статистики. В результате формально обоснованы преимущества нового метода обратного вывода в различных его модификациях.

Дальнейшие исследования на рассматриваемом направлении могут быть связаны с выбором LP-структур более сложных типов и соответствующих им предметных областей, перенос концепций настоящей работы на эти модели. Например, при использовании в качестве решетки алгебры Линденбаума-Тарского [3, 95] моделируемые условные правила смогут в качестве предпосылок и заключений содержать формулы пропозиционального исчисления. Таким образом, можно будет рассматривать расширенные модели логических систем и решать для них задачи ускорения и распараллеливания процессов обратного вывода с общей стратегией снижения запросов к внешним источникам информации. Общие методы исследования при этом останутся прежними.

Приложение А. Таблицы результатов тестов

А.1. Тесты релевантного вывода

№ теста	Количество фактов	Количество правил	Задано вопросов в обычном выводе	Задано вопросов в LP-выводе
1	5	20	5	4
2	5	50	5	5
3	6	20	6	4
4	6	50	5	4
5	7	20	8	7
6	7	50	7	5
7	8	20	9	7
8	8	50	8	3
9	9	20	12	9
10	9	50	16	7
11	10	50	15	7
12	10	70	14	10
13	10	90	10	7
14	11	50	11	10
15	11	70	12	10
16	11	90	11	9
17	12	50	14	8
18	12	70	15	6
19	12	90	16	12
20	13	50	15	8
21	13	70	15	13
22	13	90	18	7
23	14	50	17	9
24	14	70	13	12
25	14	90	16	14
26	15	50	12	7
27	15	70	14	13
28	15	90	15	13
29	20	50	23	18
30	20	70	24	16
31	20	90	27	19
32	25	50	25	23
33	25	70	23	22
34	25	90	25	25
35	30	50	36	30
36	30	70	30	27
37	30	90	34	19

38	30	120	35	30
39	30	140	33	28
40	35	70	37	34
41	35	80	38	31
42	35	90	42	35
43	35	100	42	25
44	35	110	39	25
45	40	70	46	37
46	40	80	43	34
47	40	90	46	24
48	40	100	48	42
49	50	70	50	38
50	50	80	51	41
51	50	90	63	50
52	50	100	55	44
53	50	120	57	49
54	55	70	43	41
55	55	80	57	53
56	55	90	59	48
57	55	100	52	42
58	56	80	56	56
59	57	90	58	49
60	58	90	58	51
61	59	90	59	53
62	60	100	48	31
63	60	120	63	54
64	60	130	65	32
65	65	100	69	53
66	65	120	37	18
67	65	130	74	65
68	65	150	68	61
69	65	170	69	46
70	70	100	73	61
71	70	120	76	47
72	70	130	74	64
73	70	150	71	65
74	70	170	79	64
75	75	180	76	71
76	75	120	73	73
77	75	130	79	63
78	75	150	82	75
79	75	170	85	71
80	80	100	89	52
81	80	120	83	54
82	80	130	86	80
83	85	150	88	64
84	85	120	81	75
85	85	130	93	84
86	85	150	91	75
87	85	170	89	76
88	90	120	98	81

89	90	150	93	84
90	90	170	91	83
91	90	190	78	61
92	100	130	103	89
93	100	140	125	103
94	100	170	111	94
95	100	160	107	91
96	100	170	91	80
97	105	130	95	82
98	105	140	103	101
99	110	150	101	89
100	110	170	121	112

A.2. Тесты кластерно-релевантного вывода

Номер теста	**Кол-во фактов**	**Кол-во правил**	**Число запросов в обычном выводе**	**Число запросов в релевант-ном выводе**	**Количество запросов в кластерно-релевантном выводе (Количество прообразов)**				
					1	**10**	**50**	**100**	**200**
1	90	100	86	57	72	69	65	61	61
2	90	120	82	52	80	78	78	75	66
3	90	130	84	61	74	70	70	69	63
4	95	100	85	62	83	78	78	75	75
5	95	120	69	67	79	79	79	79	79
6	95	130	95	62	79	76	76	68	68
7	95	150	89	59	85	83	83	77	74
8	95	170	95	63	81	76	76	70	70
9	100	120	93	61	83	81	77	64	60
10	100	150	99	58	87	85	85	76	69
11	100	170	98	69	90	81	81	69	69
12	100	190	91	58	85	79	79	71	71
13	110	130	104	66	93	88	88	81	73
14	110	140	107	77	94	86	86	83	77
15	110	150	110	67	106	96	92	78	67
16	110	160	104	73	91	90	90	90	84
17	110	170	107	82	100	91	91	87	87
18	115	130	113	78	98	94	94	81	78
19	115	140	112	88	101	96	96	96	96
20	120	150	120	72	94	84	81	81	73
21	120	170	118	64	96	93	93	83	71
22	120	200	106	81	102	99	92	92	89
23	140	200	136	92	106	101	101	97	92
24	140	220	129	77	121	102	84	84	84
25	140	230	127	79	117	105	98	98	98
26	140	250	138	86	130	116	116	105	86
27	140	300	140	92	111	101	97	97	92

28	160	200	153	99	145	133	121	111	103
29	160	220	148	126	144	138	138	138	138
30	160	230	159	125	153	139	132	128	126
31	160	250	160	113	151	144	144	139	113
32	160	300	153	128	146	144	144	133	133
33	180	220	170	142	154	149	149	149	146
34	180	230	179	127	162	159	152	151	151
35	180	250	172	147	164	164	164	164	164
36	180	300	173	124	148	138	138	129	129
37	180	350	179	127	173	162	162	147	127
38	200	250	193	145	174	163	145	145	145
39	200	300	192	147	190	194	179	179	168
40	200	330	199	153	184	179	179	162	162
41	200	350	200	167	185	178	178	178	174
42	200	370	182	175	194	188	188	183	183
43	230	300	199	120	199	172	161	161	146
44	230	330	214	159	191	191	191	191	191
45	230	350	221	164	213	201	201	201	181
46	230	370	206	183	206	203	203	187	183
47	230	400	221	173	221	214	211	211	204
48	250	350	234	195	221	217	215	215	203
49	250	400	243	211	237	231	231	229	216
50	250	500	231	174	231	204	204	188	185

A.3. Тесты пропорционального подсчета релевантности

№ теста	Количество фактов	Количество правил	Задано вопросов в обычном выводе	Задано вопросов в LP-выводе с линейным подсчетом релевантности	Задано вопросов в LP-выводе с пропорциональным подсчетом релевантности
1	90	100	82	64	62
2	90	120	89	59	59
3	90	130	77	52	56
4	95	100	81	64	63
5	95	120	92	63	57
6	95	130	91	71	71
7	95	150	89	64	69
8	95	170	93	68	51
9	100	120	96	66	66
10	100	150	92	64	59
11	100	170	89	69	65
12	100	190	94	74	56
13	110	130	110	88	82
14	110	140	104	67	67
15	110	150	107	87	84
16	110	160	110	82	72
17	110	170	110	86	86

18	115	130	112	92	92
19	115	140	106	72	66
20	120	150	103	93	82
21	120	170	114	84	79
22	120	200	111	69	51
23	140	200	127	110	101
24	140	220	140	118	112
25	140	230	137	98	98
26	140	250	132	97	86
27	140	300	129	77	73
28	160	200	148	134	134
29	160	220	158	122	113
30	160	230	160	131	120
31	160	250	152	132	132
32	160	300	158	123	105
33	180	220	177	148	144
34	180	230	179	159	159
35	180	250	175	126	122
36	180	300	168	139	133
37	180	350	179	159	159
38	200	250	193	168	168
39	200	300	194	152	128
40	200	330	193	173	173
41	200	350	199	163	142
42	200	370	200	171	168
43	230	300	228	199	199
44	230	330	227	195	181
45	230	350	224	204	204
46	230	370	201	166	166
47	230	400	224	193	182
48	250	350	247	227	214
49	250	400	250	220	220
50	250	500	246	215	209

A.4. Тесты параллельных алгоритмов

Номер теста	**Количество правил**	**Время нахождения решения в секундах**					
		В релевантном выводе	**В параллельном релевантном выводе**				
			2 потока	**5 потоков**	**8 потоков**	**10 потоков**	**15 потоков**
1	50	0.0172	0.0142	0.0118	0.0086	0.0153	0.0464
2	50	0.0268	0.0248	0.0218	0.0174	0.0214	0.0472
3	50	0.0441	0.0399	0.0264	0.0158	0.0385	0.0498
4	50	0.0302	0.0296	0.0258	0.0197	0.0184	0.0358
5	50	0.0956	0.0843	0.0742	0.0496	0.0596	0.0964
6	75	0.1818	0.1732	0.1633	0.1354	0.1478	0.0348
7	75	0.2912	0.2843	0.2274	0.1742	0.2853	0.7433

8	75	0.4498	0.4151	0.2853	0.1854	0.2734	0.5754
9	75	0.8237	0.4536	0.3348	0.2745	0.7434	0.9843
10	75	1.0023	0.9654	0.7545	0.6434	1.0085	1.1963
11	100	1.1034	1.0589	1.0012	0.9523	1.1154	1.3956
12	100	1.1204	1.0863	1.0542	0.7432	1.0285	1.5864
13	100	1.2011	1.1999	1.0654	1.0001	1.0432	1.1875
14	100	1.1032	1.0965	1.0732	1.0454	1.0925	1.2694
15	100	1.2074	1.1975	1.1767	1.0544	1.0732	1.1453
16	125	1.4016	1.3290	1.3296	1.2496	1.3659	1.5603
17	125	1.3994	1.3087	1.1854	1.0247	1.2953	1.4609
18	125	1.4132	1.1956	1.0965	1.0247	1.2953	1.4263
19	125	1.4965	1.3954	1.2696	1.2017	1.3601	1.5603
20	125	1.5001	1.4276	1.2065	1.0843	1.4307	1.6335
21	150	1.7023	1.4803	1.2959	0.8976	1.7053	1.7865
22	150	1.8354	1.5953	1.3955	1.2953	1.5603	1.9532
23	150	1.8403	1.4724	1.3064	0.8953	1.6694	1.9654
24	150	1.9986	1.7403	1.6594	1.2548	1.8743	2.0635
25	150	2.0013	1.9754	1.6596	1.3585	1.9786	2.9846
26	175	2.1437	1.7653	1.5962	1.2349	2.9864	3.0053
27	175	2.2364	2.1437	1.8534	1.5698	1.9756	2.4596
28	175	2.1853	2.0613	1.9534	1.4859	2.6418	2.9864
29	175	2.2985	2.0617	1.9652	1.5697	1.9543	2.5485
30	175	2.4001	2.3106	2.2196	1.9765	2.5983	2.8797
31	200	2.6539	2.4842	2.1955	1.9543	2.9254	3.0915
32	200	2.7484	2.5904	2.4193	2.1064	2.5797	2.9864
33	200	2.7494	2.5948	2.1740	1.8545	2.8648	3.2195
34	200	2.8965	2.5935	1.9875	1.4832	2.4692	3.9521
35	200	2.9651	2.8541	2.3591	1.7786	2.7654	3.5193
36	225	3.2754	2.8543	2.5967	1.8995	2.4657	4.5607
37	225	3.4584	2.8547	2.4063	2.0862	3.6043	3.8659
38	225	3.4592	3.1063	2.8754	2.4859	2.9764	3.5607
39	225	3.3569	3.1603	3.0896	2.5982	2.9864	3.6064
40	225	3.4592	3.1492	2.8434	2.1549	3.9643	4.6096
41	250	3.8769	3.5963	3.1247	2.4549	2.8643	4.2847
42	250	3.9765	3.6095	3.1593	3.0096	3.6789	4.9624
43	250	4.1852	3.9783	3.5923	2.9836	4.2769	4.8756
44	250	4.1594	3.1593	2.7854	2.1694	2.9653	5.9522
45	250	4.2583	4.2064	3.7442	3.1482	3.9575	4.9644
46	275	4.5692	4.3952	3.7543	3.1428	3.6732	5.2397
47	275	4.5985	4.2483	3.7432	2.1484	4.0251	5.9062
48	275	4.9623	4.7845	4.1245	3.2594	4.2604	5.3277
49	275	5.8524	4.2360	3.8532	3.1274	4.2160	5.8532
50	275	6.1086	5.8532	5.2148	3.2149	4.2603	7.1235

Приложение В. Некоторые тексты программ

В.1. Модули класса ParallelLPStructure

```
#pragma mark – Threaded

// Функция обратного вызова
void minPreImageCallback (const ElemContainer eRes, const ElemItem pbRes)
{
   // аккумулируем результаты работы
   for (ElemContainer::iterator i = shared_ecRes->begin(); i != shared_ecRes->end(); )
   {
      if (LE(*i, eRes))  { isBad = TRUE; break; } // Новый - лишний
      if (LT(eRes, *i)) i = shared_ecRes->erase(i); // Удалить худшие
      else i++;
   }
   if (!isBad)
   {
      PElemItem newRes = new ElemItem[eLengthItems];
      CopyMemory(newRes, pbRes, eLengthBytes);
      shared_ecRes->insert((Elem)newRes);
   }
   if (wasOver)
  {
      lpEvent (pbRes,currentUser);
   }
}
// контрольная функция. Выполняется в главном потоке
void threadedMinPreImages (const Elem eDest, const Elem aNegative,
                                    const INT aMaxCount,
                                    const OnParallelLPEvent onEvent,
                                    const DWORD dwUser,
```

```
                                const int nItem,
                                const int nBit,
                                ElemContainer *ecRes)
{
/***  Выделить все элементы, слева связанные с целевым  ***/
   lpEvent = onEvent;
   currentUser = user;
   LPThreadPool * threadPool = LPThreadPool::getInstance(); // общий инстанс объекта пула потоков
   PElemItem pbDest = new ElemItem[eLengthItems]; ZeroMemory(pbDest, eLengthBytes);
   ElemItem nMask = 1 << (eItemBitSize - 1 - nBit);
   pbDest[nItem] |= nMask;
   eDest = (Elem)pbDest;
   PElemItem pbConnected = new ElemItem[eLengthItems];
   getLConnected(eDest, (Elem)pbConnected);
   CopyMemory(shared_ecRes, ecRes, eLengthBytes);
   /***  Обратный логический вывод в каждом связанном слое  ***/
   ElemContainer::iterator *iLayer = new ElemContainer::iterator[nAtomCount]; // Итераторы слоев
   // Создаем итераторы слоев
   for (int i = 0, cc = ecLVector.size(); i < cc ; i++)
   {
      if (ecLVector[i])
      {
                  iLayer[i] = ecLVector[i]->begin();
      }
   }
   wasOver = FALSE;
   //инициализируем критическую секцию перед запуском потоков
   InitializeCriticalSection(&thread_criticalSection);
   while (!wasOver)
   {
      //запускаем minPreImageInParallel в отдельном потоке и проверяем окончание выполнения
      Thread_Arguments thread_args;
      thread_args.elem = eDest;
```

```
        thread_args.layer = iLayer;
        thread_args.callback = &minPreImageCallback;
        if (!threadPool.AddTask((LPTHREAD_START_ROUTINE)&minPreImageInParallel,
(LPVOID) &thread_args))
        {
           printf ("Error Adding task to thread pool");
        }
        if (!wasOver)
        {
           wasOver = TRUE;
           for (int i = ecLVector.size(); --i >=0 && wasOver; )
           {
              if (!ecLVector[i] || !isON((Elem)pbConnected, i) ) continue;
              if (++iLayer[i] != ecLVector[i]->end() ) wasOver = FALSE;
              else iLayer[i] = ecLVector[i]->begin();
           }
        }
    }
}
// выполняется во второстепенном потоке
void minPreImageInParallel (LPVoid param_list)
{
    Thread_Arguments args = (Thread_Arguments)param_list;
    Elem eDest = args.elem;
    ElemContainer *iLayer = args.layer;
    CallbackEventcallback = args.callback;
    // Память для прообразов - результатов логического вывода
    PElemItem pbRes = new ElemItem[eLengthItems];
    Elem eRes = (Elem)pbRes;
    PElemItem pbApplied = new ElemItem[eLengthItems];
    CopyMemory(pbRes, (PVOID)eDest, eLengthBytes);
    ZeroMemory(pbApplied, eLengthBytes);
    BOOL res = LE(eRes, eBegs), wasApplied = TRUE;

    int thread_eLengthItems;
    int thread_eItemBitSize;
```

```
   ElemContainerVector thread_ecLVector;
   EnterCriticalSection(&thread_criticalSection);
   thread_eLengthItems = this.eLengthItems;
   thread_eItemBitSize = this.eItemBitSize;
   thread_ecLVector    = this.ecLVector;
   LeaveCriticalSection(&thread_criticalSection);
   BOOL isOver = FALSE;
   while (!res && wasApplied)
   {
      wasApplied = FALSE;
      /* Просмотр разрядов eRes слева направо */
      BOOL isBreak = FALSE;
      for (int nItem = 0; (nItem < thread_eLengthItems) && !isBreak; nItem++) //  &&
!wasCycle
      {
         if (!pbRes[nItem]) continue; // Элемент кода не содержит единиц
         ElemItem nMask = 1 << (thread_eItemBitSize - 1); // Маска для тестирования
разрядов
         for (int nBit = 0; nBit < thread_eItemBitSize; nBit++)
         {
            if (pbRes[nItem] & nMask)
            {
               int iLV = nItem * thread_eItemBitSize + nBit;
        if (!ecLVector[iLV]) { nMask >>= 1; continue; } // Для данного эл-та нет предпосылок
               // Применить предпосылку для данного эл-та
               pbRes[nItem] &= ~nMask; // Очистить "заключение"
               if ( !(pbApplied[nItem] & nMask) )
               { // Она ранее не была применена ранее
                  lJoin(eRes, *iLayer[iLV]); // (eRes) |= (*iLayer[iLV]);
                  pbApplied[nItem] |= nMask; // isApplied[iLV] = TRUE;
                  wasApplied = TRUE;
               }
               if ( LE(eRes, eBegs) ) { res = TRUE; break; } // eRes <= eBegs - все атомы
результата начальные
            }
            nMask >>= 1; // Переход к очередному биту в eRes
```

```
        }
      }
    }
    if (callback) callback (eRes,pbRes);
}
```

B.2. Подсчет релевантности

```
// Нахождение наименьшей мощности в актуальных прообразах
procedure findMinPower (nDestLegals:integer);
var
  i,cc,j:integer;
  wMinPower: TValIndex;
begin
  wMinPower := $8FFF;
  for i := 0 to nDestLegals - 1 do
  begin
    cc := Length(lpsPreImages[i]) - 1;
    for j := 0 to cc do
    begin
      if lpsPreImages[i, j].lpsState = stNotActual then continue;
      if lpsPreImages[i, j].lpsPower < wMinPower then
        wMinPower := lpsPreImages[i, j].lpsPower
    end
  end;
end;

// Подсчет релевантности
procedure findRelevance (flagMaxPreImages,flagMinPower:boolean;
                         nDestLegals:integer; ObjectsRel:TRelevantArray;);
var
  i,cc,j, nItem, nBit:integer;
  lpsCurrCode: PlpsCode;  // Указатель кода LP-структуры
  nMask: TlpsCodeItem;
  nCurrValue: TValIndex;
```

```
  CurrValPtr: ^TLegalValue;
  CurrObjPtr: ^TExpObject;
begin
  {**  Просмотр LP-кодов  **}
  for i := 0 to nDestLegals - 1 do
  begin
    cc := Length(lpsPreImages[i]) - 1;
    for j := 0 to cc do
    begin
      if lpsPreImages[i, j].lpsState = stNotActual then continue;
      {**  Просмотр LP-кода очередного прообраза  **}
      lpsCurrCode := lpsPreImages[i, j].lpsCode;
      dec(lpsCurrCode); // За левую границу кода
      for nItem := 0 to nLPSCodeSize - 1 do
      begin
        inc(lpsCurrCode); // Переход к очередному кластеру кода
        if lpsCurrCode^ = 0 then continue; // Кластер кода не содержит единиц
        // Просмотр кластера LP-кода
        nMask := 1 shl (LPS_CODEITEM_SIZE - 1); // Маска для тестирования разрядов
        for nBit := 0 to LPS_CODEITEM_SIZE - 1 do
        begin
          if (lpsCurrCode^ and nMask) > 0 then
          begin // Обнаружен установленный бит
            nCurrValue := nItem * LPS_CODEITEM_SIZE + nBit;
            CurrValPtr := ValueList[nCurrValue];
            CurrObjPtr := ObjectList.Items[CurrValPtr^.nObject];
            // Увеличить релевантность нерешенного начального объекта
            if not (otSought in CurrObjPtr^.ObjType) then
            begin
              if (flagMaxPreImages) then Inc(ObjectsRel[CurrValPtr^.nObject]);
              if (flagMinPower) and lpsPreImages[i, j].lpsPower = wMinPower then
                Inc(ObjectsRel[CurrValPtr^.nObject])
            end
          end;
          nMask := nMask shr 1 // Переход к очередному биту
        end
```

```
      end
    end
  end
end;

 // Высчитать релевантность пропорциональным способом
procedure findModifiedRelevance (first, second: boolean;
                                    nDestLegals:integer; ObjectsRel:TRelevantArray;);
var
  i,cc,j, k, l:integer;
  CurrValPtr: ^TLegalValue;
begin
  k := 0; l := 0;
  {**  Просмотр LP-кодов  **}
  for i := 0 to nDestLegals - 1 do
  begin
    cc := Length(lpsPreImages[i]) - 1;
    for j := 0 to cc do
    begin
      if lpsPreImages[i, j].lpsState = stNotActual then continue;
      if (first) then ObjectsRel[CurrValPtr^.nObject] := ObjectsRel[CurrValPtr^.nObject] +
                                                        getRelevanceInMaxCount ();
      if (second) then
      begin
        SortByPower(lpsPreImages); // Сортируем прообразы по мощности
        while (k <= nDestLegals - 1) do
        while (l <= cc) do
        begin
          if lpsPreImages[i, j].lpsPower = preImagesPower[k,l].power then
            ObjectsRel[CurrValPtr^.nObject] := ObjectsRel[CurrValPtr^.nObject] +
                                                  preImagesPower[k,l].power;
          Inc(k); Inc(l);
        end
      end
    end;
  end;
```

```
end;

// Найти индекс наиболее релевантного объекта
function mostRelevantIndex(ObjectsRel: TRelevantArray; nObjs: TObjIndex): integer;
var
  nMaxRel: DWORD;
  i:integer;
begin
  // Найти индекс наиболее релевантного объекта
  nMaxRel := 0; result := arNot;
  for i := 0 to nObjs - 1 do
  begin
    if ObjectsRel[i] > nMaxRel then
    begin
      nMaxRel := ObjectsRel[i]; result := i
    end
  end;
end;

// Линейный алгоритм подсчета релевантности
procedure getRelevantWithLinearAlgorithm (nDestLegals: integer;
                                          ObjectsRel: TRelevantArray);
var
  cc, i: integer;
  wMinPower: TValIndex;
  nObjs: TObjIndex;
begin
  findMinPower(nDestLegals);
  findRelevance (true, true, nDestLegals, ObjectsRel);
  mostRelevantIndex(ObjectsRel, nObjs);
end;

// Пропорциональный алгоритм подсчета релевантности
procedure getRelevantWithProportionalLinearAlgorithm (nDestLegals: integer;
                                                      ObjectsRel: TRelevantArray);
var
```

```
  cc, i: integer;
  wMinPower: TValIndex;
  nObjs: TObjIndex;
begin
  findMinPower(nDestLegals);
  findModifiedRelevance (true, true, nDestLegals, ObjectsRel);
  mostRelevantIndex(ObjectsRel, nObjs);
end;

// Исключение одного из коэффициентов релевантности
procedure getRelevantModifiedAlgorithms (firstPoint, secondPoint:boolean;
                                         nDestLegals: integer; ObjectsRel: TRelevantArray);
var
  cc, i: integer;
  wMinPower: TValIndex;
  nObjs: TObjIndex;
begin
  findMinPower(nDestLegals);
  if (firstPoint) then
    findModifiedRelevance (true, false, nDestLegals, ObjectsRel)
  else
    if secondPoint then
      findModifiedRelevance (false, true, nDestLegals, ObjectsRel);
  mostRelevantIndex(ObjectsRel, nObjs);
end;

 // Выбор стратегии подсчета релевантности
function getRelevant (typeRelevance : WORD; firstPoint:Boolean;
                                            secondPoint:boolean): WORD;
var
  nObjs: TObjIndex;
  ObjectsRel: TRelevantArray;
  nDestLegals: integer;
begin
  nObjs := ObjectList.Count; SetLength(ObjectsRel, nObjs); // Счетчики релевантности
  cc := nObjs - 1;
```

```
  for i := 0 to cc do ObjectsRel[i] := 0;
  nDestLegals := Length(lpsPreImages);
  if (typeRelevance = 0) then
    getRelevantWithLinearAlgorithm (nDestLegals,ObjectsRel)
  else if (typeRelevance = 1) then
    getRelevantWithProportionalLinearAlgorithm (nDestLegals,ObjectsRel)
  else
    if (typeRelevance  < 0) then
      getRelevantModifiedAlgorithms (firstPoint, secondPoint, nDestLegals,ObjectsRel);
end;
```

Литература

Публикации автора по теме диссертации приведены в заключительной части данного раздела отдельным списком.

1. *Agarwal R.* A Petri-Net based approach for verifying the integrity of production systems / R. A. Agarwal // International Journal of Man-Machine Studies. – 1992. – № 36. – P. 447–468.
2. *Aho A. V.* The transitive reduction of a directed graph. SIAM J. Computing 1 : 2 / A. V. Aho, M. R. Garey, J. D. Ulman, 1972. – P. 131–137.
3. *Andréka H.* Algebraic Logic / H. Andréka, J. D. Monk, I. Németi. – Dordrecht : North-Holland Publ. Co, 1991.
4. *Avritzer A.* Estimating the CPU utilization of a rule-based system / A. Avritzer, J. P. Ros, E. J. Weyuker // In Proceedings of the 4th international Workshop on Software and Performance (Redwood Shores, California, January 14–16, 2004). ACM. – New York : NY, 2004. – P. 1–12.
5. *Cheng A. M.* Self-Stabilizing Real-Time OPS5 Production Systems / A. M. Cheng, S. Fujii // IEEE Transactions on Knowledge and Data Engineering. – 2004. – V. 16. – №. 12. – P. 1543–1554.
6. *Cheng A. M.* A Graph-Based Approach for Timing Analysis and Refinement of OPS5 Knowledge-Based Systems / A. M. Cheng, H.-Y. Tsai // IEEE Transactions on Knowledge and Data Engineering. – 2004. – Vol. 16. – № 2. – P. 271–288.
7. *Cook S. A.* The complexity of theorem-proving procedure / S. A. Cook // Proc. 3rd Annual ACM Symposium on the Theory of Computing. – 1971. – P. 151–159.

8. *Cook S. A.* A. The relative efficiency of propositional proof systems / S. A. Cook, R. A. Reckhow // Journal of Symbolic Logic. – 1979. – V. 44, N. 1. – P. 36–50.
9. *Davis R.* An overview of production systems / R. Davis, J. King // Mahine Intelligence. – Chichester : Ellis Horwood Limited. – 1977. – Vol 8. – P. 300–332.
10. *De Baets B.* Analytical Solution Methods for Fuzzy Relational Equations / B. De Baets // Fundamentals of Fuzzy Sets / eds. D. Dubois, and H. Prade. – Boston : Kluwer Academic Publishers, 2000. – P. 291–340.
11. *Doorenbos R. B.* Production Matching for Large Learning Systems. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-22942., Carnegie Mellon University / R. B. Doorenbos, 1995.
12. *Forgy C. L.* OPS5 user's manual. Technical Report CMU-CS-81-135, Computer Science Department, Carnegie Mellon University, 1981.
13. *Forgy C. L.* Rete: A fast algorithm for the many pattern/many object pattern match problem / C. L. Forgy // Artificial Intelligence. – 1982. – № 19(1). – P. 17–37.
14. *Ginsberg A.* Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy / A. Ginsberg // Proceedings of the Seventh National Conference on Artificial Intelligence, 1988. – P. 585–589.
15. *Giraud-Carrier C. G.* An Integrated Framework for Learning and Reasoning / C. G. Giraud-Carrier, T. R. Martinez // Journal of Artificial Intelligence Research. – 1995. – V. 3. – P. 147–185.
16. *Gupta A.* Parallelism in Production Systems / A. Gupta. – Los Altos : Morgan Kaufmann, CA, 1987.
17. *Gupta U. G.* Validation and verification of knowledge-based systems: A survey / U. G. Gupta // Journal of Applied Intelligence. – 1993. – V. 3. – P. 343–363.
18. *Halib M.* Bit-vector encoding for partially ordered sets / M. Halib, L. Nourine // Lect. Notes Comput. Sci. – 1994. – V. 831. – P. 1–12.

19. *Halmos P.* Algebraic logic, IV: Equality in polyadic algebras / P. Halmos // Trans. Amer. Math. Soc. – 1957. – N. 86. – P. 1–27.
20. *Halmos P.* Algebraic Logic / P. Halmos. – New York : Chelsea Publishing Co, 1962.
21. *Hammer P. L.* Optimal compression of propositional Horn knowledge bases: complexity and approximation / P. L. Hammer, A. Kogan // Artif. Intell. – 1993. – V. 64, N. 1. – P. 131–145.
22. *Henkin L.* Cylindric algebras / L. Henkin, A. Tarski // Proc. of Symposia in Pure Mathematics II (1961), Lattice theory. – P. 83–113.
23. *Homeier P. V.* ECLIPS: An extended CLIPS for backward chaining and goal-directed reasoning / P. V. Homeier, T. C. Le // NASA. Johnson Space Center, Second CLIPS Conference Proceedings. – 1991. – Vol. 2. – P. 273–283.
24. *Kadar B.* Approaches to Increase the Performance of Agent-Based Production Systems / B. Kadar, L. Monostori // Proceedings of the 14th international Conference on industrial and Engineering Applications of Artificial intelligence and Expert Systems: Engineering of intelligent Systems (June 04–07, 2001) / eds. L. Monostori, J. Váncza, M. Ali // Lecture Notes In Computer Science. – London : Springer-Verlag. – 2001. – Vol. 2070. – P. 612–621.
25. *Kang J. A.* Shortening Matching Time in OPS5 Production Systems / J. A. Kang, A. M. Cheng // IEEE Transactions on Software Engineering. – July 2004. – Vol. 30, № 7. – P. 448–457.
26. *Lee S.* Developing a strategy for expert system verification and validation / S. Lee, R. M. O'Keefe // IEEE Trans. on Systems, Man and Cybernetics. – 1994. – Vol. 24. – P. 643–655.
27. *Lee Y. H.* Integration of Forward and Backward Inferences Using Extended Rete Networks / Y. H. Lee, S. I. Yoo // Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 9th, Proceedings of the Ninth International Conference. – 1997. – P.339–345.
28. *Lee Y. H.* Optimizing Real-Time Equational Rule-Based Systems / Y.-H. Lee, A. M. Cheng // IEEE Transactions on Software Engineering. – Feb. 2004. – Vol. 30, № 2. – P. 112–125.

29. *Liberatore P.* Redundancy in logic II: 2CNF and Horn propositional formulae / P. Liberatore // Artificial Intelligence. – Feb. 2008. – Vol. 172, № 2–3. – P. 265–299.
30. *Liu N. K.* An Approach Towards the Verification of Expert Systems Using Numerical Petri Nets / N. K. Liu, T. Dillon // International Journal of Intelligent Systems. – 1991. – № 6. – P. 255–276.
31. *Maciol A.* An application of rule-based tool in attributive logic for business rules modeling / A. Maciol // Expert Systems with Applications. – April 2008. – Vol. 34, №. 3. – P. 1825–1836.
32. *Miranker D.* Performance estimates for the DADO machine: A comparison of TREAT and Rete / D. Miranker // Fifth Generation Computer Systems, ICOT. – Tokyo, 1984.
33. *Miranker D.* On the Performance of Lazy Matching in Production Systems / D. Miranker, D. Brant, B. Lofaso and D. Gadbois // Proc. National Conference on Artificial Intelligence, 1990.
34. *Munro I.* Efficient determination of the transitive closure of a directed graph / I. Munro // Inf. Processing. Letters. – 1971. – V. 1. – P. 56–58.
35. *Neiman D. E.* Issues in the Design and Control of Parallel Rule-Firing Production Systems / D. E. Neiman // J. Parallel Distrib. Comput. – 1994. – № 23. – P. 338–363.
36. *Nemeti I.* Algebraization of Quantifier Logics; an Overview / I. Nemeti // Studia Logica L. – 1991. – nos 3/4. – P. 485–569.
37. *Nguyen T. A.* Knowledge Base Verification / T. A. Nguyen, W. A. Perkins, T. J. Laffey and D. Pecora // AI Magazine. – 1987. – V. 8, № 2. – P. 69–75.
38. *Oles F. J.* An Application of Lattice Theory to Knowledge Representation / F. J. Oles // Theor. Comput. Sci. 249, 1 (Oct. 2000). – P. 163–196.
39. *Poli R.* Backward-chaining evolutionary algorithms / R. Poli, W. B. Langdon // Artificial Intelligence. – August 2006. – Vol. 170, № 11. – P. 953–982.
40. *Sahni S.* Computationally related problems / S. Sahni // SIAM J. Computing 3 : 2. – 1974. – P. 262–279.

41. *Schmolze J. G.* Guaranteeing Serializable Results in Synchronous Parallel Production Systems / J. G. Schmolze // J. Parallel Distrib. Comput. – 1991. – № 13(4). – P. 348–365.
42. *Sowa J. F.* Conceptual Structures: Information Processing in Mind and Machine / J. F.Sowa. – Reading, MA : Addison-Wesley, 1984.
43. *Sowa J. F.* Knowledge Representation: Logical, Philosophical and Computational Foundations / J. F. Sowa. – Brooks Cole Publishing Co., Pacific Grove, CA,1999.
44. *Sowyer B.* Programming Expert Systems in Pascal / B. Sowyer, D. Foster. – John Wiley & Sons, Inc., 1986.
45. *Tomić B.* JavaDON: an open-source expert system shell / B. Tomić, H. Jovanović, V. Devedžić // Expert Systems with Applications. – October 2006. – Vol. 31. – № 3. – P. 595–606.
46. *Warren H. S.* A modification of Warshall's algorithm for transitive closure of binary relations / H. S. Warren // Communs. ACM. – 1975. – Vol.18, № 4. – P. 218–220.
47. *Warshall S.* A Theorem on Boolean matrices / S. Warshall // J. Assoc. Computing Machinery. – 1962. – Vol. 9. – P. 11–12.
48. *Zimmermann T.* Toward intelligent object-oriented scientific applications / T. Zimmermann, P. Bomme ; eds. B. H. Topping and Z. Bittnar // Engineering Computational Technology. – Edinburgh : Civil-Comp Press, UK, 2002. – P. 271–311.
49. *Аверкин А. Н.* Толковый словарь по искусственному интеллекту / А. Н. Аверкин, М. Г. Гаазе-Рапопорт, Д. А. Поспелов // М. : Радио и связь, 1992. – 256 с.
50. *Айвазян С.А.* Прикладная статистика в 3-х томах. Справочное издание / С.А.Айвазян. – М.: Финансы и статистика, 1989. – 608 с.
51. *Антонов А.С.* Под законом Амдала / А.С. Антонов // Компьютерра. – 2002. – № 5. – С. 24–27.
52. *Арлазаров В. Л.* Об экономичном построении транзитивного замыкания ориентированного графа / В. Л. Арлазаров, Е. А. Диниц, М. А. Крон-

форд, И. А. Фараджев // Докл. АН СССР. – 1970. – Т. 194, № 3. – С. 487–488.

53. *Артемьева И.Л.* Методы управления распараллеливанием логического вывода для системы конфлюэнтных продукций / И.Л. Артемьева, М.Б. Тютюнник // Научно-технические ведомости СПбГПУ. – 2008. – №3. – С.99–103.
54. *Бениаминов Е. М.* Алгебраические методы в теории баз данных и представлении знаний / Е. М. Бениаминов. – М. : Научный мир, 2003. – 184 с.
55. *Биркгоф Г.* Теория решеток : пер. с англ. / Г. Биркгоф. – М. : Наука, 1984. – 568 с.
56. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++ : пер. с англ. / Г. Буч. – М. : Бином, 2000. – 560 с.
57. *Вагин В. Н.* Параллелизм в продукционных моделях представления знаний / В.Н. Вагин, А.П. Еремеев // Изв. АН. Техническая кибернетика. – 1994. – №2. – С. 48–55.
58. *Васильев С. Н.* Метод редукции и качественный анализ динамических систем, I / С. Н. Васильев // Изв. РАН. ТиСУ. – 2006. – № 1. – С. 21–29.
59. *Васильев С. Н.* Метод редукции и качественный анализ динамических систем, II / С. Н. Васильев // Изв. РАН. ТиСУ. – 2006. – № 2. – С. 5–17.
60. *Варламов О.О.* Основы многомерного информационного развивающегося (миварного) пространства представления данных и правил / О. О. Варламов // Информационные технологии. – 2003. – № 5. – С. 42–47.
61. *Воеводин В. В.* Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб. : БХВ-Петербург, 2002. – 608 с.
62. *Вуколов Э. А.* Основы статистического анализа. Практикум по статистическим методам и исследованию операций с использованием пакетов STATISTICA и EXCEL. – М.: Форум, 2008. – 464 с.
63. *Гмурман В. Е.* Теория вероятностей и математическая статистика : учеб. пособие для вузов / Гмурман В. Е. – 9-е изд. – М. : Высш. шк., 2003. – 479 с.

64. *Джосьютис Н.* C++ Стандартная библиотека. Для профессионалов : пер. с англ. / Н. Джосьютис. – СПб. : Питер, 2004. – 730 с.
65. *Еремеев А. П.* О корректности продукционной модели принятия решений на основе таблиц решений / А. П. Еремеев // Автоматика и телемеханика. – 2001. – № 10. – С. 78–90.
66. *Закревский А. Д.* Логические уравнения / А. Д. Закревский. – изд. 2-е, стереотипное. – М. : Едиториал УРСС, 2003. – 96 с.
67. *Замулин А. В.* Алгебраическая семантика императивного языка программирования / А. В. Замулин // Программирование. – 2003. – № 6. – С. 51–64.
68. *Иванов А. С.* Модель представления продукционных баз знаний на ЭВМ / А.С. Иванов // Известия Саратовского университета. Серия Математика. Механика. Информатика. – Саратов, 2007. – Т. 7, вып 1. – С. 83–88.
69. *Ивченко Г. И.* Математическая статистика : учеб. пособие для втузов / Г.И. Ивченко, Ю.И. Медведев // – М.: Высшая школа, 1984. – 248 с.
70. Интеллектуализация сложных систем. Язык схем радикалов. Методы и алгоритмы / Под ред. А.В. Чечкина, А.В. Рожнова. М.: «Радиотехника», 2008г. – 96с.
71. *Касьянов В. Н.* Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. – СПб. : БХВ-Петербург, 2003. – 1104 с.
72. *Катериненко Р.С.* Метод ускорения логического вывода в продукционной модели знаний / Р.С. Катериненко, И. А. Бессмертный // Программирование. – 2011. – №3. – С. 76–80.
73. *Клини С.* Математическая логика / С. Клини. – М. : Мир, 1973. – 480 с.
74. *Колмогоров А. Н.* Элементы теории функций и функционального анализа / А. Н. Колмогоров, С. В. Фомин. – М.: Наука, 1972. – 256 с.
75. Критически важные объекты и кибертерроризм. Часть 1. Системный подход к организации противодействия / О. О. Андреев и др. ; под ред. В. А. Васенина. – М. : МЦНМО, 2008. – 398 с.

76. Критически важные объекты и кибертерроризм. Часть 2. Аспекты программной реализации средств противодействия / О. О. Андреев и др. ; под ред. В. А. Васенина. – М. : МЦНМО, 2008. – 607 с.
77. *Кормен Т.* Алгоритмы: построение и анализ, 3-е издание : пер. с англ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – М. : «Вильямс», 2013. – 1328 с.
78. *Кузнецов С. О.* Быстрый алгоритм построения всех пересечений объектов из конечной полурешетки / С. О. Кузнецов // НТИ. Сер. 2. – 1993. – № 1. – С. 17–20.
79. *Левин В. И.* Бесконечнозначная логика в задачах кибернетики / В. И. Левин. – М. : Радио и связь, 1982. – 176 с.
80. *Махортов С. Д.* Логические отношения на решетках / С. Д. Махортов // Вестник ВГУ. Серия Физика, математика. – Воронеж. – 2003. – № 2. – С. 203–209.
81. *Махортов С. Д.* Логические уравнения на решетках / С. Д. Махортов // Вестник ВГУ. Серия Физика, математика. – Воронеж. – 2004, № 2. – С. 170–178.
82. *Махортов С.Д.* О приложениях LP-структур в теории программирования // Вестник ВГУ. Серия системный анализ и информационные технологии. – Воронеж, 2007, № 2. – С. 40–49.
83. *Махортов С. Д.* LP-структуры на решетках типов и некоторые задачи рефакторинга / С. Д. Махортов // Программирование. – 2009. – Т. 35, № 4. – С. 5–14.
84. *Махортов С. Д.* Интегрированная среда логического программирования LPExpert / С. Д. Махортов // Информационные технологии. – 2009. – № 12. – С. 65–66.
85. *Махортов С. Д.* Основанный на решетках подход к исследованию и оптимизации множества правил условной системы переписывания термов / С. Д. Махортов // Интеллектуальные системы. – 2009. – Т. 13, вып 1–4. – С. 51–68.

86. *Махортов С. Д.* Релевантный обратный вывод и верификация логических программ на основе решения уравнений в LP-структурах / С. Д. Махортов // Методы и средства обработки информации: Третья Всероссийская научная конференция. Москва, 6-8 октября 2009 г.: Труды конференции / Под ред. Л.Н. Королева. – М. : ВМиК МГУ, 2009. – С. 143–148.
87. *Махортов С. Д.* LP-структуры для обоснования и автоматизации рефакторинга в объектно-ориентированном программировании / С. Д. Махортов // Программная инженерия. – 2010, № 2. – С. 15–21.
88. *Минский М.* Фреймы для представления знаний : пер. с англ. / М. Минский. – М. : Энергия, 1979. – 152 с.
89. *Найханова Л.В.* Технология создания методов автоматического построения онтологий с применением генетического и автоматного программирования: Монография / Л.В. Найханова. – Улан-Удэ: Изд-во БНЦ СО РАН, 2008. – 244 с.
90. *Нигиян С.А.* О семантике бестиповых функциональных программ / С. А. Нигиян, С. А. Аветисян // Программирование. – 2002. – № 3. – С. 5–14.
91. *Нильсон Н.* Принципы искусственного интеллекта : пер. с англ. / Н. Нильсон. – М. : Радио и связь, 1985. – 376 с.
92. Официальный сайт StatSoft Russia [Электронный ресурс]. – М. : StatSoft Russia, 2013. – (Электронная библиотека).
93. *Подловченко Р. И.* Иерархия моделей программ / Р. И. Подловченко // Программирование. – 1981. – № 2. – С. 3–14.
94. *Поспелов Д.А.* Моделирование рассуждений. Опыт анализа мыслительных актов / Д.А. Поспелов. – М.: Радио и связь, 1989. – 184 с.
95. *Расёва Е.* Математика метаматематики : пер. с англ. / Е. Расёва, Р. Сикорский. – М. : Наука, 1972. – 591 с.
96. *Рихтер Дж*. Windows для профессионалов : Программирование для Windows 95 и Windows NT 4 на базе Win32 API / Пер. с англ. – 3-е изд. – М. : "Рус. редакция", 1997. – 679 с.

97. *Рыбаков В. В.* Базисы допустимых правил логик *S4* и *Int* / В. В. Рыбаков // Алгебра и логика. – 1985. – Т. 24, № 1. – С. 87–107.

98. *Рыбаков В. В.* Базисы допустимых правил модальной системы *Grz* и интуиционистской логики / В. В. Рыбаков // Матем. сборник. – 1985. – Т. 128 (170), № 3. – С. 321–338.

99. *Рыбаков В. В.* Независимые базисы для правил, допустимых в предтабличных логиках / В. В. Рыбаков // Алгебра и логика. – 2000. – Т. 39, № 2. – С. 206–226.

100. *Рыбина Г. В.* Верификация баз знаний в интегрированных экспертных системах / Г. В. Рыбина, В. В. Смирнов // Новости искусственного интеллекта. – 2005. – № 3. – С. 7–19.

101. *Семенов А. А.* Консервативные преобразования систем логических уравнений / А. А. Семенов // Вестник Томского госуниверситета. Приложение. – 2007. – № 23. – С. 52–59.

102. *Сикорский Р.* Булевы алгебры : пер. с англ. / Р. Сикорский. – М. : Мир, 1969. – 375 с.

103. *Таненбаум Э.* Архитектура компьютера : пер. с англ. / Э. Таненбаум. – СПб. : Питер, 2002. – 704 с.

104. *Тейз А.* Логический подход к искусственному интеллекту: от классической логики к логическому программированию : пер. с франц. / А. Тейз, П. Грибомон и др. – М. : Мир, 1990. – 432 с.

105. *Уэно Х.* Представление и использование знаний : пер. с яп. / Х. Уэно, Т. Кояма, Т. Окамото, Б. Мацуби, М. Исидзука. – М. : Мир, 1989. – 220 с.

106. *Цейтин Г. С.* О сложности вывода в исчислении высказываний / Г. С. Цейтин // Записки научных семинаров ЛОМИ АН СССР. – 1968. – Т. 8. – С. 234–259.

107. *Чечкин А. В.* Математическая информатика / А. В. Чечкин. – М. : Наука, Гл. ред. физ.-мат. лит., 1991. – 416 с.

108. *Чечкин А.В.* Интеллектуализация сложной системы, как средство обеспечения ее информационно-системной безопасности / А.В. Чечкин,

М.В. Пирогов // Фундаментальная и прикладная математика. – 2009. – Т. 15, № 3. – С. 225–239.

109. *Чечкин А.В.* Ультрамножественная модель базы данных и ультраоператорная модель базы знаний проблемной области сложной системы на основе среды нейрорадикалов / А.В. Чечкин // Нейрокомпьютеры: разработка, применение. – 2009. – №12. – С. 4–9.
110. *Щупак Ю. А.* Win32 API. Эффективная разработка приложений / Ю. А. Щупак. – СПб. : Питер, 2007. – 572 с.

Публикации автора по теме диссертации

1. *Болотова С.Ю.* Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений / С.Ю. Болотова, С.Д. Махортов // Искусственный интеллект и принятие решений. – 2011. – № 2. – С. 40–50.

   В работе [1] Болотовой С.Ю. принадлежат математические результаты и алгоритмы.
2. *Болотова С.Ю.* Алгебраическая модель релевантного обратного вывода на основе решения уравнений / С.Ю. Болотова // Математическое моделирование. – 2012. – № 12, т. 24. – С. 3–8.
3. *Болотова С.Ю.* Применение многопоточности в релевантном LP-выводе / С.Ю. Болотова // Нейрокомпьютеры. Разработка, применение. – 2013, № 9. – С. 53–57.
4. *Болотова С.Ю.* Реализация многопоточности в релевантном LP-выводе / С.Ю. Болотова // Программная инженерия. – 2014, № 1. – С. 12–18.
5. *Болотова С.Ю.* О релевантном обратном выводе в системах знаний продукционного типа / С.Ю. Болотова, С.Д. Махортов // Третья Всероссийская конференция с международным участием «Знания – Онтологии – Теории» (ЗОНТ-2011). Новосибирск, 3–5 октября 2011г.:

Материалы конференции. – Новосибирск: Институт математики им. С.Л. Соболева СО РАН, 2011. – Т.1. – С. 73–77.

В работе [5] Болотовой С.Ю. принадлежат математические результаты и алгоритмы.

6. *Болотова С.Ю.* Релевантный обратный вывод и верификация знаний на основе решения уравнений / С.Ю. Болотова // X Всероссийская научная конференция «Нейрокомпьютеры и их применение» (НКП-2012), Москва, 20 марта 2012 года. Тезисы докладов. – М: МГППУ, 2012. – С. 12.

7. *Болотова С.Ю.* Алгебраическая модель релевантного обратного вывода на основе решения уравнений / С.Ю. Болотова // Современные проблемы прикладной математики и информатики (MPAMCS'2012): Тезисы докладов международной молодежной конференции-школы (Дубна, 22–27 августа 2012г.). – Дубна: ОИЯИ, 2012. – С. 67–69.

8. *Болотова С.Ю.* Использование параллельных вычислений в методе релевантного обратного вывода / С.Ю. Болотова // XI Всероссийская научная конференция «Нейрокомпьютеры и их применение» (НКП-2013), Москва, 19 марта 2013 года. Тезисы докладов. – М: МГППУ, 2013. – С. 16.

9. *Bolotova S.* Multi-threaded relevant LP-inference / S. Bolotova, S. Makhortov // Distributed Intelligent Systems and Technologies (DIST'2013): Proceedings of the International Conference (St. Petersburg, July 1–4, 2013). – St. Petersburg: Spbstu, 2013. – Pp. 7–14. (In English)

В работе [9] Болотовой С.Ю. принадлежат математические результаты и алгоритмы.

10. *Bolotova S.* Using multi-threading in the relevant LP-inference method / S. Bolotova, S. Makhortov // Mathematical Modeling and Computational Physics (MMCP'2013): Book of Abstracts of the International Conference (Dubna, July 8–12, 2013). – Dubna: JINR, 2013. – P. 58. (In English)

В работе [10] Болотовой С.Ю. принадлежат математические результаты и алгоритмы.

11. *Болотова С.Ю.* Использование параллельных вычислений в методе релевантного обратного вывода / С.Ю. Болотова // Материалы Всероссийской конференции с международным участием «Знания – Онтологии – Теории» (ЗОНТ-2013, Новосибирск, 8–10 октября 2013 г.). – Новосибирск: Институт математики им. С.Л. Соболева СО РАН, 2013. – Т.1. – С. 50–59.

12. *Болотова С.Ю.* Стратегии релевантного обратного вывода на основе решения логических уравнений / С.Ю. Болотова // Современные проблемы прикладной математики, теории управления и математического моделирования (ПМТУММ-2011): материалы IV Международной научной конференции (Воронеж, 12–17 сентября 2011 г.). – Воронеж : ИПЦ ВГУ, 2011. – С. 37–40.

13. *Болотова С.Ю.* Статистика релевантного обратного вывода / С.Ю. Болотова // Современные проблемы прикладной математики, теории управления и математического моделирования (ПМТУММ-2012): материалы V Международной научной конференции (Воронеж, 11–16 сентября 2012 г.). – Воронеж : ИПЦ ВГУ, 2012. – С. 48–50.

14. *Болотова С.Ю.* Статистические результаты релевантного обратного вывода / С.Ю. Болотова // Актуальные проблемы прикладной математики, информатики и механики: Сб. трудов Международной конференции (Воронеж, 26–28 ноября 2012 года), ч.2. – Воронеж : ИПЦ ВГУ, 2012. – С. 45–49.

15. *Болотова С.Ю.* Библиотека ParallelLPInference / С.Ю. Болотова // Свидетельство о государственной регистрации программы для ЭВМ. – М.: Федеральная служба по интеллектуальной собственности. – № 2013617293 от 04.10.2013.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2013619460

Библиотека ParallelLPInference

Правообладатель: *федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Воронежский государственный университет» (RU)*

Автор: *Болотова Светлана Юрьевна (RU)*

Заявка № 2013617293
Дата поступления 13 августа 2013 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *04 октября 2013 г.*

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов