ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

На правах рукописи

ЛЕЩИНСКАЯ Мария Владимировна

РАЗРАБОТКА НОВЫХ СТРАТЕГИЙ УПРАВЛЕНИЯ ВЫВОДОМ В КЛАССИЧЕСКОМ И НЕЧЕТКОМ МЕТОДЕ РЕЗОЛЮЦИЙ

Специальность 1.2.1. Искусственный интеллект и машинное обучение

Диссертация

на соискание учёной степени кандидата технических наук

Научный руководитель: Леденева Т.М.
Доктор технических наук, профессор

Воронеж – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. Искусственный интеллект (ИИ) становится краеугольным камнем в эволюции технологического прогресса, принимая на себя вызовы современной промышленности и научных исследований. С его помощью решаются задачи анализа огромных массивов данных и автоматизации процессов, что было недостижимо в прошлом. Эти возможности не просто раскрывают новые перспективы для технологических инноваций, но и становятся неотъемлемой частью нашего ежедневного опыта, проникая в самые разнообразные аспекты жизни.

Использование ИИ для решения технических задач вносит значительный вклад в повышение эффективности производственных и исследовательских процессов. Это способствует более быстрой разработке и внедрению инновационных продуктов, обеспечивая при этом их высочайшее качество. В результате, компании и научные учреждения могут не только оптимизировать свою деятельность, но и предлагать рынку более совершенные и конкурентоспособные решения. Важность ИИ особенно заметна в таких высокотехнологичных отраслях, как информационные технологии, разработка программного обеспечения, автомобилестроение, робототехника и космическая промышленность. В этих сферах интеллектуальные системы не только ускоряют процессы разработки и производства, но и способствуют созданию продуктов и услуг нового поколения, которые могут радикально изменить представления о возможном.

Учитывая широкие возможности ИИ, его роль в технологическом развитии и инновационном прорыве не ограничивается узкими специализациями. Напротив, его потенциал распространяется на все более широкий спектр применений, от здравоохранения до образования, от экологии до городского планирования, демонстрируя его важность как ведущего двигателя эволюции во многих аспектах современного общества. Таким образом, ИИ не только выступает как мощный инструмент для решения

существующих задач, но и открывает двери в будущее, где его влияние на наш мир будет только расти.

Одним из важнейших направлений искусственного интеллекта (ИИ) является развитие систем, основанных на знаниях. Логическая модель, базирующаяся на исчислении предикатов первого порядка, отличается возможностью строгого теоретического обоснования процедур обработки знаний, имеет высокий уровень формализации и обладает объяснительной способностью. Если предметная область включает детерминированные жесткие знания, то использование логической модели и алгоритмов логического вывода становится целесообразным. Пусть $H_1,...,H_n$ – гипотезы и φ – некоторое утверждение. *Доказательством* называется поиск ответа на вопрос: является ли формула φ логическим следствием гипотез $H_1,...,H_n$, или, в соответствии с правилом обратной дедукции, является ли формула $H_1 \wedge ... \wedge H_n \to \varphi$ общезначимой (в этом случае рассуждения называются правильными). Поиск доказательств для истинных формул (теорем) достаточно сложен, а раздел ИИ, посвященный данной проблеме, называется *автоматическим доказательством теорем* (существующие компьютерные системы доказательства теорем (пруверы) – Otter, SETHEO, E, Isabelle, Vampire, Waldmeister). Теоретической базой для построения большинства методов автоматического доказательства теорем является *метод резолюций*. Системы логического вывода и метод резолюций, в частности, выступают как уникальные инструменты для автоматизации процессов решения различных задач. Подход, основанный на системах логического вывода, играет все более важную роль в задаче автоматической верификации программ; для синтеза программного и аппаратного обеспечения (система помогает специалисту в создании программ из спецификаций); для разработки поисковых систем, основанных на рассуждениях; в задачах медицинской диагностики; при создании компьютерных игр.

Резолютивный вывод, являясь разновидностью логического вывода, имеет существенный недостаток, который заключается в формировании такого множества резольвент, большинство из которых оказываются ненужными. В связи с этим актуальными являются исследования по теме диссертации, связанной с разработкой и обоснованием стратегий управления выводом, в том числе в условиях неопределенных суждений, для формализации которых используется нечеткая логика.

Степень разработанности темы исследования. Автоматическое доказательство теорем берет начало от работ J. Herbrand; принцип резолюций был предложен J.A. Robison, а затем развит в исследованиях следующих ученых: J.R. Slagle, R.S. Boyer, D. Luckham, D.A. Plaisted, P.V. Andrews, W. Bibel, Н.А. Шанин, В.Н. Вагин (модификации принципа резолюций); R. Kowalski (основы логического программирования); L. Zadeh, D. Dubois, H. Prade, R.C.T. Lee, M. Mukaidono, M.A.C. Viedma, F.A. Fontano, B. Mondal, S. Raha (рассуждения в условиях неопределенности и нечеткий метод резолюций).

Диссертация выполнена в рамках одного из основных научных направлений Воронежского государственного университета «Математическое моделирование, программное и информационное обеспечение, методы вычислительной и прикладной математики и их применение к фундаментальным исследованиям в естественных науках».

Цель исследования заключается в разработке и обосновании новых стратегий управления выводом как в классической, так и в нечеткой логике для повышения эффективности метода резолюций в системах автоматического доказательства теорем.

Для достижения цели решались следующие **задачи:**

1. Анализ метода резолюций, включая его основные принципы, преимущества и недостатки, а также формулировка целей и задач исследования.

2. Разработка новых стратегий управления выводом в методе резолюций с учетом особенностей как исходных дизъюнктов, так и формируемых на основе резолютивного вывода.

3. Исследование подходов к определению нечеткой резольвенты и разработка стратегий управления выводом в методе резолюций для неопределенных суждений.

4. Программная реализация стратегий управления выводом в классическом и нечетком вариантах метода резолюций.

Объектом исследования являются автоматические рассуждения на основе формальной логики.

Предмет исследования – метод резолюций в классической и нечеткой логике.

Методы исследования базируются на принципах искусственного интеллекта и математического моделирования; теоретическая основа для получения научных результатов – это математическая логика, теория нечетких множеств и нечеткая логика, теория графов, теория алгоритмов. Для программной реализации использовались современные методы и технологии объектно-ориентированного программирования.

Научная новизна. В диссертации получены следующие результаты, характеризующиеся научной новизной**:**

– комплекс стратегий управления выводом в форме эвристических правил для выбора дизъюнктов при построении резольвенты, учитывающий особенности дизъюнктов, такие как унифицируемость, наличие одночленов, частоту использования при построении предыдущих резольвент; специальным образом формируемые весовые коэффициенты, что позволяет сократить количество формул в резолютивном выводе и, тем самым, обеспечить быстродействие соответствующих алгоритмов;

– критерии для сравнения стратегий, отличающиеся учетом структурных характеристик графа вывода, при этом для каждой стратегии определена оценка сложности алгоритмической реализации, учитывающая количество

предложений и среднее количество атомов в предложении, что позволяет на этапе выбора стратегии определить наиболее подходящую для повышения эффективности метода резолюций;

– совокупность утверждений о существовании логически значимой нечеткой резольвенты в смысле определений Lee и Mukaidono при условии, что для формализации основных логических связок используются треугольные нормы и конормы, позволяющие формализовать соответствующие стратегии управления выводом в условиях неопределенности с учетом семантики нечетких логических операций;

– алгоритмы построения резолютивного вывода для нечеткого метода резолюций, отличающиеся различными подходами к определению нечеткой резольвенты, что расширяет область применимости метода для решения прикладных задач в контекстах, требующих учета неопределенности и обработки неопределенных суждений;

– структура и программная реализация приложения с различными вариантами метода резолюций, отличающегося возможностью выбора подходящей стратегии и предназначенного как для исследовательских целей, так и для решения практических задач, связанных с обоснованием принимаемых решений.

Соответствие Паспорту специальности. Полученные в диссертации результаты, характеризующиеся научной новизной, соответствуют следующим пунктам Паспорта специальности 1.2.1 «Искусственный интеллект и машинное обучение»: п.1 «Естественно-научные основы и методы искусственного интеллекта»; п. 3 «Методы и алгоритмы моделирования мыслительных процессов: рассуждений, аргументации, распознавания и классификации, формирования понятий»; п. 15 «Математические исследования в области статистики, логики, алгебры, топологии, анализа функций и других областях, необходимых для решения задач искусственного интеллекта и машинного обучения».

Теоретическая и практическая значимость. Предложенные стратегии управления выводом для классического и нечеткого метода резолюций расширяют возможности для повышения эффективности метода за счет сокращения количества резольвент и уменьшения времени обработки рассуждений. Исследования, касающиеся нечетких резольвент Lee и Mukaidono, а также их обобщения на случай использования для формализации нечетких логических связок (треугольных норм и конорм), расширяют теоретическую базу для построения методов автоматического доказательства теорем при решении задач в условиях неопределенности.

Практическая значимость исследования заключается в том, что предложенные подходы обеспечивает б*о*льшую гибкость, улучшенную точность и эффективность принятия решений, что ведёт к повышению общей производительности систем, основанных на знаниях. Результаты диссертационной работы внедрены в деятельность отдела качества IT-компании «Серф» в форме мобильного приложения на iOS для отслеживания качества тестирования разрабатываемого программного обеспечения. Предложенные стратегии управления выводом и элементы нечеткого метода резолюций используются в учебном процессе Воронежского государственного университета в форме обучающих модулей по дисциплинам «Математическая логика и теория алгоритмов» и «Основы нечеткого моделирования».

Степень достоверности и апробация результатов. Достоверность результатов диссертационного исследования основана на корректном использовании математического аппарата; обосновании выбора алгоритмических решений и их согласованностью с результатами вычислительного эксперимента; практическом внедрении результатов диссертации в учебный процесс и деятельность IT-компании, связанную с автоматизацией тестирования программного обеспечения. Результаты диссертации докладывались на профильных научных конференциях: Международная научно-техническая конференция "Актуальные проблемы

прикладной математики, информатики и механики" (Воронеж, 2021-2023), Международная конференция по математическому моделированию систем управления, автоматизации и энергоэффективности (International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency) – SUMMA (Липецк, 2022); научные сессии профессорско-преподавательского состава, аспирантов и студентов Воронежского государственного университета (Воронеж, 2019-2023); Межвузовская научная конференция молодых ученых и студентов «Математика, информационные технологии, приложения» (Воронеж, 2023).

Результаты и положения, выносимые на защиту:

– комплекс стратегий и соответствующих им алгоритмов управления выводом, расширяющий существующий арсенал стратегий, – это позволяет адаптироваться к различным задачам, в которых используется логическая модель представления знаний, а решение сводится к процессу автоматического доказательства теорем;

– совокупность критериев как основа для всесторонней оценки и анализа стратегий управления выводом в контексте различных условий и сценариев применения метода резолюций – это позволяет обосновать понятие эффективности и применимости стратегий;

– совокупность утверждений о существовании и свойствах нечеткой резольвенты, а также алгоритмы, реализующие стратегии на основе альтернативных определений нечеткой резольвенты, – это позволяет использовать метод резолюций в условиях неопределенности;

– структура приложения и программная реализация вариантов метода резолюций с различными стратегиями вывода, способная адаптироваться к разнообразным требованиям и специфике входной информации.

Публикации. По теме диссертации опубликовано 9 научных работ (3 из которых опубликованы без соавторов), в том числе: 3 – статьи в рецензируемых научных изданиях, рекомендованных ВАК; 1 – статья в Материалах международной конференции, которые проиндексированы в

Scopus; 4 – в других изданиях; 1 – свидетельство о государственной регистрации программы для ЭВМ.

Личный вклад автора. В работах, опубликованных в соавторстве, лично автором получены следующие результаты: [7] – элементы теоретического анализа определений нечетких резольвент, обоснование стратегий нечеткого метода резолюций; [8] – формулы для индексов сходства/несходства, структура метода резолюций, основанного на введенных индексах; [9,11,13] – разработка и формализация новых стратегий, которые учитывают особенности дизъюнктов как в исходном множестве, так и в множествах, генерируемых в процессе вывода; [7,8,9,10,11,12,13] – разработка алгоритмической и программной реализации предложенных стратегий вывода, проведение вычислительных экспериментов.

Объем работы. Диссертация состоит из введения, четырех глав, заключения, списка использованных источников, включающего 93 наименования. Основная часть изложена на 152 страницах, содержит 26 рисунков, 15 таблиц.

ГЛАВА 1. АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ В ОБЛАСТИ ЛОГИЧЕСКОГО ВЫВОДА С АКЦЕНТОМ НА МЕТОД РЕЗОЛЮЦИЙ

В начале первой главы обращается внимание на обширное поле исследований, посвящённых логическому выводу, которое занимает центральное место в разработке интеллектуальных систем [15]. Логический вывод обеспечивает математический фундамент, позволяющий машинам рассуждать, делать обоснованные выводы и принимать решения на основе заданного набора утверждений. Среди разнообразия методов, применяемых в этой области, метод резолюций выделяется как мощный и широко применяемый подход. Он базируется на правилах преобразования логических выражений с целью достижения определённого заключения и используется во многих системах автоматического доказательства теорем.

Существующие решения в области логического вывода охватывают широкий спектр технологий, начиная от простых экспертных систем, использующих правила вывода, и заканчивая сложными алгоритмами машинного обучения, способными самостоятельно извлекать и обобщать знания из больших объёмов данных. В данной главе подробно рассматривается историческое развитие метода резолюций, его основные принципы и теоретическая база.

1.1 Актуальность и анализ подходов к построению систем ИИ на основе логического вывода

Системы логического вывода стали особенно актуальными в современном информационном обществе, особенно в контексте взаимодействия человека с машиной [23,38,57,61]. В областях, где требуются сложные решения, основанные на наборе предварительно определенных правил и рационального мышления, таких как искусственный интеллект, информационный поиск и автоматизированное управление процессами, системы логического вывода предоставляют эффективные и точные способы

решения проблем [70,73,78,82]. Они позволяют машинам воспроизводить процесс рассуждения человека, что упрощает взаимодействие между людьми и компьютерами и способствует достижению высокого уровня автоматизации [38].

Системы логического вывода служат не только для улучшения взаимодействия между людьми и машинами, но также могут быть использованы для создания более прозрачных и объяснимых ИИ систем [38,66,70]. В условиях возрастающей сложности и применения ИИ, доступ к пониманию принципов работы этих систем становится основой для их надежного и эффективного использования. Системы логического вывода могут предложить структурированный и последовательный подход к принятию решений, что может помочь разработчикам и пользователям лучше понять и управлять этими системами. В научных исследованиях системы логического вывода также важны для обеспечения строгого рассуждения и обоснования выводов, способствуя продвижению и передаче знаний в области науки [70].

Автоматическое доказательство теорем, как важный компонент логического вывода, имеет долгую историю развития, начиная с работ J. Herbrand. Его вклад в логику и методы автоматического доказательства теорем оказал существенное влияние на последующие исследования в этой области [40,70,78].

Принцип резолюций, предложенный J.A. Robinson, стал ключевым моментом в развитии методов автоматического доказательства теорем [69]. Этот метод позволил значительно упростить процесс поиска доказательств, делая его более доступным для компьютерной обработки. Дальнейшее развитие метода резолюций было связано с работами таких ученых, как J.R. Slagle [30,31], R.S. Boyer [25,26,27,28,29], D. Luckham [58,59,60] и D.A. Plaisted [54,74,75]. Их исследования способствовали усовершенствованию и расширению применения метода резолюций, что привело к повышению его эффективности и гибкости.

Особо следует отметить вклад W. Bibel, Н.А. Шанина, и В.Н. Вагина [1,2,3] в модификацию принципа резолюций. Их работы позволили дополнительно углубить понимание логического вывода и расширили возможности его использования в различных сферах ИИ. Эти модификации играют важную роль в совершенствовании методов автоматического доказательства теорем, делая их более адаптируемыми к разнообразным задачам и условиям.

R. Kowalski [45,46,47] оказал существенное влияние на развитие логического программирования, представив идеи, которые тесно переплетаются с логическим выводом. Основываясь на принципе резолюций, логическое программирование открыло новые перспективы для создания сложных систем ИИ, способных к эффективному решению логических задач.

Исследования в области рассуждений в условиях неопределенности, в которых принимали участие L. Zadeh, D. Dubois, H. Prade, и другие, расширили границы традиционного логического вывода [34,84,90,91,92,93]. Введение нечеткого метода резолюций позволило системам ИИ обрабатывать неопределенную и нечеткую информацию, что значительно увеличило их практическую применимость в реальных условиях.

Развитие автоматического доказательства теорем и метода резолюций является результатом многолетних исследований и коллаборации ученых со всего мира. Их работы легли в основу современных подходов к логическому выводу в ИИ, открывая новые возможности для создания более интелл ектуальных и адаптируемых систем. Эти разработки не только улучшили способность ИИ к логическому рассуждению, но и расширили применение ИИ в таких областях, как автоматическое планирование, обработка естественного языка, искусственное зрение, и многие другие.

Сегодня, когда вычислительные системы становятся всё более сложными, возрастает необходимость в их точной и эффективной работе. В таком контексте ключевую роль начинают играть системы логического вывода, основанные на автоматическом доказательстве теорем. Метод резолюций,

значительно упрощает процесс устранения логических противоречий, предоставляя эффективный инструмент для достижения этой цели.

С ростом объемов и разнообразия данных, особенно в сфере больших данных и интернета вещей (IoT), появляется потребность в более продвинутых методах логического вывода, способных адекватно обрабатывать неоднозначную и нечеткую информацию. Нечеткая логика выступает в качестве мощного инструмента для моделирования неопределенности и нечеткости в данных, а ее интеграция с методом резолюций открывает двери к созданию гибких и адаптивных искусственных интеллектуальных систем, способных к эффективному рассуждению даже в условиях неопределенности.

Однако существующие подходы к управлению выводом сталкиваются с проблемами, связанными с вычислительной сложностью и ограниченной адаптивностью к различным задачам и ситуациям. Разработка новых стратегий управления выводом обещает значительно улучшить их эффективность, оптимизировать процессы принятия решений и повысить качество выводов.

В заключение, актуальность данной темы диссертации определяется не только стремительным развитием технологий искусственного интеллекта, но и необходимостью решения комплексных задач, связанных с обработкой неопределенной информации и улучшением эффективности систем ИИ. Это исследование раскрывает новые возможности для использования логического и нечеткого вывода, способствуя созданию более адаптивных, надежных и умных систем, что представляет собой значимый вклад в будущее искусственного интеллекта.

1.2 Описание метода резолюций

Одной из основных задач формальной теории предикатов первого порядка является задача определения правильности рассуждений. Введем необходимые понятия и определения, базирусь на [18,21].

Пусть $H_1,...,H_n$ – гипотезы, φ – заключение (гипотезы $H_1,...,H_n$ и формула φ представляют собой формулы логики предикатов). Рассуждения являются *правильными*, и этот факт обозначается в виде $\frac{H_1,...,H_n}{\varphi}$, если формула $H_1 \wedge ... \wedge H_m \to \varphi \equiv \bar{H}_1 \vee ... \vee \bar{H}_m \vee \varphi$ является общезначимой. Но тогда отрицание этой формулы $\bar{H}_1 \vee ... \vee \bar{H}_m \vee \varphi \equiv H_1 \wedge ... \wedge H_m \wedge \bar{\varphi}$ есть невыполнимая формула. Таким образом, чтобы доказать, что формула φ выводима из заданного множества гипотез, нужно показать, что множество формул $\{H_1,...,H_m,\bar{\varphi}\}$ противоречиво. Для решения данной задачи применим метод резолюций. На этапе подготовки к его использованию осуществляется преобразование множества $\{H_1,...,H_m,\bar{\varphi}\}$ в множество предложений S. В S каждое предложение представляет собой дизъюнкт – конечную дизъюнкцию атомов с отрицанием или без. Атом состоит из предикатного символа (литеры) и списка термов. Данный переход обеспечивается за счет исключения кванторов и приведения каждой формулы к конъюнктивной нормальной форме. Множество полученных элементарных дизъюнкций (дизъюнктов) образует базовое множество предложений S.

Под резолюцией понимается правило вывода, применение которого к двум дизъюнктам, содержащим контрарные литеры (литеры L и $\bar{L}$ называются контрарной парой), позволяет получить резольвенту в форме дизъюнкта как следствие [5,6]. Суть метода резолюций заключается в построении резолютивного вывода, и, если он заканчивается пустым дизъюнктом $\square$, то этот факт является основанием для утверждения о том, что множество предложений S противоречиво, и, следовательно, рассуждения являются правильными. Если, используя все возможные комбинации подходящих дизъюнктов, получить пустой дизъюнкт не удается, то множество предложений является выполнимым.

Особенностью метода резолюций в исчислении предикатов является то, что атомы с контрарными литерами, которые используются для построения

резольвенты, могут зависеть от различных переменных. Поэтому перед построением резольвенты требуется предварительная унификация, которая не всегда возможна. Унификация осуществляется на основе подходящей подстановки [5,6,17].

Подстановкой называется конечное множество вида

$$\theta = \{x_1 / t_1, ..., x_n / t_n\},$$

где x_i – переменная; t_i – терм, отличный от x_i, причем все переменные $x_1, ..., x_n$ различны, и равенство $x_i = x_j$ влечет $t_i = t_j$ для всех $1 \le i \le j \le n$.

Пусть θ – подстановка, F – какое-либо выражение (атом, терм или формула), тогда $F\theta$ обозначает выражение, полученное путем одновременной замены всех свободных вхождений переменных x_i в F на термы t_i соответственно. Если необходимо последовательно выполнить несколько подстановок, то используется композиция подстановок. Пусть $\{F_1, ..., F_m\}$ – множество различных выражений формулы F. Подстановка θ, такая, что в результате ее применения все выражения становятся одинаковыми, т.е. $F_1\theta = ... = F_m\theta$, называется унификатором. Если такая подстановка θ существует, то множество выражений $\{F_1, ..., F_m\}$ называется унифицируемым [49].

Итак, унификатор – это подстановка, при применении которой к заданным формулам, эти формулы превращаются в одну и ту же формулу. Применение унификатора к атомам с контрарными литерами, которые используются для построения резольвенты, позволяет унифицировать их списки переменных. Очевидно, что унификаторов может быть несколько. Наиболее общий унификатор – это минимальная подстановка, которая унифицирует формулы.

Пусть D_1 и D_2 – дизъюнкты, содержащие соответственно атомы с контрарными литерами L и $\overline{L}$, которые в общем случае имеют различные списки переменных. Если для них существует наиболее общий унификатор

θ^*, то дизъюнкт $res(D_1, D_2) = (D_1\theta^* \setminus L\theta^*) \vee (D_2\theta^* \setminus \bar{L}\theta^*)$ называется резольвентой D_1 и D_2. Принято, что $res(L\theta^*, \bar{L}\theta^*) = \square$, где символом $\square$ обозначен пустой дизъюнкт. Пустой дизъюнкт по определению тождественно ложен, его можно понимать как невыполнимую формулу.

Пусть S – множество предложений. Последовательность формул $\varphi_1, ..., \varphi_n$ называется резолютивным выводом, если любая формула $\varphi_i \left(i = \overline{1,n}\right)$ либо принадлежит S, либо является резольвентой каких-либо предыдущих формул.

Имеет место следующее утверждение [50]: если D_1 и D_2 – дизъюнкты и $res(D_1, D_2) = D$, то $D_1, D_2 \mapsto D$, т.е. резольвента выводима из множества дизъюнктов $\{D_1, D_2\}$.

Таким образом, резолютивный вывод – это разновидность логического вывода. Целью в методе резолюций является пустой дизъюнкт $\square$. В общем случае, если пустой дизъюнкт выводим из множества S, то он может быть получен на основе полного перебора. Однако для повышения быстродействия алгоритма вывода возможно сократить полный перебор за счет использования стратегий – эвристических правил поиска пустого дизъюнкта.

В области искусственного интеллекта и теоретической информатики термин "граф вывода" обычно используются в контексте логики предикатов и теории резолюций, которые являются важными инструментами в автоматическом доказательстве теорем.

Граф вывода – это ориентированный граф, где узлы представляют выражения или утверждения, а ребра представляют шаги вывода, то есть логические переходы, которые позволяют получить одно утверждение из других. В некоторых контекстах, например в теории типов, графы вывода также могут иметь более сложную структуру, где узлы и ребра могут иметь аннотации или метки, которые представляют дополнительную информацию о шагах вывода.

Вот ***некоторые общие критерии сравнения графов***:

1. Один из наиболее очевидных критериев сравнения – это размер графа, то есть количество узлов и ребер в графе. В общем случае, чем меньше граф, тем он эффективнее, поскольку он требует меньше ресурсов для обработки.

2. Сложность графа может быть оценена по количеству уровней или глубины графа. Граф с большим количеством уровней может быть более сложным для анализа и требовать более продвинутых стратегий для навигации.

3. Как граф вывода, так и граф резолюций могут быть использованы для поиска решений в некоторой проблемной области. Качество этих решений может быть оценено по различным критериям, включая точность, полноту, и надежность.

4. Эффективность процессов вывода и резолюции, представленных в графе, может быть оценена по их вычислительной сложности. Это может включать в себя такие параметры, как время выполнения, требуемые ресурсы памяти и т.д.

5. Графы могут быть оценены с точки зрения их читаемости и понятности для человека. Это может быть особенно важно при визуализации графа для анализа или при презентации результатов.

Эти критерии могут быть применены для сравнения как отдельных графов вывода, так и для сравнения алгоритмов или методов, которые генерируют эти графы. В контексте автоматического доказательства теорем, графы вывода могут быть весьма сложными и могут иметь тысячи узлов и ребер. Анализ графов и поиск эффективных стратегий для навигации по ним является активной областью исследований в области искусственного интеллекта.

Метод резолюций является одним из основных методов автоматического доказательства теорем в логике. Он позволяет строить доказательства на основе применения правил резолюции к логическим выражениям.

Процесс применения метода резолюций состоит из нескольких шагов [4,5,14,19,78], каждый из которых имеет свою специфику и выполняется в определенной последовательности (рис. 1.1).

Рис. 1.1 – Процесс применения метода резолюций

Первым шагом метода является предварительная подготовка. В этом шаге логическое выражение, которое требуется доказать, приводится к противоречивой форме. Для этого применяются логические преобразования, которые позволяют получить отрицание целевого утверждения.

Далее следует шаг преобразования в КНФ. Логическое выражение преобразуется в конъюнктивную нормальную форму (КНФ). В КНФ каждый дизъюнкт состоит из литералов, которые могут быть переменными или их отрицаниями, и объединены операцией "ИЛИ". Все дизъюнкты объединены операцией "И". Такое преобразование облегчает последующее применение правил резолюции.

Третий шаг – построение резольвенты. В этом шаге применяются правила резолюции. Правила резолюции позволяют получать новые дизъюнкты, называемые резольвентами, путем слияния пары дизъюнктов, содержащих комплементарные литералы. Процесс построения резольвенты продолжается до достижения определенного условия остановки.

Четвертый шаг – проверка пустого дизъюнкта. Если в результате применения правил резолюции получается пустой дизъюнкт, то противоречие считается доказанным. В этом случае исходное утверждение, которое нужно было доказать, считается истинным.

Если пустой дизъюнкт не достигается, но больше нет новых резольвент, то исходное утверждение не доказано.

Анализ подходов к реализации метода резолюций, используемого в логических исчислениях и автоматическом доказательстве теорем, является важным аспектом исследования. Этот метод основан на применении правила резолюции, которое позволяет вывести новые логические заключения из набора предпосылок.

Далее, опираясь на [4,14,19,36,88,89] рассматриваются ключевые аспекты реализации метода резолюций, включая выбор подходящего представления данных, определение порядка применения правил резолюции, применение оптимизаций и эвристик, управление памятью, параллельную и распределенную обработку, а также использование дополнительных методов и техник для повышения эффективности и расширения возможностей доказательства.

Первым аспектом, требующим внимания при реализации метода резолюций, является *выбор подходящего представления данных*. Для эффективной работы метода необходимо выбрать структуры данных, способные эффективно хранить дизъюнкты и связи между ними. Распространенными выборами являются графы или деревья, которые обеспечивают удобный доступ к необходимым данным.

Вторым аспектом является *определение порядка применения правил резолюции*. Этот аспект включает выбор стратегии для выбора пары дизъюнктов, которые будут использоваться для резольвирования. Существует несколько стратегий, таких как выбор первой доступной пары, выбор наиболее специфичной или наиболее общей пары. Правильный выбор стратегии может значительно повлиять на эффективность доказательства и время работы алгоритма.

Оптимизации и эвристики являются третьим аспектом реализации метода резолюций. В данном контексте рассматриваются различные методы ускорения процесса доказательства. Например, можно применять ограничения на выбор пар дизъюнктов для резольвирования, отсекать ненужные резольвенты или использовать эвристики для выбора наиболее перспективных пар дизъюнктов. Эти оптимизации и эвристики помогают снизить вычислительную сложность и повысить эффективность метода резолюций.

Управление памятью является четвертым аспектом, который следует учесть при реализации метода резолюций. В процессе работы метода требуется хранить множество дизъюнктов и промежуточных результатов. Эффективное управление памятью является критически важным для успешной реализации метода резолюций. Для этого можно применять стратегии сокращения или удаления ненужных дизъюнктов, использовать компактные структуры данных или применять техники сжатия информации.

Пятый аспект связан с *параллельной и распределенной обработкой*, которая может быть применена для ускорения работы метода резолюций в случае больших объемов данных или сложных доказательств. Возможно разделение вычислений на подзадачи, которые выполняются параллельно на нескольких процессорах или узлах сети. Это позволяет эффективно использовать вычислительные ресурсы и сократить время выполнения доказательства.

Наконец, реализации метода резолюций могут использовать дополнительные методы и техники для повышения эффективности и расширения возможностей доказательства. Это может включать использование семантического разрешения, ограничений на переменные, оптимизаций для особых классов задач и другие методы. Применение этих дополнительных методов и техник способствует более точному и быстрому доказательству.

В целом, реализация метода резолюций требует выбора подходящих стратегий, оптимизаций и техник, а также учета особенностей предметной области и поставленных задач. Рассмотрение указанных аспектов позволяет повысить эффективность и скорость работы метода резолюций, что делает его более применимым в реальных задачах автоматического доказательства и логических исчислениях.

Существуют различные подходы к реализации метода резолюций [4,14,19,36,39,55,78,88,89], среди которых выделим следующие.

Классическая резолюция является стандартным подходом к методу резолюций. Она основывается на применении простых правил резолюции и правила унификации для вывода новых логических высказываний из исходных предпосылок. Его основная идея состоит в преобразовании исходного набора формул в конъюнктивную нормальную форму (КНФ), затем применяется процесс, называемый резолюцией, чтобы выводить новые клозы (конъюнкции литералов) до тех пор, пока не будет получен пустой клоз (отрицание исходной формулы) или больше не сможет быть получено новых клозов.

Модифицированная резолюция, в свою очередь, представляет собой расширение классической резолюции с целью повышения эффективности и улучшения возможностей вывода. В этом подходе могут быть применены различные оптимизации, такие как упрощение формул, предварительная обработка и использование стратегий выбора.

Автоматический метод резолюций является автоматизированным подходом к резолюции с использованием компьютерных алгоритмов. Он широко используется в искусственном интеллекте и формальной верификации, и включает в себя различные техники и методы для автоматического применения правил резолюции к задачам логического вывода.

Модальная резолюция представляет собой расширение метода резолюций для работы с модальными логиками. Она обеспечивает средства для вывода в модальных логиках с использованием правил резолюции.

Динамическая резолюция объединяет метод резолюций с идеями динамической логики, представляя правила изменения и обновления логических формул в процессе вывода. Это позволяет учитывать изменения состояний и динамическое поведение системы в процессе логического вывода.

Резолюция с ограничениями объединяет метод резолюций с техниками ограничений, позволяя решать задачи, в которых кроме логических формул присутствуют также ограничения на переменные и их значения.

Резолюция в вероятностном контексте применяется для выполнения логического вывода с использованием вероятностных моделей. Она объединяет метод резолюций с теорией вероятностей, позволяя оценивать вероятности логических высказываний и принимать статистические решения.

Нелогическая резолюция позволяет применять метод резолюций не только к логическим выражениям, но и к другим типам данных или проблемам, таким как графы или реляционные базы данных.

Модульная резолюция разбивает исходные формулы на модули или подзадачи, которые решаются независимо друг от друга. Затем результаты объединяются для получения окончательного вывода, что повышает эффективность и масштабируемость метода резолюций, особенно в случаях, когда задача имеет большой размер или сложность.

Эвристическая резолюция включает в себя применение эвристик для ускорения процесса вывода и улучшения его эффективности. Это включает

выбор оптимальной стратегии резолюции, эффективное управление памятью и выбор подходящих аксиом или предпосылок для вывода.

Таким образом, изучение различных подходов к методу резолюций, позволяет расширить понимание и применение метода резолюций в контексте автоматического доказательства теорем и логического вывода.

Ниже приведено ***несколько важных применений метода резолюций***, охватывающих следующие области:

1. Метод резолюции используется в автоматическом доказательстве теорем для поиска противоречий в системе утверждений. С помощью этого метода, если из системы утверждений можно вывести противоречие, то исходная теорема считается доказанной.

2. Метод резолюции используется в ИИ для обработки информации и принятия решений. Он позволяет разрешать противоречия в информации и создавать новую, более точную информацию.

3. Метод резолюции может быть использован для анализа естественного языка и формирования выводов на основе этого анализа. Например, он может использоваться для определения семантической связности предложений или для разрешения неоднозначности в тексте.

4. В области проектирования аппаратного и программного обеспечения, метод резолюции может быть использован для проверки корректности моделей и алгоритмов.

5. Метод резолюции может быть использован для оптимизации и решения задач поиска, например в области планирования и маршрутизации.

Помимо этого, метод резолюции может быть полезен в любой области, где требуется логический анализ и обработка информации. Например, для генерации кода:

6. Метод резолюций может быть использован в контексте синтеза кода, где целью является генерация кода, который удовлетворяет некоторому заданному условию. Здесь задача может быть сформулирована как задача

удовлетворения ограничений, а метод резолюций может быть использован для нахождения решения.

7. В обучении с подкреплением метод резолюций может быть использован для формирования стратегии, которая затем кодируется в программный код. Здесь стратегия может быть представлена как набор логических утверждений, а метод резолюций используется для выяснения, какие действия следует предпринять в различных ситуациях.

8. Метод резолюций может быть использован для оптимизации кода. Сначала код может быть представлен в виде логической формулы. Затем можно использовать метод резолюций, чтобы упростить эту формулу, что в итоге приведет к оптимизированному коду.

1.3 Примеры практического применения метода резолюций

Ниже представлены известные компьютерные системы, которые используются для автоматического доказательства теорем.

1. **Otter** [64,65,87] – это автоматизированный прувер для доказательства теорем на основе логики первого порядка, разработанный Уильямом МакКьюном в Аргоннской национальной лаборатории. Otter ввел несколько важных техник в области автоматического доказательства теорем, таких как резолюция и параметризация, ограниченные упорядочениями термов, аналогичными тем, что используются в калькуле суперпозиции. Otter примечателен тем, что стал первым широко распространенным высокопроизводительным прувером, однако, сейчас он больше не развивается, последняя запись в журнале изменений датируется 2004 годом. Его преемник – Prover9.

2. **SETHEO** [24,56,68,80,83] была высокопроизводительной системой, основанной на направленном на цели исчислении модели исключения, изначально разработанной под руководством Вольфганга Бибеля. Она была известна своим применением в различных соревнованиях по логике и

доказательству теорем и была интегрирована с другими системами, включая E, для формирования композитных доказателей теорем, таких как E-SETHEO.

3. **Isabelle** [48,49] – это универсальный помощник в проведении доказательств, который позволяет выражать математические формулы на формальном языке и предоставляет инструменты для доказательства этих формул в логическом исчислении. Он был интегрирован с автоматизированными пруверами теорем и широко используется для синтеза программ и верификации формальных спецификаций.

4. **Vampire** [77,86] изначально был разработан в Университете Манчестера Андреем Воронковым и Криштофом Ходером. Сейчас над ним работает растущая международная команда. Он регулярно выигрывает дивизион FOF (среди других дивизионов) на соревновании CADE ATP System с 2001 года. Vampire известен своей эффективностью и надежностью в автоматизированном доказательстве теорем.

5. **Waldmeister** [41] специализируется на единично-уравнительной логике первого порядка и доминировал в дивизионе CASC UEQ для доказательства на основе единичных уравнений с 1997 по 2010 год. Разработанный Арнимом Бухом и Томасом Хилленбрандом, Waldmeister известен своей специализацией и эффективностью в обработке единичных уравнений.

Эти системы демонстрируют разнообразие и развитие в области автоматического доказательства теорем, подчеркивая их уникальные сильные стороны, историческое значение и вклад как в теоретические, так и в практические аспекты автоматического рассуждения и логики.

Помимо прочего, подход автоматического доказательства теорем может быть использован в таких приложениях, как системы управления базами данных, системы автоматического планирования, верификация программного обеспечения, медицинская диагностика, системы рекомендаций и другие.

Далее представлены конкретные предложения по применению метода резолюций, целью которых является не только теоретическое изучение его

потенциала, но и практическая реализация в решении реальных задач различного масштаба и направленности. Таким образом, данные предложения подчеркивают практическую значимость метода резолюций, демонстрируя его способность решать актуальные и важные задачи в различных сферах, от производства до разработки программного обеспечения, что способствует повышению общей эффективности и качества работы предприятий и организаций.

Предложение по использованию метода резолюций для крупной транспортной компании

В рамках данного примера исследуется разработка системы планирования для крупной транспортной компании, специализирующейся на доставке товаров с использованием грузовых машин. Каждый грузовик обладает уникальными характеристиками, такими как максимальная грузоподъемность, скорость, расход топлива и прочие параметры, которые влияют на его эффективность и экономическую выгодность. Более того, разные виды товаров и маршруты могут потребовать использования определенных типов грузовиков для оптимальной доставки.

Целью приложения является разработка системы, основанной на методе резолюций, которое способно определить оптимальный выбор грузовика для каждой доставки, с минимизацией общих затрат и максимизацией эффективности. В связи с множеством переменных и потенциальных противоречий, такая система должна учитывать разнообразные факторы при выборе грузовика.

Система правил, разрабатываемая с использованием метода резолюций, включает в себя ряд факторов, которые влияют на выбор грузовика для каждой доставки. Сформулируем некоторые из этих правил:

– Если доставляемые товары содержат хрупкие элементы, необходимо выбирать грузовик с улучшенной системой амортизации.

– Если грузоподъемность доставки превышает определенное значение, предпочтение отдается грузовику с большой грузоподъемностью.

– Если требуется выполнить доставку в кратчайшие сроки, следует выбирать грузовик с высокой максимальной скоростью.

Система использует эти правила для анализа информации о каждой доставке и выбора подходящего грузовика. Применение метода резолюций позволяет разрешить возможные противоречия между различными правилами и принять оптимальное решение.

Предлагаемая система планирования предоставляет крупной транспортной компании инструмент для оптимизации выбора грузовиков и повышения эффективности доставки.

Предложение по использованию метода резолюций в области разработки интеллектуальных систем поддержки принятия решений

В данном случае предлагается использовать метод резолюции в области разработки интеллектуальных систем поддержки принятия решений (Decision Support Systems, DSS). Основной является создание DSS для компании, специализирующейся на розничной торговле, с целью оптимизации процесса управления запасами товаров. Учитывая множество факторов, таких как предыдущая история продаж, сезонные колебания, прогнозы спроса и проводимые акции, разработанная система должна предоставлять рекомендации и поддерживать принятие решений менеджерами компании.

Метод резолюции, используемый в данном исследовании, предполагает создание набора правил, основанных на знаниях и опыте предметной области. Например, установлены следующие правила: если товар был популярен в прошлом году в тот же сезон, то его запасы следует увеличить; если товар продается плохо в последние месяцы, его запасы следует уменьшить; если ожидается проведение акции на определенный товар, его запасы следует увеличить. Таким образом, система DSS способна обрабатывать данные о продажах и на основе заданных правил резолюции создавать оптимальные рекомендации для менеджеров компании.

Целью использования метода резолюции в данном контексте является разрешение возможных противоречий между различными правилами,

возникающими при принятии решений по управлению запасами. Путем анализа и обработки имеющихся данных о продажах, система DSS обеспечивает наиболее оптимальные решения, с учетом предварительно установленных правил резолюции. Такой подход позволяет повысить эффективность процесса принятия решений в области управления запасами розничной торговли.

Предложение по использованию метода резолюций для проверки корректности программного кода

Одним из важных применений метода резолюции в области разработки программных продуктов является его использование для проверки корректности программного кода. В случае, когда имеется сложная программа, состоящая из множества взаимосвязанных функций и процедур, необходимо обеспечить ее правильное функционирование во всех возможных сценариях.

Для достижения этой цели, разработчик может создать набор логических утверждений, описывающих ожидаемое поведение программы. Например, можно сформулировать следующие утверждения:

– при вызове функции f с параметром x, она должна возвращать результат y;

– функция φ должна быть вызвана после функции f и использовать результат функции f;

– в случае возникновения ошибки ε, программа должна обработать ее с помощью обработчика ошибок ψ.

После формулировки данных утверждений, метод резолюции может быть применен для проверки корректности выполнения программой данных утверждений. Это позволяет выявить ошибки в коде, которые могут привести к неправильному функционированию программы, и произвести необходимые исправления.

Особенно актуальное применение подобных методов наблюдается в области формальной верификации программ, где требуется математически доказать корректность работы программного обеспечения на логическом уровне. Использование метода резолюции позволяет формализовать и автоматизировать процесс проверки корректности кода, что способствует повышению качества и надежности программных продуктов.

1.4 Цель и задачи исследования

Цель исследования заключается в разработке и обосновании новых стратегий управления выводом в методе резолюций. Эти стратегии предназначены для улучшения эффективности систем автоматического доказательства теорем путем интеграции как классической, так и нечеткой логики в процесс резолютивного вывода. Основываясь на глубоком анализе существующих подходов и методов, исследование направлено на обнаружение и устранение недостатков текущих методик, а также на разработку более совершенных и адаптивных алгоритмов управления выводом.

Для достижения поставленной цели были определены и решены следующие задачи:

1. Анализ метода резолюций, включая его основные принципы, преимущества и недостатки, а также формулировка целей и задач исследования.

2. Разработка новых стратегий управления выводом в методе резолюций с учетом особенностей как исходных дизъюнктов, так и формируемых на основе резолютивного вывода.

3. Исследование подходов к определению нечеткой резольвенты и разработка стратегий управления выводом в методе резолюций для неопределенных суждений.

4. Программная реализация стратегий управления выводом в классическом и нечетком вариантах метода резолюций.

Исследование базируется на комплексном подходе, включающем теоретический анализ существующих методов и стратегий в области автоматического доказательства теорем, разработку новых концепций и методик, а также их программную реализацию и экспериментальную проверку.

Методология охватывает следующие аспекты:

– Теоретический анализ: изучение литературных источников, существующих методов и алгоритмов, анализ их преимуществ и недостатков.

– Разработка алгоритмов: создание новых стратегий управления выводом, основанных на классической и нечеткой логике.

– Программная реализация: кодирование разработанных алгоритмов, создание программного обеспечения для их тестирования и оценки.

– Экспериментальная проверка: применение разработанных стратегий в различных условиях, сбор и анализ результатов, сравнение эффективности предложенных подходов с существующими методами.

В результате выполнения поставленных задач ожидается достижение следующих целей:

1) Улучшение эффективности систем автоматического доказательства теорем: повышение скорости и точности вывода в таких системах за счет внедрения разработанных стратегий управления выводом.

2) Расширение применимости метода резолюций: адаптация метода резолюций к обработке неопределенных и нечетких данных, что позволит расширить сферу его применения.

3) Повышение адаптивности систем автоматического доказательства теорем: разработка алгоритмов, способных адаптироваться к различным типам данных и условиям их обработки, что обеспечит более гибкое и эффективное использование таких систем в разнообразных задачах.

1.5 Выводы

Исходя из анализа, представленного в первой главе диссертации, можно сформулировать следующие выводы:

1. Системы логического вывода и метод резолюций занимают ключевую роль в развитии искусственного интеллекта и логического программирования. Эти инструменты не только способствуют решению множества задач, основываясь на формализованных правилах логики, но и играют важную роль в автоматизации процессов логического рассуждения. Актуальность систем логического вывода особенно выражена в их способности автоматизировать рассуждения, что позволяет достигать новых выводов на основе представленных фактов и правил.

2. Современные вычислительные системы становятся все более сложными и многофункциональными, что требует повышения точности и эффективности их работы.

На основе проведенного анализа была сформулирована цель диссертационного исследования: разработка и обоснование новых стратегий управления выводом как в классической, так и в нечеткой логике для повышения эффективности метода резолюций в системах автоматического доказательства теорем. Эта цель подчеркивает стремление углубить и расширить применение систем логического вывода и метода резолюций в самых разнообразных областях, от искусственного интеллекта до автоматического доказательства теорем, и отразить их потенциал в улучшении взаимодействия между человеком и машиной, а также в повышении эффективности и надежности компьютерных систем и программного обеспечения.

Данное исследование направлено на создание фундаментальной основы для будущих инноваций в области автоматического доказательства теорем и искусственного интеллекта, обеспечивая тем самым вклад в научный прогресс и технологическое развитие. Предложенные в диссертации новые стратегии управления выводом обещают не только теоретическую значимость, но и

широкие практические перспективы, открывая новые горизонты для разработки более эффективных и интеллектуальных систем в самых разнообразных прикладных областях.

ГЛАВА 2. НОВЫЕ СТРАТЕГИИ УПРАВЛЕНИЯ ВЫВОДОМ В КЛАССИЧЕСКОЙ ЛОГИКЕ

В данной главе осуществляется исследование стратегий управления выводом в методе резолюций, который относится к универсальным инструментам доказательства общезначимости формул в классической логике. Этот метод сохраняет свою актуальность и является предметом постоянных исследований. Основной интерес к методу резолюций обусловлен его способностью обеспечивать систематический и автоматизированный подход к доказательству теорем, а также к поиску ответов на запросы в базе данных.

Метод резолюций в классической логике базируется на понятии резолютивного вывода, когда на основе правил резолюции можно выводить новые формулы из уже имеющихся. Это делает его мощным инструментом для логического вывода, способным обрабатывать большие и сложные наборы данных.

2.1 Стратегии управления выводом

По сути, стратегия управления выводом – это эвристическое правило, согласно которому выбирается пара дизъюнктов для построения резольвенты.

Среди некоторых стратегий [34,39,55,72,76,78] управления выводом можно выделить следующие (табл. 2.1):

1. Линейная резолюция: это простейшая стратегия, в которой каждое новое доказательство получается путем применения резолюции к предыдущему доказательству и одной из исходных дизъюнкций.

2. Сетевая стратегия: в стратегии сохраняются все полученные резолютивные связи; для каждой полученной резольвенты ищется возможность ее дальнейшего участия в процессе вывода.

3. Выбор разрешения с учетом ввода: в этой стратегии одним из родителей нового утверждения всегда является исходное предложение. Это делает метод более эффективным, ограничивая область поиска.

4. Случайный выбор разрешения: в этом случае дизъюнкция выбирается случайным образом, это может быть полезно в тех случаях, когда другие стратегии не дают результатов, но она может быть менее эффективной.

5. Стратегии ограничения: здесь для резолюции выбираются только те дизъюнкты, которые удовлетворяют определенному критерию, например, в стратегии ограничения по весу выбираются формулы с наименьшим весом.

6. Стратегии управления порядком: они используются для определения порядка, в котором будут выбраны дизъюнкции для резолюции. Это может быть основано на различных критериях, таких как вес дизъюнкции, количество литералов и так далее.

Эти стратегии могут быть применены в разных комбинациях для управления выводом в методе резолюций и эффективного решения задач логического вывода. Однако каждая из них имеет свои собственные преимущества и недостатки, и выбор конкретной стратегии будет зависеть от специфики задачи.

К наиболее известным стратегиям относят следующие стратегии [5]: *стратегия полного перебора*, которая опирается на вычисление всевозможных резольвент каждого уровня; *стратегия опорного множества*, согласно которой один из дизъюнктов выбирается из опорного множества, первоначально полученного на основе отрицания формулы, а затем дополненного потомками; *стратегия предпочтения одночленам*, при использовании которой в качестве родительской вершины каждый раз пытаются выбрать одночлен; *линейная стратегия*, которая для каждой резольвенты выделяет родителя (хотя бы одного), принадлежащего к базовому множеству.

В табл. 2.1 представлены преимущества и недостатки перечисленных стратегий управления выводом.

Таблица 2.1 – Некоторые стратегии управления выводом в методе резолюций

Стратегия	**Преимущества**	**Недостатки**
Линейная резолюция	Простота применения; Эффективность в задачах с линейной структурой	Может быть менее эффективной для сложных задач с нелинейными структурами
Сетевая резолюция	Полный поиск всех возможных цепочек вывода, что обеспечивает более глубокий анализ.	Большое потребление памяти и ресурсов из-за необходимости хранить все резольвенты и связи.
Выбор разрешения с учетом ввода	Уменьшает область поиска, делая процесс более эффективным	Может пропустить некоторые потенциальные решения
Случайный выбор разрешения	Может быть полезен при отсутствии прогресса с другими стратегиями	Возможно, менее эффективна из-за случайного характера выбора
Стратегии ограничения	Позволяет контролировать сложность доказательств, выбирая более "легкие" дизъюнкции	Могут пропустить более сложные, но более быстрые или точные решения
Стратегии управления порядком	Позволяет контролировать порядок, в котором рассматриваются доказательства	Требует определения критериев для упорядочения, что может быть сложной задачей
Стратегия полного перебора	Гарантированно находит решение, если оно существует, проста в реализации	Очень ресурсоемкая, так как проверяет все возможные комбинации, неэффективна для больших наборов данных
Стратегия опорного множества	Позволяет сократить объем перебора путем ограничения выбора дизъюнктов, эффективна, когда формула включает в себя большое количество дизъюнктов	Не гарантирует нахождение решения, а эффективность зависит от выбора опорного множества
Стратегия предпочтения одночленам	Позволяет сократить объем перебора путем предпочтения одночленных дизъюнктов, часто приводит к быстрому нахождению решения	Не гарантирует нахождение решения, возможны случаи, когда приводит к зацикливанию процесса
Линейная стратегия	Эффективна для простых задач, когда резольвенты можно легко распределить по базовым множествам, позволяет упорядочить процесс резолюции	Не гарантирует нахождение решения, может быть неэффективна для сложных задач с большим количеством дизъюнктов

Для совершенствования описанных стратегий поиска решения в контексте автоматического доказательства теорем или решения логических задач можно предложить несколько направлений улучшения:

1. Разработка алгоритмов для эффективного хранения и доступа к резольвентам. Такой подход может существенно уменьшить потребление памяти и ресурсов, а как следствие стратегия будет более эффективной.

2. Интеграция дополнительных эвристических методов и машинного обучения. В этом случае, ожидается выбор более точных и эффективных путей поиска, с минимизацией риска пропуска потенциальных решений.

3. Внедрение адаптивных алгоритмов, которые учитывают предыдущие попытки и результаты. Может повысить эффективность случайного выбора, делая его менее случайным и более направленным на нахождение решения.

4. Создание более сложных механизмов для оценки "веса" дизъюнкций. Данный подход может помочь в выборе не только "легких", но и потенциально более эффективных в построении резольвенты дизъюнкций.

5. Разработка интеллектуальных систем, способных динамически адаптировать порядок рассмотрения предложений на основе предварительного анализа. В этом случае, есть риск как увеличить время поиска решения, если аптация будет занимать большое количество времени, но и можно получить невероятную эффективность процесса.

6. Применение параллельных вычислений и распределенных систем. Параллелизация и распределение ресурсов может существенно ускорить процесс поиска решения, делая его более приемлемым для использования на практике.

7. Разработка методов для автоматического выбора оптимального опорного множества на основе анализа структуры задачи. Если правильно выбрать опорное множество, и вовремя его адаптировать под процесс решения задачи, можно повысить шансы на нахождение решения и общую эффективность стратегии.

8. Введение механизмов контроля за процессом выбора дизъюнктов. Такими механизмами может быть: ограничение количества последовательных выборов одночленных дизъюнктов или анализ предыдущих шагов.

Каждое направление направлено на устранение недостатков и улучшение преимуществ существующих стратегий.

Для разработки новых стратегий резолюции были выбраны несколько перспективных направлений, которые ориентированы на улучшение эффективности процессов принятия решений. Одно из ключевых направлений — это внедрение адаптивных алгоритмов, способных анализировать и учитывать результаты предыдущих попыток решения задач, тем самым повышая вероятность успеха в последующих итерациях. Также важное место занимает создание сложных механизмов для оценки "веса" дизъюнкций, что позволяет более точно определять приоритетность тех или иных дизъюнктов в процессе резолюции. Разработка интеллектуальных систем, способных динамически адаптировать порядок рассмотрения предложений на основе предварительного анализа задачи и доступных данных, обещает значительно повысить общую эффективность решения. Наконец, введение методов для автоматического выбора оптимального опорного множества дает возможность существенно сократить объемы перебора и ускорить нахождение решения.

2.2 Разработка новых стратегий и их алгоритмическая реализация

2.2.1 Алгоритмы, реализующие рейтинговую стратегию управления выводом

На основе анализа существующих стратегий вывода можно утверждать, что перспективным направлением совершенствования является учет всех преимуществ, которые характерны для рассмотренных выше стратегий, а именно предпочтение одночленов, использование опорного множества, резольвирование атомов из разных множеств и линейный подход к поиску опровержения. Новая стратегия основана на построении рейтинга

предложений и позволяет проверять литералы на возможность унификации в определенном порядке [9]. Она нацелена на предварительную работу с унифицируемыми литералами (именно такие литералы имеют наивысший рейтинг).

Пусть имеется базовое множество предложений. Каждому предложению присвоим рейтинг в соответствии со следующими правилами:

1) если атом имеет отрицание, то рейтинг предложения увеличить на 2, а если отрицание отсутствует, то на 1 (это позволит быстрее найти противоположные атомы и сократить время поиска унифицируемых литеров);

2) если предложение состоит из одного литерала, то его рейтинг увеличивается на 1 (данное правило реализует предпочтение одночленам, если они имеются, поскольку использование одночленов позволяет сократить полный перебор и прийти пустому дизъюнкту и или к множеству предложений, для которого не существует резольвент);

3) если в предложении имеется часто встречающийся атом, то его рейтинг увеличивается на 1 (данное свойство дает возможность быстрее проверить на унификацию те литералы, что находятся в предложениях, состоящих из одинаковых атомов, так как просмотр предложений из одинаковых атомов может скорее привести к контрарным литералам).

Найденные на данной итерации рейтинги используются для формирования списка предложений в соответствии с убыванием рейтингов. Рейтинг предложений пересчитывается после каждой итерации, при этом под итерацией подразумевается единичный полный просмотр всего списка предложений и их возможная унификация. Цикл обновления рейтинга и унификации происходит до тех пор, пока не будет найден пустой дизъюнкт, или же пока не останется конечное множество предложений, состоящих из не унифицируемых литералов.

Такой метод поиска опровержения эвристически позволяет в некоторых задачах быстрее прийти к результату, сократить количество выводимых

резольвент и уровней, за счет предварительного анализа предложений перед унификацией согласно вышеописанным свойствам.

Для изложения алгоритма введем следующие обозначения:

Пусть S – исходное множество дизъюнктов; P_t – t-предложение или t-дизъюнкт; R_t – рейтинг предложения t; D_q – текущий q-атом, R_q – рейтинг атома q.

Алгоритм поиска часто встречающихся дизъюнктов

Шаг 1. Выбрать $C \neq \varnothing$, состоящее из исходных атомов текущей задачи. Построить множество $C_1^{'}$ из n элементов (где $n > 0$ – количество неповторяющихся атомов текущей задачи). Перейти к шагу 2.

Шаг 2. Построить множество $C_2^{'}$, в котором атомы из множества $C_1^{'}$ расположены по убыванию встречаемости во множестве C. Перейти к шагу 3.

Шаг 3. Определить множество часто встречающихся элементов $C'' = \varnothing$. Положить $j = 0$. Перейти к шагу 3.1

Шаг 3.1 Добавить j-элемент множества $C'[j]$ во множество C'' и перейти к шагу 4.2.

Шаг 3.2 Если $j < n - 1$, то перейти к шагу 3.3, иначе к шагу 5.

Шаг 3.3 Если значение встречаемости j-элемента $C_2^{'}[j]$ совпадает со значением встречаемости $j+1$-элемента $C_2^{'}[j+1]$, то положить $j = j + 1$ и перейти к шагу 3.1, иначе перейти к шагу 4.

Шаг 4. Алгоритм завершает свою работу. Построенное множество C'' состоит из часто встречающихся атомов текущей задачи.

Алгоритм построения рейтинга предложений

Шаг 1. Положить $R_t = 0, R_q = 0, t = 0, q = 0$.

Шаг 2. Если $D_q \in P_t$ содержит отрицание, то положить $R_t = R_t + 2$, и перейти к шагу 3.

Шаг 3. Если $D_q \in P_t$ единственный атом в предложении P_t, то $R_t = R_t + 1$. Перейти к шагу 4.

Шаг 4. Если $D_q \in M$ встречается во множестве C" на текущей итерации (где C" строится во время работы алгоритма поиска часто встречающихся дизъюнктов), то $R_t = R_t + 1$, где t – номер предложения, который содержит атом D_q. Перейти к шагу 5.

Шаг 5. Сформировать массив предложений по убыванию их рейтингов (данный массив представляет собой список предложения для получения резольвент и формируется на каждой итерации в методе резолюций).

Пусть M – исходное множество предложений, изменяющееся со временем построения резольвент; M_k – упорядоченное в соответствии с рейтингом множество M на k-ой итерации; S_k – множество резольвент на k-ой итерации;

Алгоритм, основанный на построении рейтингов

Шаг 1. Положить $k = 0,\ M = S \setminus S_k,\ S_{k+1} = \varnothing$.

Шаг 2. Если во множестве S_k содержится пустой дизъюнкт $\square$, то исходное множество S предложений противоречиво, следовательно, рассуждения являются правильными, и алгоритм завершает свою работу. Иначе перейти к шагу 3.

Шаг 3. Если существуют дизъюнкты $D_i \in M$ и $D_j \in M$ такие, что из них можно получить резольвенту, то перейти к шагу 3.1, иначе – к шагу 5.

Шаг 3.1 Положить $i = 1,\ j = n$ (где i, j – индексы элементов массива M_k, а n – количество предложений во множестве M). Построить упорядоченное множество предложений M_k из M и перейти к шагу 3.2.

Шаг 3.2 Если существует резольвента для $M_k[i]$ и $M_k[j]$, то перейти к шагу 4, иначе перейти к шагу 3.3.1.

Шаг 3.3.1 Если $j > i$, положить $j = j - 1$ и перейти к шагу 3.2, иначе перейти к шагу 3.3.2

Шаг 3.3.2 Если $i < n$, положить $i = i + 1,\ j = n$ и перейти к шагу 3.2, иначе перейти к шагу 5.

Шаг 4. Положить $S_{k+1} = S_{k+1} \cup \left\{ res\left(D_i, D_j\right) \right\}$ и перейти к шагу 5.

Шаг 5. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), и алгоритм завершает свою работу. Иначе положить $k = k + 1,\ S_{k+1} = \varnothing$, и перейти к шагу 5.1.

Шаг 5.1 Если множество M не пусто, то изменить множество M по следующему правилу: $M = M \setminus M_k[i] \cup M \setminus M_k[j],\ n = n - 1$, и перейти к шагу 2, иначе перейти к шагу 5.2.

Шаг 5.2. Алгоритм заканчивает свою работу. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), в противном случае – однозначно сказать нельзя.

Во время работы алгоритма по построению рейтинга предложений на каждой итерации, получаем массив M_k дизъюнктов таких, что в начале последовательности располагаются предложения, состоящие по возможности из отрицательных часто встречающихся атомов или в принципе из одного атома, а в конце – дизъюнкты с атомами без отрицания. А значит, рассматривая первый и последний элементы массива, высока вероятность унификации таких дизъюнктов, ведь потенциально мы ищем атом и его отрицание. Таким образом, данный алгоритм позволяет сократить образование лишних дизъюнктов и множество резольвент, так как резольвированием потенциально унифицируемых дизъюнктов можно быстрее получить пустой дизъюнкт.

2.2.2 Алгоритм, реализующий стратегию на основе минимального дизъюнкта

Заметим, что в процессе работы метода резолюций базовое множество дизъюнктов S, сформированное на входе метода, дополняется резольвентами, которые образуются на следующих итерациях. Пусть на некоторой итерации в этом множестве оказался пустой дизъюнкт □, тогда базовое множество S противоречиво. Поскольку каждая итерация связана с построением резольвенты, то необходимо найти пару с контрарными литерами. Перебор всей полученной последовательности формул можно упорядочить и сократить, если на каждой итерации формировать два множества предложений с потенциально унифицируемыми литерами, начиная с базового множества S [12].

Для изложения алгоритма введем следующие обозначения:

Пусть S – исходное множество дизъюнктов; R – подмножество дизъюнктов из S, которое обрабатывается на определенном этапе алгоритма; S'_k, S''_k – два множества предложений, которые построены из S_k на k-ой итерации; d – дизъюнкт из S'_k; ad – дизъюнкт из S''_k, который выбирается для построения резольвенты (этот дизъюнкт будем называть активным).

Количество атомов, входящих в дизъюнкт, назовем его длиной. Под минимальным дизъюнктом будем понимать дизъюнкт, который имеет минимальную длину по сравнению с другими дизъюнктами, входящими в данное множество. Заметим, что каждый одночлен является минимальным дизъюнктом единичной длины.

На каждой итерации k формируются два множества S'_k и S''_k. Множество S'_k содержит либо некоторый минимальный дизъюнкт, либо резольвенту, которая получена на предыдущем шаге. Множество S''_k состоит из дизъюнктов, которые совместно с дизъюнктом из S'_k потенциально могут образовывать резольвенту.

Идея алгоритма заключается в поиске на каждой итерации резольвенты дизъюнктов $d \in S'_k$ и $ad \in S''_k$ в форме минимального дизъюнкта. Полученная резольвента помещается в множество S'_k, а из множества S''_k удаляется активный дизъюнкт, но в него добавляется дизъюнкт d. Если резольвенты на данной итерации не существует, то множества S'_k и S''_k обновляются на основе поиска минимального дизъюнкта в множестве $D = S''_k$. Заметим, что резольвента может не существовать по двум причинам: либо в выбранных дизъюнктах не существует контрарных литер, либо невозможна унификация дизъюнктов, содержащих подходящие атомы.

Приведем описание алгоритма.

Шаг 1 (инициализация). Положить $k = 0,\ D = S$.

Шаг 2 (правило 1 формирования множеств S'_k и S''_k). Выбрать любой минимальной дизъюнкт из множества D и поместить его в множество S'_k. Положить $S''_k = D \setminus S'_k$.

Шаг 3 (критерии останова). Если S''_k существуют дизъюнкты, резольвента которых образует пустой дизъюнкт □, то останов (исходное множество S является противоречивым). Иначе проверить: если $D = \varnothing$, то останов (исходное множество S является выполнимым); если $D \neq \varnothing$, то перейти к следующему шагу.

Шаг 4 (построение резольвенты). Если существует резольвента $res(d, ad)$ дизъюнктов $d \in S'_k$ и $ad \in S''_k \setminus d$ в форме минимального дизъюнкта или такая, которая является единственно возможной, то перейти к следующему шагу. Иначе положить $D = S''_k,\ k = k + 1$ и перейти к шагу 2.

Шаг 5 (правило 2 формирования множеств S'_k и S''_k). Положить $k = k + 1$, $S'_k = \{res(d, ad)\},\ S''_k = (S'_k \setminus ad) \cup \{d\}$ и перейти на шаг 3.

2.2.3 Алгоритм, реализующий стратегию управления выводом на основе поиска похожих предложений

Следующая стратегия основана на поиске двух похожих предложений и позволяет проверять литералы на возможность унификации быстрее.

Пусть имеется базовое множество предложений. Ищем два похожих предложения: первое из которых берется по порядку во множестве всех, будем называть его базовым на текущей итерации, а второе – подбирается к базовому. Подбор двух похожих предложений осуществляется по описанному ниже алгоритму.

Алгоритм поиска похожих предложений основан на поиске контрарных литералов. Алгоритм осуществляет свою работу для каждого предложения из множества всех предложений задачи.

Такой метод поиска опровержения эвристически позволяет в некоторых задачах быстрее прийти к результату, сократить количество выводимых резольвент и уровней.

Для изложения алгоритма введем следующие обозначения:

Пусть S – исходное множество дизъюнктов; R – подмножество дизъюнктов из S, которое обрабатывается на определенном этапе алгоритма; D_i, D_j – два потенциально похожих предложения, которые построены из R на k-ой итерации при помощи алгоритма поиска похожих предложений; S_k - множество резольвент, полученных на k-ой итерации.

Похожими будем называть такие предложения, которые содержат пару контрарных дизъюнктов. Заметим, что каждые два контрарных одночлена являются похожими.

На каждой итерации k формируются два предложения D_i и D_j, и в случае, если они являются похожими - можно построить резольвенту. Множество R содержит не участвовавшие в построении резольвенты предложения и резольвенту, которая получена на предыдущем шаге.

Алгоритм поиска двух похожих предложений

Шаг 1. Разбить базовое предложение D_i на атомы, каждый атом имеет свой индекс по порядку d_l, где $l = \overline{1,m}$, а m – количество атомов в предложении D_i, $j = i$ перейти к шагу 2

Шаг 2. Положить D_j - $(j+1)$- дизъюнкт, который разбить на атомы b_p, где $p = \overline{1,v}$, а v – количество атомов в предложении D_p, перейти к шагу 3.

Шаг 3. Положить $l = 1$, $p = 1$, перейти к шагу 3.1

Шаг 3.1. Если d_l и b_p - контрарные литералы, то перейти к шагу 4, иначе перейти к шаг 3.2.

Шаг 3.2. Если $p \le v$, то $p = p + 1$ и перейти к шагу 3.1, иначе – к шагу 3.3

Шаг 3.3. Если $l \le m$, то $l = l + 1$, $p = 1$ и перейти к шагу 3.1, иначе – к шагу 5.

Шаг 4. Считается, что найдены похожие предложения - D_i и D_j.

Шаг 5. Считается, что похожих предложений нет.

Алгоритм, основанный на поиске похожих предложений

Шаг 1. Положить $k = 0$, $R = S \setminus S_k$, $S_{k+1} = \varnothing$, $i = 1$.

Шаг 2. Если во множестве S_k содержится пустой дизъюнкт $\square$, то исходное множество S предложений противоречиво, следовательно, рассуждения являются правильными, и алгоритм завершает свою работу. Иначе перейти к шагу 3.

Шаг 3. Если существуют дизъюнкты $D_i \in R$ и $D_j \in R$ такие, что из них можно получить резольвенту, то перейти к шагу 3.1, иначе – к шагу 7.

Шаг 3.1 Если найдены два похожих предложений (D_i и D_j) по алгоритму (базовое предложение - D_i) , то перейти к шагу 3.2, иначе к шагу 7.

Шаг 3.2 Если существует резольвента для D_i и D_j, то перейти к шагу 4, иначе перейти к шагу 3.3.

Шаг 3.3. Если $i < n$, положить $i = i + 1$, и перейти к шагу 3.1, иначе перейти к шагу 7.

Шаг 4. Положить $S_{k+1} = S_{k+1} \cup \left\{ res\left(D_i, D_j\right) \right\}$ и перейти к шагу 5.

Шаг 5. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), и алгоритм завершает свою работу. Иначе положить $k = k + 1$ $S_{k+1} = \varnothing$, и перейти к шагу 6.

Шаг 6. Изменить множество R по следующему правилу: $R = R \setminus D_i \cup R \setminus D_j \cup S_{k+1}$ и перейти к перейти к шагу 3.

Шаг 7. Алгоритм заканчивает свою работу. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), в противном случае – однозначно сказать нельзя.

Рассматривая поочередно похожие дизъюнкты, высока вероятность унификации таких дизъюнктов, ведь потенциально мы ищем атом и его отрицание. Таким образом, данный алгоритм позволяет сократить образование лишних дизъюнктов и множество резольвент, так как резольвированием потенциально унифицируемых дизъюнктов можно быстрее получить пустой дизъюнкт.

2.2.4 Алгоритм, реализующий стратегию управления выводом на основе весов предложений

Для повышения эффективности поиска решения в методе резолюций одним из перспективных направлений является упорядоченная работа с весами атомов и предложений. Стратегия основана на построении поиска общего значения веса предложения и позволяет проверять литералы на возможность унификации в определенном порядке [11].

Итак, пусть имеется базовое множество предложений и информация по весу каждого атома, участвующего во множестве. Для всех предложений

вычислим вес в соответствии со следующими правилами: 1) если в предложении есть только один атом, то вес предложения равен весу данного атома; 2) если в предложении несколько атомов, то вес предложения равен сумме весов атомов.

Вес предложений не пересчитывается после каждой итерации, но если появляется новое предложение после получения резольвенты, то ее вес обязательно рассчитывается. Под итерацией подразумевается единичный полный просмотр всего списка предложений и их возможная унификация. Цикл подсчета веса и унификации происходит до тех пор, пока не будет найден пустой дизъюнкт или пока не закончатся новые предложения, вес которых не был посчитан на предыдущих итерациях, или же пока не останется конечное множество предложений, состоящих из не унифицируемых литералов. Ниже приведен алгоритм вычисления веса предложений (листинг 2.1), который является частью обеих стратегий. Для изложения алгоритмов стратегий введем следующие обозначения: M – исходное множество дизъюнктов; M_k – множество дизъюнктов для k-итерации; D_i – i-предложение или i-дизъюнкт; D'_{ij} – вес j-ого атома в i-м предложении; W – веса предложений; R_i – вес i-предложения.

Листинг 2.1 – Алгоритм вычисления веса предложения

```
// идем по всем предложениям
// count(M) - кол-во предложений
for (int i=0; i<count(M)-1; i++) {
 s=0.0;
 // идем по всем атомам в предложении
 // count(Di) - кол-во атомов в i-предложении
 for (int j=0; j<count(Di)-1; j++)
      // считаем вес i-предложения
      s+=D'ij;
      // запоминаем информацию о весе i-ого предложения
      W[i] = s;
}
```

При работе с этой стратегией после того, как найдены веса всех предложений, требуется отсортировать массив весов, а как следствие и предложений, по убыванию веса предложений.

Получив отсортированный массив дизъюнктов, следует попарно от наиболее увесистого просмотреть предложения для построения резольвенты.

На каждой итерации работаем со множеством M, как только удалось построить резольвенту $res(D_i, D_j)$, дизъюнкты, участвующие в построении, убираются из множества M, а резольвента $res(D_i, D_j)$ добавляется, считается ее вес и она продолжает участие в последующем поиске решения (листинг 2.2).

Листинг 2.2 – Метод резолюций, основанный построении резольвент наиболее близких по весу дизъюнктов

```
    // идем по всем предложениям, в поиске Di
    for (int i=0; i<count(M)-1; i++)
     // идем по всем предложениям, в поиске Dj
     for (int j=0; j<count(M)-1; j++)
          // если резольвенту можно построить, то идем дальше
          if (res(Di,Dj)) {
               // убираем элементы Di, Dj из множества M
               // убираем веса i-ого j-ого элемента из W
               M.delete(Di); W.delete(i);
               M.delete(Dj); W.delete(j);
               // добавляем построенную резольвенту
               M.add(res(Di,Dj)); S.add(res(Di,Dj));
               // если есть пустой дизъюнкт, то завершаем поиск
               if (check(S)) return 1;
               // считаем вес построенной резольвенты
               W.add(weight(res(Di,Dj)));
          }
          return 0; // если прошли по всем предложениям и так
и не вышли по return 1, то решение не найдено и возвращаем 0
```

Текущий алгоритм помогает строить резольвенты, состоящие из одного атома, а значит эвристически направлен на поиск пустого дизъюнкта.

Предложенная стратегия имеет следующие преимущества:

– приведенная стратегия определяют в какой последовательности должны генерироваться резольвенты и эвристически помогает снизить построение лишних резольвент;

– стратегия, основанная на весах дизъюнктов, сужает направление поиска в сторону □

– количество дизъюнктов, многократно участвующих в построении резольвент приемлемо;

– дерево вывода для описанной выше стратегии, обладает высоким уровнем объяснительной способности;

В подавляющем большинстве случаев предложенная стратегия, основанная на поиске наиболее близких по весу дизъюнктов, является эвристически потенциальным резолютивным методом решения.

2.3 Иллюстративные примеры

Пример. Пусть задано следующее множество предложений:

$$\{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}.$$

Докажем или опровергнем его противоречивость. Результаты работы алгоритмов представлены в таблице 2.2 и на рис. 2.1.

Таблица 2.2 – Результаты работы алгоритма на основе построения рейтинга

$k = 0$ $M = \{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}$ $S_0 = \varnothing$
S_0 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
$R(C(x) \vee O(x)) = 1+1+1 = 3$ $R(Z(y) \vee \neg C(y)) = 1+2+1 = 4$ $R(Y(A)) = 1+1+1 = 3$ $R(\neg Z(A)) = 2+1+1 = 4$ $R(\neg O(z) \vee \neg Y(z)) = 2+2+1 = 5$ $M_0 = \{\neg O(z) \vee \neg Y(z), Z(y) \vee \neg C(y), \neg Z(A), C(x) \vee O(x), Y(A)\}$
$D_i = M_0[1] = \neg O(z) \vee \neg Y(z),\ D_j = M_0[n-1] = Y(A)$
$S_1 = \{\neg O(z)\}$
$k = 1,\ S_1 = \varnothing$
S_1 не содержит пустого дизъюнкта
В M есть унифицируемые литералы

$R(\neg O(z)) = 2+1 = 3$ $R(Z(y) \vee \neg C(y)) = 1+2 = 3$ $R(\neg Z(A)) = 2+1 = 3$ $R(C(x) \vee O(x)) = 1+1 = 2$ $M_1 = \{\neg O(z), Z(y) \vee \neg C(y), \neg Z(A), C(x) \vee O(x)\}$
$D_i = M_1[1] = \neg O(z),\ \ D_j = M_1[n-1] = C(x) \vee O(x)$
$S_2 = \{\neg O(z), C(A)\}$
$k = 2, S_2 = \varnothing$
S_2 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
$R(C(x)) = 1+1 = 2$ $R(Z(y) \vee \neg C(y)) = 1+2 = 3$ $R(\neg Z(A)) = 2+1 = 3$ $M_2 = \{Z(y) \vee \neg C(y), \neg Z(A), C(x)\}$
$D_i = M_2[1] = Z(y) \vee \neg C(y),\ \ D_j = M_2[n-1] = C(x)$
$S_3 = \{\neg O(z), C(A), Z(x)\}$
$k = 3,\ S_3 = \varnothing$
S_3 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
$R(Z(x)) = 1+1 = 2$ $R(\neg Z(A)) = 2+1 = 3$ $M_3 = \{Z(x), \neg Z(A)\}$
$D_i = M_3[1] = Z(x),\ D_j = M_3[n-1] = \neg Z(A)$
$S_3 = \{\neg Q(z), C(A), Z(x), \square\}$
S_3 содержит пустой дизъюнкт, алгоритм завершает работу

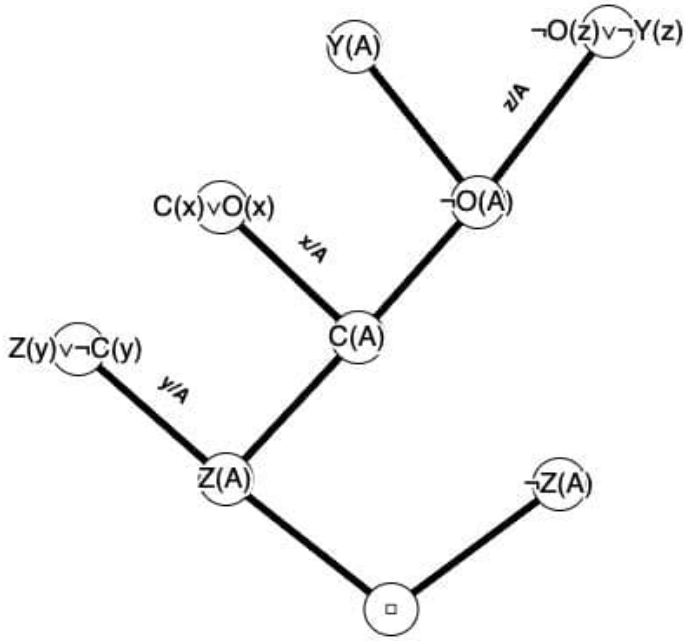


Рис. 2.1 – Дерево вывода, соответствующее алгоритму с рейтингами

Ниже приведена работа алгоритма, основанного на поиске минимального дизъюнкта (табл. 2.3, рис. 2.2).

Таблица 2.3 – Результаты работы алгоритма, основанного на поиске минимального дизъюнкта

$k=0$ $D=S=\{C(x)\vee O(x), Z(y)\vee\neg C(y), Y(A), \neg Z(A), \neg O(z)\vee\neg Y(z)\}$
$S_0'=\{Y(A)\}$, $S_0''=\{C(x)\vee O(x), Z(y)\vee\neg C(y), \neg Z(A), \neg O(z)\vee\neg Y(z)\}$
Критерии останова не выполняются, $D\neq\varnothing$.
$d=Y(A)$, $ad=\neg O(z)\vee\neg Y(z)$, $res(d, ad(z/A))=\neg O(A)$.
$k=1$, $S_1'=\{\neg Z(A)\}$, $S_1''=\{C(x)\vee O(x), Z(y)\vee\neg C(y), \neg O(A)\}$.
Критерии останова не выполняются, $D\neq\varnothing$.
$d=\neg Z(A)$, $ad=Z(y)\vee\neg C(y)$, $res(d, ad(y/A))=\neg C(A)$
$k=2$,

$S_2' = \{\neg O(A)\}$ $S_2'' = \{C(x) \vee O(y), \neg O(A), \neg C(A)\}$
Критерии останова не выполняются, $D \neq \varnothing$.
$d = \neg O(A)$, $ad = C(x) \vee O(x)$, $res(d, ad(x/A)) = C(A)$.
$k = 3$, $S_3' = \{\neg C(A)\}$, $S_3'' = \{\neg O(A), \neg C(A), C(A)\}$
В S_3'' существуют одночлены $\neg C(A)$ и $C(A)$, образующие $res(C(A), \neg C(A)) = \square$, поэтому выполняется критерий останова, и исходное множество S является противоречивым.

Заметим, что на каждой итерации активный дизъюнкт из множества S_k'', по сути, заменяется резольвентой меньшей длины. Следовательно, с каждой итерацией в данном множестве становится все больше дизъюнктов с меньшей длиной, в том числе одночленов, что обеспечивает быстрое продвижение к получению $\square$.

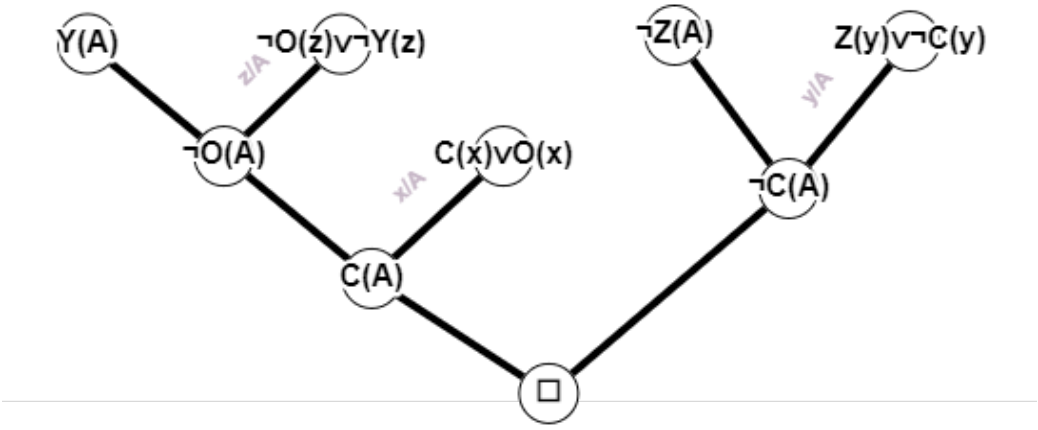


Рис. 2.2 – Дерево вывода, соответствующее алгоритму с поиском минимального дизъюнкта

Ниже продемонстрирована работа алгоритма на основе поиска похожих предложений (табл. 2.4, рис. 2.3). Алгоритм завершает свою работу в случае получения уровня, на котором находится хотя бы один пустой дизъюнкт $\square$.

Таблица 2.4 – Результаты работы алгоритма на основе поиска похожих предложений

$k=0$ $R=\{C(x)\vee O(x), Z(y)\vee\neg C(y), Y(A), \neg Z(A), \neg O(z)\vee\neg Y(z)\}$ $S_0=\varnothing$
S_0 не содержит пустого дизъюнкта
В R есть унифицируемые литералы
$i_{11}=C(x), i_{12}=O(x), i_{21}=Z(y), i_{22}=\neg C(y)$ $D_i=R[1]=C(x)\vee O(x)$, $D_j=R[2]=Z(y)\vee\neg C(y)$
Резольвента существует
$S_1=\{O(y)\vee Z(y)\}$
S_1 не содержит пустого дизъюнкта $k=1$, $S_1=\varnothing$
$R=\{Y(A), \neg Z(A), \neg O(z)\vee\neg Y(z), O(y)\vee Z(y)\}$
В R есть унифицируемые литералы
$i_{11}=Y(A), i_{31}=\neg O(z), i_{32}=\neg Y(z)$ $D_i=R[1]=Y(A)$, $D_j=R[3]=\neg O(z)\vee\neg Y(z)$
Резольвента существует
$S_2=\{O(y)\vee Z(y), \neg O(A)\}$
S_2 не содержит пустого дизъюнкта $k=2, S_2=\varnothing$
$R=\{\neg Z(A), O(y)\vee Z(y), \neg O(A)\}$
В M есть унифицируемые литералы
$i_{11}=\neg Z(A), i_{21}=O(y), i_{22}=Z(y)$ $D_i=R[1]=\neg Z(A)$, $D_j=R[2]=O(y)\vee Z(y)$
Резольвента существует
$S_3=\{O(y)\vee Z(y), \neg O(A), O(A)\}$
S_3 не содержит пустого дизъюнкта $k=3$, $S_3=\varnothing$
$R=\{\neg O(A), O(A)\}$
В R есть унифицируемые литералы
$i_{11}=\neg O(A), i_{21}=O(A)$ $D_i=R[1]=\neg O(A)$, $D_j=R[2]=O(A)$
Резольвента существует

$S_4 = \{O(y) \vee Z(y), \neg O(A), O(A), \square\}$
S_4 содержит пустой дизъюнкт, алгоритм завершает работу

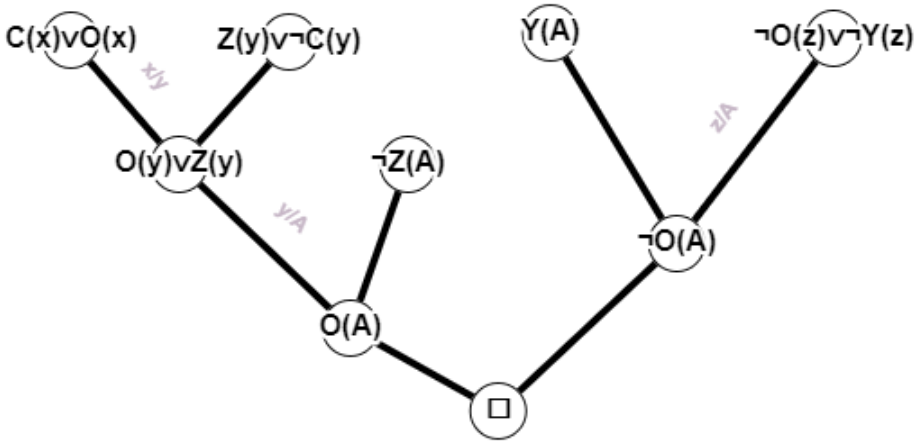


Рис. 2.3 – Дерево вывода, соответствующее алгоритму с похожими предложениями

Далее продемонстрирована работа алгоритма на основе поиска наиболее близких по весу дизъюнктов (табл. 2.5, рис. 2.4).

Таблица 2.5 – Результаты работы алгоритма на основе поиска наиболее близких по весу дизъюнктов

$k = 0,\ S_0 = \varnothing,\ M = \{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}$
S_0 не содержит пустого дизъюнкта, в M есть унифицируемые литералы
$R(C(x) \vee O(x)) = 0,2 + 0,4 = 0,6,\ R(Z(y) \vee \neg C(y)) = 0,3 + 0,8 = 1,1$ $R(Y(A)) = 0,8,\ R(\neg Z(A)) = 0,7,\ R(\neg O(z) \vee \neg Y(z)) = 0,6 + 0,2 = 0,8$ $M_0 = \{Z(y) \vee \neg C(y), \neg O(z) \vee \neg Y(z), Y(A), \neg Z(A), C(x) \vee O(x)\}$
$D_i = M_0[0] = Z(y) \vee \neg C(y),\ D_j = M_0[n-2] = \neg Z(A),\ S_1 = \{\neg C(A)\}$
$k = 1,\ S_1 = \varnothing,\ S_1$ не содержит пустого дизъюнкта, в M есть унифицируемые литералы
$R(\neg C(A)) = 0,8,\ R(\neg O(z) \vee \neg Y(z)) = 0,6 + 0,2 = 0,8,\ R(Y(A)) = 0,8$ $R(C(x) \vee O(x)) = 0,2 + 0,4 = 0,6$ $M_1 = \{\neg C(A), \neg O(z) \vee \neg Y(z), Y(A), C(x) \vee O(x)\}$
$D_i = M_1[0] = \neg C(A),\ D_j = M_1[n-1] = C(x) \vee O(x),\ S_2 = \{\neg C(A), O(A)\}$
$k = 2, S_2 = \varnothing,\ S_2$ не содержит пустого дизъюнкта, в M есть унифицируемые литералы
$R(O(A)) = 0,4,\ R(\neg O(z) \vee \neg Y(z)) = 0,6 + 0,2 = 0,8,\ R(Y(A)) = 0,8$ $M_2 = \{\neg O(z) \vee \neg Y(z), Y(A), O(A)\}$
$D_i = M_2[0] = \neg O(z) \vee \neg Y(z),\ D_j = M_2[n-1] = O(A),\ S_3 = \{\neg C(A), O(A), \neg Y(A)\}$
$k = 3,\ S_3 = \varnothing,\ S_3$ не содержит пустого дизъюнкта, в M есть унифицируемые литералы

$R(\neg Y(A))=0,2,\ R(Y(A))=0,2, M_3=\{\neg Y(A),\neg Y(A)\}$
$D_i=M_3[0]=\neg Y(A),\ D_j=M_3[n-1]=Y(A),\ S_4=\{\neg C(A),O(A),\neg Y(A),\square\}$
S_4 содержит пустой дизъюнкт, алгоритм завершает работу

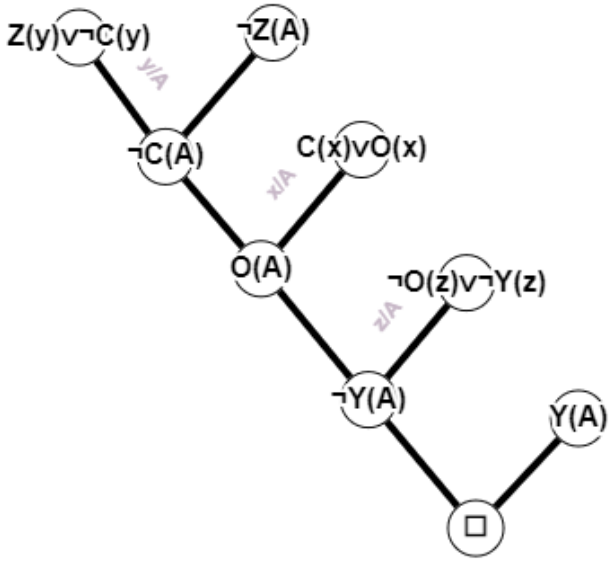


Рис. 2.4 – Дерево вывода, соответствующее алгоритму с наиболее близкими по весу предложениями

Ниже продемонстрирована работа алгоритма полного перебора (табл. 2.6, рис. 2.5). Алгоритм завершает свою работу в случае получения уровня, на котором находится хотя бы один пустой дизъюнкт $\square$.

Таблица 2.6 – Результаты работы алгоритма путем полного перебора

$k=0$ $M=\{C(x)\vee O(x),Z(y)\vee\neg C(y),\neg Z(A),\neg O(z)\vee\neg Y(z)\}$ $S_0=\{Y(A)\}, S_1=\varnothing$
S_0 не содержит пустого дизъюнкта
$D_i=\neg O(z)\vee\neg Y(z), D_j=Y(A)$
$S_1=S_1\cup\{res(\neg O(z)\vee\neg Y(z),Y(A))\}=\{\neg O(A)\}$
$D_i=C(x)\vee O(x), D_j=\neg O(z)\vee\neg Y(z)$
$S_1=S_1\cup\{res(C(x)\vee O(x),\neg O(z)\vee\neg Y(z))\}=\{\neg O(A),C(z)\vee\neg Y(z)\}$
$D_i=C(x)\vee O(x), D_j=Z(y)\vee\neg C(y)$

$S_1 = S_1 \cup \{res(C(x) \vee O(x), Z(y) \vee \neg C(y))\} =$ $\{\neg O(A), C(z) \vee \neg Y(z), O(y) \vee \neg Z(y)\}$
$D_i = Z(y) \vee \neg C(y), D_j = \neg Z(A)$
$S_1 = S_1 \cup \{res(Z(y) \vee \neg C(y), \neg Z(A))\} =$ $= \{\neg O(A), C(z) \vee \neg Y(z), O(y) \vee Z(y), \neg C(A)\}$
Нельзя построить резольвенту из дизъюнктов множеств M и S_1
$k = 1, S_1 = \varnothing, S_2 = \varnothing$
S_1 не содержит пустого дизъюнкта
$D_i = C(x) \vee O(x), D_j = \neg O(A)$
$S_2 = S_2 \cup \{res(C(x) \vee O(x), \neg O(A))\} = \{C(A)\}$
$D_i = C(x) \vee O(x), D_j = \neg C(A)$
$S_2 = S_2 \cup \{res(C(x) \vee O(x), \neg C(A))\} = \{C(A), O(A)\}$
$D_i = Y(A), D_j = C(z) \vee \neg Y(z)$
$S_2 = S_2 \cup \{res(Y(A), C(z) \vee \neg Y(z))\} = \{C(A), O(A)\}$
$D_i = \neg Z(A), D_j = O(y) \vee Z(y)$
$S_2 = S_2 \cup \{res(\neg Z(A), O(y) \vee Z(y))\} = \{C(A), O(A)\}$
$D_i = \neg O(z) \vee \neg Y(z), D_j = O(y) \vee Z(y)$
$S_2 = S_2 \cup \{res(\neg O(z) \vee \neg Y(z), O(z) \vee Y(z))\} =$ $= \{C(A), O(A), \neg Y(y) \vee Z(y)\}$
Нельзя построить резольвенту из дизъюнктов множеств M и S_1, S_2
$K = 2, S_2 = \varnothing, S_3 = \varnothing$
S_2 не содержит пустого дизъюнкта
$D_i = C(A), D_j = \neg C(A)$
$S_3 = S_3 \cup \{res(C(A), \neg C(A))\} = \{\square\}$
$D_i = O(A), D_j = \neg O(A)$
$S_3 = S_3 \cup \{res(O(A), \neg O(A))\} = \{\square\}$
Нельзя построить резольвенту из дизъюнктов множеств M и S_2, S_3
$k = 3, S_3 = \varnothing, S_4 = \varnothing$
S_3 содержит пустой дизъюнкт, алгоритм завершает работу

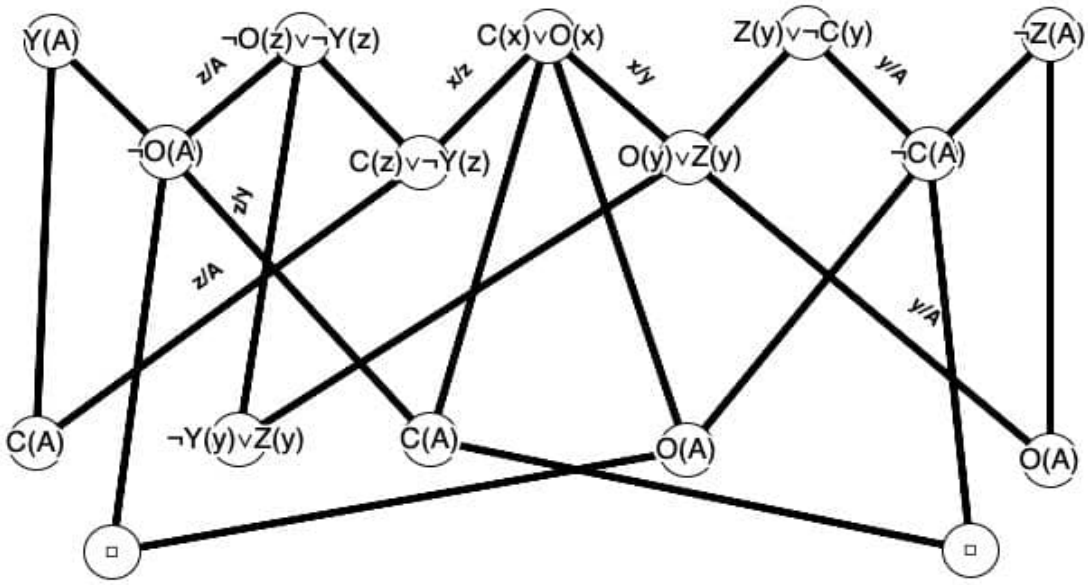


Рис. 2.5 - Дерево вывода, соответствующее алгоритму полного перебора

2.4 Сравнительный анализ новых стратегий управления выводом

Для того, чтобы осуществить сравнение новых стратегий управления выводом, первостепенное значение приобретает разработка объективных критериев, определяющих параметры сравнения. Эти критерии служат основой для всесторонней оценки и анализа стратегий, позволяя выявить как их преимущества, так и возможные недостатки в контексте различных условий и сценариев применения.

Важность разработанных в данном исследовании критериев для сравнения стратегий управления выводом не может быть переоценена. Каждый из выработанных критериев играет ключевую роль в объективной оценке и понимании эффективности и применимости стратегий в различных условиях:

- полнота стратегии;
- количество построенных резольвент;
- скорость роста дерева вывода;
- близость к двоичному дереву;
- реализуемость стратегии;
- количество уровней в дереве вывода;

– количество дизъюнктов многократно участвующих в построении резольвент;

Под *полной* подразумевается такая стратегия поиска опровержения, про которую известно, что если существует какое-либо опровержение по методу резолюции, то существует и опровержение по данной стратегии. Полнота стратегии обеспечивает уверенность в том, что стратегия будет работать во всех предполагаемых сценариях.

Количество построенных резольвент демонстрирует, насколько эффективна стратегия, что непосредственно влияет на скорость достижения результата.

Под *скоростью роста* будем понимать коэффициент увеличения количества элементов во множестве построенных резольвент от уровня к уровню, при этом *общей скоростью роста* будем называть делитель количества резольвент, полученных на первом уровне, на значение количества резольвент, полученных на последнем уровне. Такой критерий непосредственно связан с производительностью и может указывать на оптимальность стратегии в условиях ограниченных ресурсов.

Близость к двоичному дереву отражает структурную эффективность стратегии, насколько дерево вывода демонстрирует понятное визуально решение.

Реализуемость стратегии является критически важным аспектом, указывающим на практическую применимость и возможность внедрения стратегии в реальные процессы.

Такой критерий как, *количество уровней в дереве вывода*, демонстрирует скорость поиска опровержения – чем меньше уровней, тем алгоритм быстрее, так как решение находится за наименьшее число шагов (глобальных итераций).

Анализ различных стратегий обосновывает использование нового критерия – *количество дизъюнктов, многократно участвующих в построении резольвент*. Наименьшее количество действий унификации ускоряет работу

алгоритма, позволяя не тратить время на попытки построения резольвент между множественными дизъюнктами, а также упрощает дерево вывода, позволяя быстрее анализировать поиск решения.

Комплексный анализ стратегий с использованием этих критериев позволяет не только выявить наиболее эффективные подходы, но и дает понимание того, как именно различные аспекты влияют на успешность стратегий в целом. Это, в свою очередь, обеспечивает глубокое и всестороннее сравнение, выявляя не только сильные стороны, но и потенциальные ограничения каждой стратегии.

С учетом перечисленных выше критериев стратегии имеют следующие преимущества по сравнению со стратегией полного перебора (табл. 2.7):

– количество резольвент новых стратегии (рис. 2.1, рис. 2.2, рис. 2.3, рис. 2.4) значительно меньше, чем в случае стратегии полного перебора;

– новые стратегии определяют в какой последовательности должны генерироваться резольвенты и эвристически помогают снизить построение лишних резольвент;

– количество дизъюнктов, многократно участвующих в построении резольвент, также больше в случае алгоритма полного перебора, о чем наглядно свидетельствует дерево вывода (рис. 2.5);

– деревья вывода для новых стратегии имеют больше уровней чем дерево, которое соответствует полному перебору (рис. 2.1, рис. 2.2, рис. 2.3, рис. 2.4), но в первом случае имеем упорядоченное двоичное дерево, которое обладает высоким уровнем объяснительной способности.

Анализ проведенных вычислительных экспериментов подтверждает приведенные выше выводы. В подавляющем большинстве случаев новые стратегии являются наиболее предпочтительными.

Сравнительный анализ новых стратегий представлен в табл. 2.8.

Поиск рейтинговой стратегии сконцентрирован в направлении создания пустого дизъюнкта, так как одночлены рассматриваются в первую очередь.

Таблица 2.7 – Сравнение новых стратегий со стратегией полного перебора

Стратегии	**Кол-во резольвент**	**Кол-во дизъюнктов, участвующих в построении резольвент**	**Кол-во уровней**	**Понятность**
Стратегия с построением рейтинга Стратегия с поиском минимального дизъюнкта Стратегия с поиском похожих предложений	значительно ***меньше***, чем в случае стратегии полного перебора (примерно в 3 раза)	количество дизъюнктов, многократно участвующих в построении резольвент, ***меньше***, чем в случае алгоритма полного перебора (рис. 2.4) (примерно в 2,1 раза)	***больше*** уровней (рис. 2.1, рис. 2.2, рис. 2.3) чем в дереве, которое соответствует полному перебору (рис. 2.4)	деревья, описывающие реализацию новых алгоритмов, ***более понятны,*** так как обладают высоким уровнем объяснительной способности

Аналогично работает и стратегия, основанная на алгоритме поиска минимального дизъюнкта. Стратегия же по поиску похожих предложений снижает количество просмотров группы предложений, которые не могут образовать резольвенту, так как «похожие предложения» отбираются изначально такими, что из них можно построить резольвенту.

Таблица 2.8 – Сравнение новых стратегий между собой

Стратегии	**Необходимость и частота доп. работ перед методом резолюций**	**Сложность доп. работ в начале каждой итерации**	**Преимущества**	**Недостатки**
Стратегия с построением рейтинга	да, каждую итерацию	- сложный подсчет рейтинга каждого дизъюнкта; - подсчет рейтинга каждого предложения; - построение рейтинга	- определяет, в какой последовательности должны генерироваться резольвенты - поиск сконцентрирован в направлении создания пустого дизъюнкта	- в случае равно-рейтинговых предложений стратегия может оказаться менее эффективной - на каждом уровне строится

				только одна резольвента - сложные подсчеты
Стратегия с поиском минимального дизъюнкта	да, каждую итерацию	- определение минимального дизъюнкта	- минимальное количество дополнительных работ перед выполнением метода резолюций - простота и понятность как в анализе, так и реализации - поиск сконцентрирован в направлении создания пустого дизъюнкта	- на каждом уровне строится только одна резольвента - невыигрышна в случае больших предложений и долгом достижении первого пустого дизъюнкта при резольвировании
Стратегия с поиском похожих предложений	да, каждую итерацию	- просмотр предложений и поиск среди них похожих, за счет определения атомов с отрицанием и без	- снижает количество просмотров группы предложений, которые не могут образовать резольвенту	- на каждом уровне строится только одна резольвента

Пусть N – количество предложений, а M – среднее количество атомов в предложении. В таблице 2.9 представлена оценка сложности описанных алгоритмов.

Таблица 2.9 – Оценка сложности алгоритмов представленных стратегий и полного перебора

Стратегия построения рейтинга	**Стратегия поиска минимального дизъюнкта**	**Стратегия поиска похожих предложений**	**Стратегия с весами предложений**	**Стратегия полного перебора**
$O(M \cdot N^2)$	$O(M \cdot N)$	$O(M^2 + N^2)$	$O(N^2)$	$O(N^N)$

Для подсчета оценки сложности стратегий, разобьем общий алгоритм на составляющие, после чего вычислим сумму.

1) Стратегия с построением рейтинга

Стратегия, основанная на построении рейтинга [9], состоит из алгоритма поиска часто встречающихся дизъюнктов, непосредственно алгоритма построения рейтинга и самого алгоритма метода резолюций с использованием текущей стратегии.

Таким образом, получим: алгоритм поиска часто встречающихся дизъюнктов имеет сложность $=O(M \cdot N)$, алгоритм построения рейтинга = $O(N)$, а алгоритм метода резолюций с использованием текущей стратегии = $O(N^2)$, а значит общая сложность составляет $O(M \cdot N) + O(N^2) = O(M \cdot N^2)$.

2) Стратегий с поиском минимального дизъюнкта

Стратегия, основанная на поиске минимального дизъюнкта [12,49], состоит из алгоритма поиска такого дизъюнкта и основного алгоритма метода резолюций.

Таким образом, получим: сложность алгоритма поиска минимального дизъюнкта $=O(M \cdot N)$, а алгоритма метода резолюций = $O(N-1)$, то есть общая сложность = $O(M \cdot N) + O(N-1) = O(M \cdot N)$.

3) Стратегия с поиском похожих предложений

Стратегия, основанная на построении и поиске похожих предложений, состоит из алгоритма поиска таких предложений и основного алгоритма метода резолюций.

Таким образом, получим: сложность алгоритма поиска двух похожих предложений = $O(M^2 + N^2)$, а алгоритма метода резолюций с использованием текущей стратегии = $O(N^2)$, в итоге получаем общую сложность алгоритма = $O(M^2 + N^2) + O(N^2) = O(M^2 + N^2)$.

4) Стратегия, основанная на весах дизъюнктов

Данная стратегия состоит из алгоритма поиска весов дизъюнктов [11] и основного алгоритма метода резолюций.

А значит, получим, сложность алгоритма поиска весов дизъюнктов = $O(N)$, а алгоритма метода резолюций с использованием текущей стратегии = $O(N^2)$, в итоге получаем общую сложность алгоритма = $O(N^2)$.

Таким образом, можно сделать следующие выводы:

1. Стратегия полного перебора является самой низкопроизводительной, так как ее сложность составляет $O(N^N)$. Это означает, что она будет требовать значительно больше времени и ресурсов для обработки больших наборов данных, по сравнению с другими алгоритмами.

2. Стратегия построения рейтинга также имеет высокую сложность $O(M \cdot N^2)$. Это означает, что при увеличении количества элементов (M и N), время, необходимое для выполнения этого алгоритма, будет расти квадратично, что делает его неэффективным для обработки больших объемов данных.

3. Стратегия поиска минимального дизъюнкта и стратегия с весами предложений имеют среднюю эффективность с точки зрения времени исполнения, со сложностями $O(M \cdot N)$ и $O(N^2)$ соответственно. Они могут быть эффективными.

4. Стратегия поиска похожих предложений имеет сложность $O(M^2 + N^2)$. Это означает, что время исполнения будет расти квадратично с увеличением количества элементов. Это может быть проблематично при работе с большими наборами данных.

В целом, лучшим подходом будет выбор стратегии на основе конкретного размера и типа данных.

2.5 Выводы

В данной главе проведен тщательный анализ и систематизация стратегий и алгоритмов управления выводом в контексте метода резолюций, что имеет критическое значение для повышения эффективности и точности в

автоматическом доказательстве теорем — ключевой области искусственного интеллекта и логического программирования.

Таким образом, достигнуты следующие основные цели диссертационного исследования:

1. Разработан комплекс стратегий и соответствующих им алгоритмов управления выводом, позволяющий адаптироваться к различным задачам и оптимизировать процессы автоматического доказательства теорем.

2. Предложены критерии для сравнения и выбора оптимальных стратегий при построении резолютивного вывода, обеспечивающих высокую точность и эффективность решений. Данные критерии учитывают структурные характеристики графа вывода, при этом для каждой стратегии определена оценка сложности алгоритмической реализации, учитывающая количество предложений и среднее количество атомов в предложении. Повышение эффективности метода резолюций в плане быстродействия возможно за счет выбора оптимальной стратегии с учетом структуры исходного множества дизъюнктов и особенностей графа вывода.

Эти достижения являются вкладом в область автоматического доказательства теорем, предоставляя теоретическую базу и практические инструменты для решения сложных задач.

ГЛАВА 3. СТРАТЕГИИ УПРАВЛЕНИЯ ВЫВОДОМ В НЕЧЕТКОМ МЕТОДОМ РЕЗОЛЮЦИЙ

Метод резолюций является одним из ключевых инструментов в области логического вывода и поиска решений. Он нашел широкое применение в традиционной логике, однако с развитием нечеткой логики возникла необходимость адаптировать этот метод к нечетким условиям. Стратегии поиска решений методом резолюций в нечеткой логике открывают новые возможности для моделирования и решения сложных проблем, где данные могут быть нечеткими или неопределенными.

Основная идея метода резолюций заключается в поиске противоречий и логических следствий из имеющихся фактов и правил. В контексте нечеткой логики, где значения истинности могут находиться в интервале от 0 до 1, стратегии поиска решений методом резолюций сталкиваются с особыми вызовами. Нечеткие условия требуют учета степени принадлежности и возможности, а также определения операций резолюции для нечетких предикатов и правил.

Этот пункт исследования посвящен разработке и анализу стратегий поиска решений методом резолюций в нечеткой логике. Предлагается исследовать различные подходы к моделированию нечетких предикатов и правил, а также разработать эффективные методы резолюции для нечетких условий. Это позволит применять метод резолюций в широком спектре приложений, включая принятие решений, интеллектуальный анализ данных и системы управления, где нечеткость и неопределенность играют важную роль.

3.1 Основные понятия нечеткой логики

Нечеткую логику можно определить как алгебру $\langle [0,1], \wedge, \vee, \neg \rangle$, в которой множество значений истинности составляет замкнутый интервал $[0,1]$, а логические связки определяются следующим образом: $a \vee b = \max\{a,b\}$, $a \wedge b = \min\{a,b\}$, $\neg a = \overline{a} = 1 - a$. Заметим, что для операций max и min не

выполняются только законы комплементарности, поэтому в нечеткой логике высказывание, в котором a и $\bar{a}$ встречаются одновременно, оказывается содержательным в отличии от классической логики [9,16,22,78].

Помимо операций max и min, для формализации связок *или* и *и* могут использоваться другие операции, например, связка *или* может задаваться алгебраической суммой $x+y-xy$, а связка *и* – произведением xy [50]. В общем случае, обобщением операций, формализующих связку *и*, является семейство треугольных норм, в то время как связка *или* формализуется треугольными конормами [42,43].

Треугольные нормы (T-нормы) и треугольные конормы (S-конормы) (таблица 3.1) играют ключевую роль в теории нечетких множеств и приложениях, связанных с обработкой нечеткой информации. Они используются для обобщения и формализации понятий "и" и "или" в контексте нечеткой логики [42,43,44,51,52].

Треугольные нормы – это класс операций, который обобщает концепцию пересечения (логического И) для нечетких множеств. Они должны удовлетворять четырем основным свойствам:

a) коммутативность $T(x,y)=T(y,x)$;

b) ассоциативность $T((x,y),z)=T(x,(y,z))$;

c) монотонность: если $x_1 \le x_2$ и $y_1 \le y_2$, то $T(x_1,y_1) \le T(x_2,y_2)$;

d) граничные условия $T(x,1)=T(1,x)=x, T(0,0)=0$.

Треугольные конормы – это класс операций, который обобщает концепцию объединения (логического ИЛИ) для нечетких множеств. Аналогично t-нормам, t-конормы должны удовлетворять свойствам коммутативности, ассоциативности, монотонности, но граничные условия отличаются: $S(x,0)=S(0,x)=x, S(1,1)=1$.

Выбор конкретной t-нормы или t-конормы зависит от контекста задачи и требуемых свойств операций. Например, использование операции минимум (для T-норм) и максимум (для S-конорм) обеспечивает большую

чувствительность к изменению значений по сравнению с логическими операциями И и ИЛИ в классической логике, что делает их предпочтительными во многих приложениях нечеткой логики.

Таблица 3.1 – Известные семейства треугольных норм и конорм [43-45]

№	$T(x,y)$	$S(x,y)$
1	$T_M(x,y)=min(x,y)$	$S_M(x,y)=max(x,y)$
2	$T_P(x,y)=xy$	$S_P(x,y)=x+y-xy$
3	$T_L(x,y)=max(0,x+y-1)$	$S_L(x,y)=min(1,x+y)$
4	$T_D(x,y)=\begin{cases}0, если\ (x,y)\in[0,1)^2\\ min(x,y), иначе\end{cases}$	$S_D(x,y)=\begin{cases}1, если\ (x,y)\in[0,1)^2\\ max(x,y), иначе\end{cases}$
5	$T_0(x,y)=\frac{xy}{x+y-xy}$	$S_{-1}(x,y)=\frac{x+y-2xy}{1-xy}$
6	$T_\alpha(x,y)=\frac{xy}{\alpha+(1-\alpha)(x+y-xy)}$	$S_\beta(x,y)=\frac{(\beta-1)xy+x+y}{1+\beta xy}$
7	$T_\lambda(x,y)=max(0,x+y-1-\lambda(1-x)(1-$ $\lambda\geq-1$	$S_\lambda(x,y)=min(1,x+y+\lambda xy),$ $\lambda\geq-1$
8	$T_w(x,y)=max\left(0,\frac{x+y+wxy-1}{1+w}\right)$ $w>-1$	$S_w(x,y)=min\left(1,x+y-\frac{w}{1-w}xy\right)$ $w>-1$

Каждой треугольной норме соответствует треугольная конорма относительно операции стандартного отрицания. Необходимость использования операций, отличных от max и min обусловлена тем, что данные операции являются «жесткими» в том смысле, что они малочувствительны к изменению своих аргументов. Однако и у других операций имеется проблема, связанная с тем, что если для max и min не выполняется закон комплементарности, то для других операций, помимо комплементарности, не выполняются законы идемпотентности и дистрибутивности. В связи с этим необходимо проявлять аккуратность при выполнении всевозможных преобразований формул [42,43,51,52].

Каждому элементарному высказыванию x поставим в соответствие значение его истинности $t(x)\in[0,1]$, при этом, если значение $t(x)$ близко к 1, то этот факт можно интерпретировать, например, следующим образом: x *почти истинно*, x *в бо́льшей степени истинно*; если значение $t(x)$ близко к 0, то x *почти ложно*; если $x\in[0,0.5)$, то x *скорее ложно*; если $x\in(0.5,1]$, то x *скорее истинно*. Заметим, что для оценки истинности $t(x)$ могут быть использованы другие подходящие модификаторы. Значение $t(x)=0.5$ соответствует максимальной неопределенности.

На множестве значений истинности введем отношение частичного порядка. Пусть x,y – высказывания, $t(x),t(y)\in[0,1]$ – их значения истинности. Будем считать, что эти значения находятся в отношении *частичного порядка* $\gg$, если $0.5\geq t(x)\geq t(y)$ $(t(x)\gg t(y))$ или $0.5\leq t(x)\leq t(y)$ $(t(x)\ll t(y))$. Очевидно, что если $t(x)\in[0,0.5]$, а $t(y)\in[0.5,1]$, то $t(x)$ и $t(y)$ не сравнимы.

Пусть $t(x)\in[0,1]$. *Степенью доверия* к $t(x)$ называется величина

$$c_x=2\cdot(t(x)-0.5)\in[-1,1],$$

а ее модуль $|c_x|$ характеризует уровень определенности высказывания x.

Заметим, что если $t(x)\in[0,0.5)$, то $c_x\in[-1,0)$; если $t(x)\in(0.5,1]$, то $c_x\in(0,1]$; если $t(x)=0.5$, то $c_x=0$ [60].

Пусть F – некоторая формула. При заданной интерпретации I значение истинности каждой переменной определяется однозначно [7]. Выбрав подходящую формализацию логических связок, можно получить значение истинности формулы F. Будем говорить, что интерпретация I удовлетворяет (или не удовлетворяет) формуле F, если значение $t_I(F)$ истинности F в интерпретации I не меньше (не больше) 0.5.

Если $F=\{F_1,...,F_m\}$ – множество формул, то $t_I(F)=t_I(F_1\wedge...\wedge F_m)$.

Говорят, что формула F является *выполнимой* в интерпретации I, если $t(F) > 0.5,$ и *невыполнимой*, если $t(F) < 0.5.$ Если $t(F) = 0.5,$ то интерпретация I одновременно удовлетворяет и опровергает F.

3.2 Понятие нечеткой резольвенты и ее свойства

Пусть $D_1 = p \vee D_1',\ D_2 = \overline{p} \vee D_2',\ res(D_1, D_2) = D_1' \vee D_2'$ [1].

В классической логике действует следующий постулат [5,6,7,78]: истинность множества формул остается неизменной при всех интерпретациях, даже если к высказываниям добавить их резольвенту, т.е.

$$t(D_1 \wedge D_2) = t(D_1 \wedge D_2 \wedge res(D_1, D_2)). \quad (1)$$

Заметим, что из (1) следует, что

$$t(D_1 \wedge D_2) \le t(res(D_1, D_2)). \quad (2)$$

В нечеткой логике резольвента становится или не становится значимым логическим следствием в зависимости от значения истинности переменной p. Найдем условия, при которых в нечеткой логике выполняется соотношение (2), при этом будем считать, что $t(a \vee b) = \max\{t(a), t(b)\}$, $t(a \wedge b) = \min\{t(a), t(b)\},\ t(\overline{a}) = 1 - t(a).$

Положим $t(D_1 \wedge D_2) = \min\{t(D_1), t(D_2)\} = a$. Без ограничения общности будем считать, что

$$t(D_1) = t(p \vee D_1') = \max\{t(p), t(D_1')\} = a, \quad (3)$$

$$t(D_2) = t(\overline{p} \vee D_2') = \max\{t(p), t(D_1')\} = \max\{1 - t(p), t(D_2')\} > a. \quad (4)$$

В соответствии с введенными предположениями возможны следующие ситуации:

для (3) имеем a) $t(p) = a, t(D_1') < a$; b) $t(p) < a, t(D_1') = a$;

для (4) имеем c) $1 - t(p) > a, t(D_2') < a$; d) $1 - t(p) < a, t(D_2') > a$.

При выполнении условий a) и c) получим

$$t\left(res\left(D_1,D_2\right)\right)=t\left(D_1'\vee D_2'\right)=\max\left\{\underbrace{t\left(D_1'\right)}_{<a},\underbrace{t\left(D_2'\right)}_{<a}\right\}<a,$$

и, следовательно, $t\left(D_1\wedge D_2\right)>t\left(res\left(D_1,D_2\right)\right)$, при этом так как $1-t\left(p\right)>a$ и $t\left(p\right)=a$, то $a<0.5$.

При выполнении условий a) и d) получим

$$t\left(res\left(D_1,D_2\right)\right)=t\left(D_1'\vee D_2'\right)=\max\left\{\underbrace{t\left(D_1'\right)}_{<a},\underbrace{t\left(D_2'\right)}_{>a}\right\}>a$$

и $t\left(D_1\wedge D_2\right)<t\left(res\left(D_1,D_2\right)\right)$, причем так как $t\left(p\right)=a$ и $1-t\left(p\right)<a$, то $a>0.5$.

При выполнении условий b) и c) получим

$$t\left(res\left(D_1,D_2\right)\right)=t\left(D_1'\vee D_2'\right)=\max\left\{\underbrace{t\left(D_1'\right)}_{=a},\underbrace{t\left(D_2'\right)}_{<a}\right\}=a,$$

и $t\left(D_1\wedge D_2\right)=t\left(res\left(D_1,D_2\right)\right)$, при этом поскольку $t\left(p\right)<a$ и $1-t\left(p\right)>a$, то $t\left(p\right)<0.5$.

При выполнении условий b) и d) получим

$$t\left(res\left(D_1,D_2\right)\right)=t\left(D_1'\vee D_2'\right)=\max\left\{\underbrace{t\left(D_1'\right)}_{=a},\underbrace{t\left(D_2'\right)}_{>a}\right\}>a$$

и $t\left(D_1\wedge D_2\right)<t\left(res\left(D_1,D_2\right)\right)$, при этом так как $t\left(p\right)<a$ и $1-t\left(p\right)<a$, то $t\left(p\right)>0.5$.

Таким образом, получили следующие результаты:

1) если $t\left(p\right)=a$, $t\left(D_1'\right)<a$, $t\left(D_2'\right)<a$ и $a<0.5$, то соотношение (2) не выполняется;

2) если $t\left(p\right)=a$, $t\left(D_1'\right)<a$, $t\left(D_2'\right)>a$ и $a>0.5$, то соотношение (2) выполняется как строгое неравенство;

3) если $t\left(D_1'\right)=a$, $t\left(D_2'\right)<a$, $t\left(p\right)<0.5$, то соотношение (2) выполняется как равенство;

4) если $t(D_1')=a$, $t(D_2')>a$, $t(p)>0.5$, то соотношение (2) выполняется.

Анализируя полученные результаты, можно сформулировать следующие утверждения:

1) если $t(p)=t(D_1\wedge D_2)>0.5$, то $0.5<t(D_1\wedge D_2)\le t(res(D_1,D_2))$ и, следовательно, $t(D_1\wedge D_2)\ll t(res(D_1,D_2))$;

2) если $0.5<t(p)<t(D_1\wedge D_2)=t(D_1')<t(D_2')$, то $t(res(D_1,D_2))>t(D_1\wedge D_2)$.

$t(D_2')>t(D_1\wedge D_2)=t(p)>0.5$

Утверждение 1. Если $t(p)>0.5$ и хотя бы для одного дизъюнкта $D'\in\{D_1',D_2'\}$ выполняется $t(p)\ll t(D')$, то $t(D_1\wedge D_2)\ll t(res(D_1,D_2))$, т.е. резольвента является значимым логическим следствием.

Утверждение 2. Если $t(p)<0.5$, значения истинности $t(D_1')$ и $t(D_2')$ различны и меньше $1-t(p)$, то $t(D_1\wedge D_2)=t(res(D_1,D_2))$.

Утверждение 3. Если $t(D_1\wedge D_2)>0.5$, то $0.5<t(D_1\wedge D_2)<t(res(D_1,D_2))$, и, следовательно, $t(D_1\wedge D_2)\ll t(res(D_1,D_2))$.

Утверждение 4. Если для переменной p и каждого из дизъюнктов D_1' и D_2' степень истинности меньше 0.5, то резольвента не является значимым логическим следствием.

Известно, что если $t(p\wedge\overline{p})\le t(res(D_1,D_2))$, то $t(D_1\wedge D_2)\le t(res(D_1,D_2))$ причем, если $t(p\wedge\overline{p})=t(res(D_1,D_2))$, то $t(D_1\wedge D_2)=t(res(D_1,D_2))$.

Следовательно, в нечеткой логике резольвенту можно принять в качестве логического следствия лишь тогда, когда переменная p удовлетворяет условию приведенного утверждения.

Пусть $S=\{D_1,...,D_p\}$ – множество дизъюнктов, $R^n(S)$ – резолюционное множество n-го порядка, тогда имеют место следующие утверждения [7]:

1) пусть S – множество дизъюнктов и $D \in R^n(S)$ – некоторый дизъюнкт, тогда если $t(D) \le 0.5$, то в S найдется такой дизъюнкт D', что $t(D') \le 0.5$;

2) пусть S – множество дизъюнктов и $D \in R^n(S)$ – некоторый дизъюнкт, тогда если $t(D) \ge 0.5$, то в S найдется такой дизъюнкт D', что $t(D') \le t(D)$;

3) пусть S – множество дизъюнктов и $D \in R^n(S)$ – некоторый дизъюнкт, тогда если для всех дизъюнктов $D' \in S$ выполняется условие $t(D') > 0.5$, то $t(S) \le t(D)$;

4) пусть S – множество дизъюнктов, тогда если для всех дизъюнктов $D \in S$ выполняется условие $t(D) > 0.5$, то для любого n имеем $t(R^n(S)) = t(S) > 0.5$;

5) пусть S – множество дизъюнктов и $D \in R^n(S)$ – некоторый дизъюнкт, тогда если для всех $D' \in S$ выполняется условие $t(D) \le t(D')$, то в S найдется такой дизъюнкт D'', что $t(D'') \le 0.5$;

6) пусть $S = \{D_1,...,D_n\}$, $\min\{t(D_1),...,t(D_n)\} = \underline{d} > 0.5$, $\max\{t(D_1),...,t(D_n)\} = \overline{d}$, тогда для всех $D \in R^n(S)$ имеем $\underline{d} \le t(D) \le \overline{d}$.

Из последнего утверждения вытекает, что если все значения истинности дизъюнктов больше 0.5, то значение истинности резольвенты всегда заключено между минимальным и максимальным значениями истинности дизъюнктов.

Резольвента, удовлетворяющая приведенным выше условиям, называется *резольвентой Lee* (L-резольвента).

Заметим, что в приведенных выше рассуждениях для формализации логических связок использовались классические операции max и min. Рассмотрим случай, когда конъюнкция задается треугольной нормой T, дизъюнкция – треугольной конормой S, так что пара (T,S) образует пару двойственных операций относительно стандартного отрицания $n(x) = 1 - x$.

Исследуем условие (2) для произвольной пары (T,S), учитывая, что для нее не выполняются законы дистрибутивности, комплементарности и идемпотентности.

Найдем

$$t(D_1)=t(D_1'\vee p)=S\{t(D_1'),t(p)\},\ t(D_2)=t(D_2'\vee \bar{p})=S\{t(D_2'),t(\bar{p})\},$$

$$t(\bar{D}_1)=t(\overline{D_1'\vee p})=t(\bar{D}_1'\wedge \bar{p})=T(t(\bar{D}_1'),t(\bar{p})),$$

$$t(\bar{D}_2)=t(\overline{D_2'\vee \bar{p}})=t(\bar{D}_2'\wedge p)=T(t(\bar{D}_2'),t(p)),$$

$$t(res(D_1,D_2))=t(D_1'\vee D_2')=S(t(D_1'),t(D_2')).$$

Тогда

$$t(D_1\wedge D_2)=T(t(D_1),t(D_2))=1-S(1-t(D_1),1-t(D_2))=1-S(t(\bar{D}_1),t(\bar{D}_2))=$$

$$=1-S(T(t(\bar{D}_1'),t(\bar{p})),T(t(\bar{D}_2'),t(p)))=$$

$$=1-S(1-S(t(D_1'),t(p)),1-S(t(D_2'),t(\bar{p})))=$$

$$=T(S(t(D_1'),t(p)),S(t(D_2'),t(\bar{p})))\le \min(S(t(D_1'),t(p)),S(t(D_2'),t(\bar{p})))\le$$

$$\le \max(S(t(D_1'),t(p)),S(t(D_2'),t(\bar{p})))\le S(S(t(D_1'),t(p)),S(t(D_2'),t(\bar{p})))=$$

$$=S(S(S(t(D_1'),t(p)),t(D_2')),t(\bar{p}))=S(S(S(t(p),t(D_1')),t(D_2')),t(\bar{p}))=$$

$$=S(S(t(p),S(t(D_1'),t(D_2'))),t(\bar{p}))=S(S(S(t(D_1'),t(D_2')),t(p)),t(\bar{p}))=$$

$$=S\left(\underbrace{S(t(D_1'),t(D_2'))}_{t(res(D_1,D_2))},S(t(p),t(\bar{p}))\right).$$

Заметим, что в качестве результата можно получить резольвенту, только если $S(t(p),t(\bar{p}))=0$, так как треугольных конорм имеет место свойство $S(x,0)=x$ для всех $x\in[0,1]$. Соотношение $S(t(p),t(\bar{p}))=0$ выполняется, если $\begin{cases}t(p)=0,\\ t(\bar{p})=0,\end{cases}$ что невозможно. С другой стороны, если

$S\left(t\left(p\right),t\left(\overline{p}\right)\right)=S\left(t\left(D_1'\right),t\left(D_2'\right)\right)$, то для идемпотентных конорм соотношение (2) будет выполнено. Но, согласно [43,51,52], единственной идемпотентной парой среди треугольных норм и конорм является пара $\left(\min,\max\right)$.

Таким образом, доказано следующее

Утверждение 5. Для произвольной пары $\left(T,S\right)$ двойственных треугольных операций, отличных от классических min и max, резольвента Lee не существует.

Рассмотрим случай, когда вместо треугольной конормы S используется операция max. В этом случае

$$t\left(D_1\right)=t\left(D_1'\vee p\right)=\max\left\{t\left(D_1'\right),t\left(p\right)\right\},\ t\left(D_2\right)=t\left(D_2'\vee\overline{p}\right)=\max\left\{t\left(D_2'\right),t\left(\overline{p}\right)\right\}$$

$$t\left(res\left(D_1,D_2\right)\right)=t\left(D_1'\vee D_2'\right)=\max\left(t\left(D_1'\right),t\left(D_2'\right)\right).$$

Рассмотрим

$$t\left(D_1\wedge D_2\right)=T\left(t\left(D_1\right),t\left(D_2\right)\right)\le\min\left\{t\left(D_1\right),t\left(D_2\right)\right\}=$$

$$\min\left\{\max\left\{t\left(D_1'\right),t\left(p\right)\right\},\max\left\{t\left(D_2'\right),t\left(\overline{p}\right)\right\}\right\}=$$

$$\max\left\{\min\left\{t\left(D_1'\right),t\left(D_2'\right)\right\},\min\left\{t\left(D_1'\right),t\left(\overline{p}\right)\right\},\min\left\{t\left(D_2'\right),t\left(p\right)\right\},\min\left\{t\left(p\right),t\left(\overline{p}\right)\right\}\right\}\le$$

$$\le\max\left\{\underbrace{\max\left\{t\left(D_1'\right),t\left(D_2'\right)\right\}}_{t\left(res\left(D_1,D_2\right)\right)},\min\left\{t\left(D_1'\right),t\left(\overline{p}\right)\right\},\min\left\{t\left(D_2'\right),t\left(p\right)\right\},\min\left\{t\left(p\right),t\left(\overline{p}\right)\right\}\right\}$$

Заметим, что результатом последнего выражения будет $t\left(res\left(D_1,D_2\right)\right)$, если будут одновременно выполняться следующие неравенства:

$$\begin{cases}\max\left\{t\left(D_1'\right),t\left(D_2'\right)\right\}\ge\min\left\{t\left(D_1'\right),t\left(\overline{p}\right)\right\},\\ \max\left\{t\left(D_1'\right),t\left(D_2'\right)\right\}\ge\min\left\{t\left(D_2'\right),t\left(p\right)\right\},\\ \max\left\{t\left(D_1'\right),t\left(D_2'\right)\right\}\ge\min\left\{t\left(p\right),t\left(\overline{p}\right)\right\}.\end{cases}$$

С учетом того, что существует всего 24 случая различных упорядочений значений $t\left(D_1'\right),t\left(D_2'\right),t\left(p\right),t\left(\overline{p}\right)$, установлено, что одновременно данные неравенства не выполняются для следующих случаев:

$$t\left(D_1'\right)<t\left(D_2'\right)<t\left(p\right)<t\left(\overline{p}\right),$$

$$t(D_1') < t(D_2') < t(\bar{p}) < t(p),$$
$$t(D_2') < t(D_1') < t(p) < t(\bar{p}),$$
$$t(D_2') < t(D_1') < t(\bar{p}) < t(p).$$

Полученные неравенства можно обобщить следующим образом:

$$\max\{t(D_1'), t(D_2')\} < \min\{t(p), t(\bar{p})\}.$$

Так как $t(\bar{p}) = 1 - t(p)$, то

$$\begin{cases} t(D_1') \\ t(D_2') \end{cases} < t(p) < 0.5 < t(\bar{p}) \text{ и } \begin{cases} t(D_1') \\ t(D_2') \end{cases} < t(\bar{p}) < 0.5 < t(p).$$

Таким образом, доказано следующее

Утверждение 6. Если $\max\{t(D_1'), t(D_2')\} < \min\{t(p), t(\bar{p})\}$, то $t(D_1 \wedge D_2) < t(res(D_1, D_2))$.

Резольвента Lee является или не является значимым логическим следствием, при этом степень истинности переменной p или посылки $D_1 \wedge D_2$ должна быть больше 0.5.

Данное требование является достаточно жестким и ограничивает применение метода резолюций, поэтому в [69] Mukaidono получил обобщение резольвенты Lee, сопряженное со степенью доверия.

Для резольвенты степень доверия вычисляется по формуле

$$c = |c_p| = 2 \cdot (\max\{t(p), t(\bar{p})\} - 0.5).$$

Пусть $D_1 = p \vee D_1'$ и $D_2 = \bar{p} \vee D_2'$ – дизъюнкты. *Нечеткая резольвента* (*резольвента Mukaidono* – М-резольвента) дизъюнктов D_1 и D_2, обозначаемая $res(D_1, D_2)_{c_p}$, где c_p – степень доверия к переменной p, вычисляется по формуле $res(D_1, D_2)_{c_p} = res(D_1, D_2) \vee (p \wedge \bar{p})$, при этом степень ее истинности

$$t\left(res(D_1, D_2)_{c_p}\right) = \max\{t(res(D_1, D_2)), \min\{t(p), t(\bar{p})\}\}.$$

Множество нечетких дизъюнктов невыполнимо тогда и только тогда, когда существует вывод из S пустого дизъюнкта с ненулевой степенью доверия c_p.

Используя пару (T,S) двойственных операций, степень истинности М-резольвенты можно записать в виде

$$t\left(res(D_1,D_2)_{c_p}\right)=S\left\{t\left(res(D_1,D_2)\right),T\left\{t(p),t(\bar{p})\right\}\right\}.$$

Учитывая рассуждения при доказательстве утверждения 5, имеем

$$t(D_1\wedge D_2)\le S\left(S\left(t(D_1'),t(D_2')\right),S\left(t(p),t(\bar{p})\right)\right).$$

Переходя от S к T и предполагая, что $1-T\left(t(\bar{p}),t(p)\right)\le T\left(t(\bar{p}),t(p)\right)$, с учетом свойства монотонности S получим

$$=S\left(S\left(t(D_1'),t(D_2')\right),1-T\left(t(\bar{p}),t(p)\right)\right)\le S\left(S\left(t(D_1'),t(D_2')\right),T\left(t(p),t(\bar{p})\right)\right)=$$

$$S\left(t\left(res(D_1,D_2)\right),T\left(t(p),t(\bar{p})\right)\right)=t\left(res(D_1,D_2)_{c_p}\right),$$

а, следовательно, неравенство (2) выполняется и в этом случае, но степень истинности переменной p и треугольная норма T должны быть такими, чтобы $1-T\left(t(\bar{p}),t(p)\right)\le T\left(t(\bar{p}),t(p)\right)$, что равносильно $T\left(t(\bar{p}),t(p)\right)\ge 0.5$.

Сравнивая два подхода к определению нечеткой резольвенты, заметим, что в подходе Lee [53] база знаний содержит множество дизъюнктов, для каждого из которых известна степень истинности, при этом информации о степени истинности отдельных переменных не требуется. Это означает, что существует множество интерпретаций, согласованных с базой знаний. В подходе Mukaidono [69] подразумевается, что имеются формулы, и заданы степени истинности каждой переменной или атома. Выбрав подходящую интерпретацию логических связок, можно вычислить степень истинности каждого дизъюнкта. Также предполагается вычисление степени доверия к резольвенте в виде c_p для каждой переменной p.

3.3 Алгоритмы резолютивного вывода на основе нечеткой резольвенты

3.3.1 Алгоритм резолютивного вывода на основе резольвенты Lee

Резольвента в интерпретации Lee вычисляется по следующей формуле:

$$res(D_1, D_2)_{Lee} = res(D_1, D_2),$$

при этом она значимой, если для хотя бы одной переменной p, или хотя бы одного дизъюнкта D_1, D_2 степень истинности превосходит 0.5. Таким образом, вычисляются только те резольвенты $res(D_1, D_2)_{Lee}$, которые являются значимыми [7,54].

Алгоритм определения значимости резольвенты

Пусть дано два дизъюнкта D_1, D_2, которые могут образовать резольвенту.

Шаг 1. Если в D_1 или в D_2 содержится атом со степенью истинности > 0.5, то перейти к шагу 2.1, иначе к шагу 2.2.

Шаг 2.1 Резольвента может быть построена и является значимой. Строим резольвенту $res(D_1, D_2)_{Lee} = res(D_1, D_2)$.

Шаг 2.2 Резольвента может быть построена, но не является значимой. Не строим резольвенту.

Метод резолюций в интерпретации Lee

Шаг 1. Положить $k = 0,\ M = S \setminus S_k,\ S_{k+1} = \varnothing$.

Шаг 2. Если во множестве S_k содержится пустой дизъюнкт $\square$, то исходное множество S предложений противоречиво, следовательно, рассуждения являются правильными, и алгоритм завершает свою работу. Иначе перейти к шагу 3.

Шаг 3. Если существуют дизъюнкты $D_i \in M$ и $D_j \in M$ такие, что из них можно получить резольвенту, то перейти к шагу 3.1, иначе – к шагу 5.

Шаг 3.1 Положить $i=1,\ j=n$ (где i, j – индексы элементов массива M_k, а n – количество предложений во множестве M).

Шаг 3.2 Если существует значимая резольвента для $M_k[i]$ и $M_k[j]$, то перейти к шагу 4, иначе перейти к шагу 3.2.1.

Шаг 3.2.1 Если $j>i$, положить $j=j-1$ и перейти к шагу 3.2, иначе перейти к шагу 3.2.2

Шаг 3.2.2 Если $i<n$, положить $i=i+1,\ j=n$ и перейти к шагу 3.2, иначе перейти к шагу 5.2.

Шаг 4. Положить $S_{k+1}=S_{k+1}\cup\left\{res\left(D_i,D_j\right)\right\}$ и перейти к шагу 5.

Шаг 5. Если $\square\in S_{k+1}$, то множество S – невыполнимо (противоречиво), и алгоритм завершает свою работу. Иначе положить $k=k+1\ \ S_{k+1}=\varnothing$, и перейти к шагу 5.1.

Шаг 5.1 Если множество M не пусто, то изменить множество M по следующему правилу: $M=M\setminus M_k[i]\cup M\setminus M_k[j],\ n=n-1$, и перейти к шагу 2, иначе перейти к шагу 5.2.

Шаг 5.2. Алгоритм заканчивает свою работу. Если $\square\in S_{k+1}$, то множество S – невыполнимо (противоречиво), в противном случае – однозначно сказать нельзя.

Во время работы алгоритма метода резолюций и построении резольвенты в интерпретации Lee для каждой пары дизъюнктов, которые могут построить резольвенту, проверяется значимая ли резольвента будет построена. В случае значимости, резольвента строится и алгоритм продолжает свою работу. Таким образом, данный алгоритм не допускает образование лишних незначимых в интерпретации Lee дизъюнктов.

Заметим, что, если задача содержит дизъюнкт с атомом, чья степень доверия равна 0.5, велика вероятность не найти решение.

3.3.2 Алгоритм резолютивного вывода на основе резольвенты Mukaidono

Резольвента в интерпретации Mukaidono вычисляется по следующей формуле:

$$res\left(D_1, D_2\right)_{c_p} = res\left(D_1, D_2\right) \vee \left(p \wedge \overline{p}\right),$$

при этом она сопровождается степенью доверия c_p, которая вычисляется по формуле

$$c = \left|c_p\right| = 2 \cdot \left(\max\left\{t\left(p\right), t\left(\overline{p}\right)\right\} - 0.5\right).$$

Таким образом, будем вычислять резольвенты в следующем порядке: первой строится резольвента $res\left(D_1, D_2\right)_{c_p}$ такая, что ее c_p наибольшая среди всех p , затем по убыванию степени доверия [7,69].

Алгоритм вычисления степени доверия литеры

Пусть дана литера p и ее степень истинности $t(p)$, известна литера $\overline{p}$ и ее степень истинности $t(\overline{p})$.

Шаг 1. Если $t(p) > t(\overline{p})$, то степень доверия p: $c_p = 2 \cdot \left(t\left(p\right) - 0.5\right)$, иначе перейти к шагу 2.

Шаг 2. Степень доверия p: $c_p = 2 \cdot \left(t(\overline{p}) - 0.5\right)$.

Алгоритм расположения степеней доверия литер в порядке убывания

Пусть дана последовательность степеней доверия c_p, n- количество литер, участвущих задаче, а как следствие и количество степеней доверия, а p - определенная литера.

Шаг 1. Положить $i = 0$.

Шаг 2. Если $i >= n - 1$, то останавливаемся, иначе перейти к шагу 3.

Шаг 3. Если $c_p[i] > c_p[i+1]$, то поменять степени доверия и литеры местами, иначе положить $i = i + 1$ и перейти к шагу 3.

Выполнить последовательность действий Шаг 1-3 n-раз, в результате будет получена упорядоченная по убыванию последовательность степеней доверия для литер.

Метод резолюций в интерпретации Mukaidono

Шаг 1. Положить $k=0,\ M=S\setminus S_k,\ S_{k+1}=\varnothing$.

Шаг 2. Если во множестве S_k содержится пустой дизъюнкт $\square$, то исходное множество S предложений противоречиво, следовательно, рассуждения являются правильными, и алгоритм завершает свою работу. Иначе перейти к шагу 3.

Шаг 3. Если существуют дизъюнкты $D_i \in M$ и $D_j \in M$ такие, что из них можно получить резольвенту, то перейти к шагу 3.1, иначе – к шагу 5.

Шаг 3.1 Положить $i=1,\ j=n$ (где $i,\ j-$ индексы элементов массива M_k, а n – количество предложений во множестве M). А также положить i_p равное первому элементу последовательности литер c_p. Перейти к шагу 3.2

Шаг 3.2 Если i_p входит в дизъюнкты $M_k[i]$ и $M_k[j]$, то перейти к шагу 3.3, иначе перейти к шагу 3.4.

Шаг 3.3 Если существует резольвента для $M_k[i]$ и $M_k[j]$, то перейти к шагу 4, иначе перейти к шагу 3.2.1.

Шаг 3.2.1 Если $j>i$, положить $j=j-1$ и перейти к шагу 3.2, иначе перейти к шагу 3.2.2

Шаг 3.2.2 Если $i<n$, положить $i=i+1,\ j=n$ и перейти к шагу 3.2, иначе перейти к шагу 5.

Шаг 3.4 Если $i_p+1>n_{cp}$ (n_{cp}- количество литер, участвующих в задаче), то перейти к шагу 5.2, иначе увеличить i_p на 1, взяв тем самым следующую литеру и перейти к шагу 3.2.

Шаг 4. Положить $S_{k+1}=S_{k+1}\cup\left\{res\left(D_i,D_j\right)\right\}$ и перейти к шагу 5.

Шаг 5. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), и алгоритм завершает свою работу. Иначе положить $k = k+1$ $S_{k+1} = \varnothing$, и перейти к шагу 5.1.

Шаг 5.1 Если множество M не пусто, то изменить множество M по следующему правилу: $M = M \setminus M_k[i] \cup M \setminus M_k[j]$, $n = n-1$, и перейти к шагу 2, иначе перейти к шагу 5.2.

Шаг 5.2. Алгоритм заканчивает свою работу. Если $\square \in S_{k+1}$, то множество S – невыполнимо (противоречиво), в противном случае – однозначно сказать нельзя.

Во время работы алгоритма метода резолюций и построении резольвенты в интерпретации Mukaidono порядок построения резольвент зависит от степени доверия c_p – чем она выше, тем построение резольвенты из дизъюнктов, содержащих литеру p, происходит раньше. Таким образом, данный алгоритм направлен в сторону просмотра литер с наиболее высокой степенью доверия.

Заметим, что резольвента вычисляется по формуле

$$res(D_1, D_2)_{c_p} = res(D_1, D_2) \vee (p \wedge \overline{p}),$$

а значит для корректной работы алгоритма будем преобразовывать резольвенту по операции дизъюнкции и конъюнкции в нечеткой логике:

$$a \vee b = \max\{a, b\};$$
$$a \wedge b = \min\{a, b\};$$

Тогда формула вычисления резольвенты в интерпретации Mukaidono имеет следующий вид:

$$res(D_1, D_2)_{c_p} = res(D_1, D_2) \vee (p \wedge \overline{p}) = \max\{res(D_1, D_2), \min\{p, \overline{p}\}\}$$

3.3.3 Стратегия, основанная на степени сходства

Опираясь на [8,79,85], пусть $A, B-$ обычные подмножества некоторого универсального множества U. При условии, что A и B находятся в общем

положении, т.е. пересекаются, можно выделить следующие естественные точки зрения на способы сравнения этих множеств. Измерение сходства множеств A и B сводится к оценке пересечения $A \cap B$, которое характеризует частичное совпадение множеств. Под степенью несовпадения множества A относительно множества B понимается $A \cap \overline{B}$. Аналогично, степень несовпадения B относительно A есть $\overline{A} \cap B$. Тогда под степенью несходства будем понимать оценку симметрической разности $A \Delta B = \left(A \cap \overline{B} \right) \cup \left(\overline{A} \cap B \right)$.

Для количественной оценки перечисленных отношений между нечеткими множествами введем *скалярный индекс сравнения* (scalar comparison index)

$$SCI(A,B) = f\left(g_1(A * B), g_2(A * B), ..., g_k(A * B\right) \in \left[0,1\right],$$

где $*$ – бинарная операция, задающая комбинацию нечетких множеств; g_i – скалярная оценочная функция нечеткого множества; f – оператор, обеспечивающий нормализацию. Нормализация представляет собой операцию линейного преобразования шкалы, при котором область значений любой ограниченной действительной переменной переводится в $\left[0,1\right]$.

Под *скалярной оценочной функци*ей подразумевается отображение $g : F\left(U\right) \to \left[0,1\right]$, удовлетворяющее следующим свойствам:

a) $g\left(\varnothing\right) = 0$ и $g\left(U\right) = 1$;

b) если $A \subseteq B$, то $g\left(A\right) \le g\left(B\right)$.

Оценочная функция называется *оценочной функцией существования*, если $g\left(A\right) = 0$ тогда и только тогда, когда $A = \varnothing$, и *универсальной*, если $g\left(A\right) = 1$ тогда и только тогда, когда $A = U$.

Заметим, что если g – оценочная функция существования, то $g'\left(\cdot\right) = 1 - g\left(\cdot\right)$ – универсальная оценочная функция и наоборот.

В предположении, что универсальное множество U является конечным, в качестве оценочных можно рассматривать следующие функции:

$g_{\max}(A)=\max\limits_{x\in U}\{\mu_A(x)\}$ – функция существования, которая не является универсальной;

$g_{\min}(A)=\min\limits_{x\in U}\{\mu_A(x)\}$ – универсальная функция, которая не является функцией существования;

$g_{\Sigma}(A)=\dfrac{1}{|U|}\sum\limits_{x\in U}\mu_A(x)$ – функция, которая одновременно является и универсальной, и функцией существования (здесь под мощностью нечеткого множества понимается $|A|=\sum\limits_{x\in U}\mu_A(x)$, поэтому если $A=U$, то $g_{\Sigma}(A)=1$; если $A=\varnothing$, то $g_{\Sigma}(A)=0$; если $A\subset U$, то $g_{\Sigma}(A)\in(0,1)$.

Пусть $A\subset B$, тогда любой элемент из A также принадлежит и B, т.е.

$$\forall x\left((x\in A)\to(x\in B)\right)\Leftrightarrow\forall x\left(\left(x\in\overline{A}\right)\vee(x\in B)\right).$$

Под *индексом включения* $I(A,B)$ понимается скалярная величина, удовлетворяющая следующим свойствам:

a) $I(A,B)=1$ тогда и только тогда, когда $A\subset B$, что равносильно условию $\overline{A}\cup B=U$;

b) $I(A,B)=0$ тогда и только тогда, когда $A\cap B=\varnothing$, и $\overline{A}\cup B=\overline{A}$;

c) $I(A,B)$ зависит от $g\left(\overline{A}\cup B\right)$.

С учетом перечисленных свойств, индекс включения может быть представлен следующим образом:

$$I(A,B)=\frac{g\left(\overline{A}\cup B\right)-g\left(\overline{A}\right)}{1-g\left(\overline{A}\right)},$$

где g – универсальная функция.

Например, если $g=g_{\min}$ и $\cup=\max$, то получим следующую формулу

$$I(A,B)=\frac{\min\limits_{x\in U}\left\{\max\left\{1-\mu_A(x),\mu_B(x)\right\}-\min\left\{1-\mu_A(x)\right\}\right\}}{1-\min\limits_{x\in U}\left\{1-\mu_A(x)\right\}}.$$

Если нечеткие множества A и B – нормальные, а, следовательно, $\max\limits_{x\in U}\{\mu_A(x)\}=1,\ \min\limits_{x\in U}\{1-\mu_A(x)\}=0$, то скалярный индекс включения будет иметь вид

$$I(A,B)=\min_{x\in U}\max\{1-\mu_A(x),\mu_B(x)\}.$$

Можно рассматривать несколько подходов к оценке сходства/несходства двух множеств A и B.

1) Подход, основанный на оценке симметрической разности $A\Delta B$, которая включает элементы, принадлежащие только одному из множеств, тогда если $A\Delta B=\varnothing$, то $A\Delta B=A\cup B$ и множества A и B равны, т.е. максимально схожи.

В соответствии с данным подходом под *индексом сходства* будем понимать скалярную величину $Sim(A,B)$, обладающую следующими свойствами:

a) $Sim(A,B)=1$ тогда и только тогда, когда $A\Delta B=\varnothing$;

b) $Sim(A,B)=0$, если $A\cap B=\varnothing$;

c) $Sim(A,B)=Sim(B,A)$.

Перечисленным свойствам соответствует следующая функция:

$$Sim_1(A,B)=1-\frac{g'(A\triangle B)}{g'(A\cup B)}, \qquad (*)$$

где g' – оценочная функция существования.

Используя представление $A\Delta B=(\overline{A}\cap B)\cup(A\cap\overline{B})$ и классические определения операций $\cap$ и $\cup$, получим формулу

$$Sim_1(A,B)=1-\frac{\max\left\{\min\limits_x\{1-\mu_A(x),\mu_B(x)\},\min\limits_x\{\mu_A(x),1-\mu_B(x)\}\right\}}{\max\limits_x\{\mu_A(x),\mu_B(x)\}}.$$

2) Можно сравнивать дополнительные множества $\overline{A\Delta B}$ и $\overline{A\cup B}$, тогда формула (*) преобразуется к виду

$$Sim_2\left(A,B\right)=\frac{g\left(\overline{A\Delta B}\right)-g\left(\overline{A\cup B}\right)}{1-g\left(\overline{A\cup B}\right)},$$

где g – универсальная оценочная функция.

3) Заметим, что

$$A\Delta B=\left(\overline{A}\cap B\right)\cup\left(A\cap\overline{B}\right)=\varnothing \Leftrightarrow \left[\begin{array}{l}\overline{A}\cap B=\varnothing,\\ A\cap\overline{B}=\varnothing,\end{array}\right. \Leftrightarrow \left[\begin{array}{l}A\cup\overline{B}=U,\\ \overline{A}\cup B=U.\end{array}\right.$$

Из первого условия следует, что $B\subset A$, а из второго – $A\subset B$, тогда при построении индекса сходства можно использовать индекс включения в виде

$$Sim_3\left(A,B\right)=h\left(I\left(A,B\right),I\left(B,A\right)\right),$$

где функция h должна быть симметричной, а также удовлетворять следующим свойствам: $h\left(0,0\right)=0$, $h\left(x,y\right)=1\Leftrightarrow x=y=1$. В качестве h можно использовать, например, $\min$.

Если $I\left(A,B\right)=\min\limits_{x\in U}\max\left\{1-\mu_A\left(x\right),\mu_B\left(x\right)\right\}$, то

$$Sim_3\left(A,B\right)=\min\left\{I\left(A,B\right),I\left(B,A\right)\right\}=$$

$$=\min\left\{\min_{x\in U}\max\left\{1-\mu_A\left(x\right),\mu_B\left(x\right)\right\},\min_{x\in U}\max\left\{\mu_A\left(x\right),1-\mu_B\left(x\right)\right\}\right\}=$$

$$=\min_{x\in U}\min\left\{\max\left\{1-\mu_A\left(x\right),\mu_B\left(x\right)\right\},\max\left\{\mu_A\left(x\right),1-\mu_B\left(x\right)\right\}\right\}.$$

4. Можно сравнивать пересечение множеств $A\cap B$ и их объединение $A\cup B$: если $A\cap B=A\cup B$, то множества равны и, следовательно, максимально схожи.

В этом случае под *индексом сходства* будем понимать отображение $Sim':F\left(U\right)\times F\left(U\right)\to\left[0,1\right]$, которое удовлетворяет следующим аксиомам [55-57, 87-89]:

1) $Sim'\left(A,B\right)=1$ тогда и только тогда, когда $A=B$;

2) если $A\cap B=\varnothing$, то $Sim'\left(A,B\right)=0$;

2) для любых нечетких множеств A и B имеем $Sim'\left(A,B\right)\neq 0$ только если одно из множеств не пустое;

3) $Sim'(A,B) = Sim'(B,A)$;

4) если $A \subseteq B \subseteq C$ или $A \supseteq B \supseteq C$, то $Sim'(A,C) \le \min\{Sim'(A,B), Sim'(B,C)\}$.

Перечисленным свойствам удовлетворяет, например, функция вида

$$Sim'(A,B) = \frac{g(A \cap B)}{g(A \cup B)},$$

где g_Σ – скалярная оценочная функция.

Если $g_\Sigma(A) = \frac{1}{|U|}\sum_{x \in U} \mu_A(x)$ и $|A| = \sum_{x \in U} \mu_A(x)$, откуда $g_\Sigma(A) = \frac{|A|}{|U|}$, то

$$Sim'(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

На основе индекса сходства можно построить индексы для других отношений

a) отрицание сходства $1 - S(A,B)$;

b) несходство $1 - S(\overline{A}, \overline{B})$;

c) отрицание несходства $S(\overline{A}, \overline{B})$.

Как правило, индексы $S(A,B)$ и $S(\overline{A}, \overline{B})$ отражают различную информацию о нечетких множествах, и поэтому в общем случае сходство и отрицание несходства не будут точными синонимами.

Для оценки несходства используется индекс несходства вида $D(A,B) = 1 - S(A,B)$, при этом $D(A,B) = S(\overline{A}, B) = S(A, \overline{B})$.

Распространенный подход к формированию индекса несходства основан на понятии функции расстояния.

Пусть A, B – нечеткие множества, определенные на одно и том же конечном универсальном множестве U, тогда имеет место расстояние Минковского, вычисляемое по формуле

$$\rho_t(A,B)=\left(\frac{1}{|U|}\sum_{x\in U}\left|\mu_A(x)-\mu_B(x)\right|^t\right)^{1/t},$$

где t – настраиваемый параметр.

В этом случае индекс сходства определяется правилом

$$Sim(A,B)=1-\rho_t(A,B).$$

Обоснование метода

Пусть U и V – множества и $f:U\to V$ – некоторое отображение (необязательно однозначное) U в V, так что элементу $x\in U$ соответствует элемент $y\in f(x)$. Пусть A – нечеткое подмножество U с функцией принадлежности μ_A, тогда отображение f индуцирует в V нечеткое множество B с функцией принадлежности μ_B, которая определяется правилом

$$\mu_B(y)=\begin{cases}\max\limits_{x\in f^{-1}(y)}\{\mu_A(x)\}, \text{если } f^{-1}(y)\neq\varnothing;\\ 0, \text{если } f^{-1}(y)\neq\varnothing.\end{cases}$$

Пусть $F(U)$ и $F(V)$ – семейства нечетких подмножеств множеств U и V соответственно. Нечеткое множество $B\in F(V)$ будем называть *условным* на U, и этот факт обозначается как $\mu_B(y\|x)$, где $x\in U$, $y\in V$, если его функция принадлежности зависит от $x\in U$ как от параметра.

Данная функция определяет отображение U в множество нечетких подмножеств, определенных на V. Таким образом, нечеткое подмножество $A\in F(U)$ будет индуцировать нечеткое подмножество $B\in F(V)$ с функцией принадлежности

$$\mu_B(y)=\max_{x\in U}\left\{\min\left\{\mu_B(y\|x),\mu_A(x)\right\}\right\}.$$

Заметим, что каждому нечеткому множеству $A\in F(U)$ отображение f ставит в соответствие нечеткое множество $B\in F(V)$. Такую зависимость

можно выразить условным высказыванием «если x равно a, то в соответствии с функцией f y равно b», то есть $y = f(x)$. Понятие условного нечеткого множества играет именно такую роль.

Пусть X, Y – переменные со значениями в $F(U)$ и $F(V)$ соответственно. Условному высказыванию «если $X = A$, то $Y = B$» соответствует нечеткое отношение $R \in F(U \times V)$ между U и V с функцией принадлежности $\mu_R(x, y)$, которое, по сути, играет роль функции $A \underset{R}{\rightarrow} B$. Тогда функция принадлежности нечеткого множества $B \in F(V)$ определяется правилом

$$\mu_B(y) = \max_{x \in U} \left\{ \min \left\{ \mu_R(x, y), \mu_A(x) \right\} \right\}.$$

Это выражение устанавливает другое представление условных нечетких подмножеств.

Схема метода резолюций, основанного на сходстве/несходстве

Пусть $D_1 = P \vee D_1'$, $D_2 = P' \vee D_2'$ – дизъюнкты, $\varepsilon \in [0,1]$ – пороговое значение, близкое к 1. Резольвента дизъюнктов $res(D_1, D_2) = D_1' \vee D_2'$ существует тогда и только тогда, когда сходство между $\overline{P}$ и P' не меньше заданного порога ε, т.е. $Sim(P, P') \geq \varepsilon$ [2, 90-99].

Особенностью этого определения является то, что вместо контрарных литер здесь используется понятие сходства/несходства.

Метод резолюций основан на правиле вывода, которое называется *дизъюнктивным силлогизмом* $\dfrac{x \vee y, \overline{y}}{x}$, и имеет вид

$$\frac{\begin{array}{l} \textit{посылка } 1 : x \vee y \\ \textit{посылка } 2 : \overline{y} \end{array}}{\textit{заключение} : x}.$$

В предположении, что значениями переменных являются нечеткие множества, в [60] предложено *обобщенное правило нечеткой резолюции*, которое имеет вид

$$\frac{\begin{array}{l}\textit{посылка}\,1:\, x\ \textit{есть}\ A\ \textit{или}\ y\ \textit{есть}\ B\\ \textit{посылка}\,2:\, y\ \textit{есть}\ B'\end{array}}{\textit{заключение}:\, x\ \textit{есть}\ A'}. \qquad (5)$$

Здесь нечеткие множества A и A' определены на одном и том же универсальном множестве U , а нечеткие множества B и B' – на универсальном множестве V .

Говорят, что дизъюнктивный силлогизм (5) имеет место или заключение по правилу нечеткой резолюции имеет место, если B' *близко* к $\overline{B}$ и A' *близко* к A.

Пусть $Sim\left(A', A\right)$ и $Sim\left(B', \overline{B}\right)$ – степени близости соответствующих нечетких множеств в (5). Под степенью доверия схемы будем понимать величину

$$s = Sim\left(A', A\right) \cdot Sim\left(B', \overline{B}\right).$$

Заметим, что посылка 1 и эквивалентные ей формулы формально могут быть представлены нечеткими отношениями $R \in F\left(U \times V\right)$ с учетом способов формализации логических связок. Отношение R задает отображение семейства нечетких подмножеств $F\left(U\right)$ в семейство нечетких подмножеств $F\left(V\right)$. Степень сходства между множеством B и множеством B', которое представлено посылкой 2, используется для модификации отношения R. Тогда для получения заключения A', близкого к A, применяется формула

$$\mu_{A'}\left(x\right) = \max_{y \in V}\left\{\min\left\{\mu_R\left(x, y\right), \mu_{B'}\left(y\right)\right\}\right\}, \forall x \in U.$$

Сформулируем процедуру для нахождения заключения A'.

1. Пусть $A \in F(U)$, $B \in F(V)$. Учитывая, что $\vee = \max$, посылке 1 поставить в соответствие нечеткое отношение $R \in F(U \times V)$ по правилу $\mu_R(x,y) = \max\limits_{(x,y)\in U\times V} \{\mu_A(x), \mu_B(y)\}$.

2. Для нечетких множеств $\overline{B}$ и B' вычислить индекс сходства $s = Sim(\overline{B}, B')$, используя подходящее определение.

3. Вычислить модифицированное условное отношение $R(A, B \| B')$ с функцией принадлежности

$$\mu_{R(A,B\|B')}(x,y) = s \to \mu_{R(A,B)}(x,y) = \min\left\{s, \mu_{R(A,B)}(x,y)\right\},$$

где $a \to b \Leftrightarrow \min\{a,b\}$.

4. Используя операцию проектирования, найти нечеткое множество A', функция принадлежности которого вычисляется следующим образом:

$$\mu_{A'}(x) = \max_{y\in V} \mu_{R(A,B\|B')}(x,y).$$

3.3.4 Анализ сложности алгоритмов приведенных стратегий

Анализ сложности новых стратегий для нечеткой логики представлен в табл. 3.2.

Пусть N – количество предложений, а M – среднее количество атомов в предложении. В таблице 3.2 представлена оценка сложности описанных алгоритмов.

Таблица 3.2 – Оценка сложности алгоритмов представленных стратегий и полного перебора

Стратегия в интерпретации Lee	**Стратегия в интерпретации Mukaidono**	**Стратегия на основе сходства**
$O(M+N^2)$	$O(M+N^2)$	$O(M+N^2)$

Для подсчета оценки сложности стратегий, разобьем общий алгоритм на составляющие, после чего вычислим сумму.

1) Стратегия в интерпретации Lee

Стратегия, основанная на интерпретации Lee [7,53], состоит из алгоритма определения степени доверия дизъюнкта и основного алгоритма метода резолюций с использованием текущей стратегии.

Таким образом, получим: алгоритм определения степени доверия дизъюнкта имеет сложность $O(M)$, а алгоритм метода резолюций с использованием текущей стратегии $O(N^2)$, а, следовательно, общая сложность составляет $O(M) + O(N^2) = O(M + N^2)$.

2) Стратегия в интерпретации Mukaidono

Стратегия, основанная на интерпретации Mukaidono [7,69], состоит из алгоритма определения и выбора подходящих дизъюнктов для построения значимой резольвенты и основного алгоритма метода резолюций.

Таким образом, общая сложность $O(M + N^2)$.

3) Стратегия на основе сходства

Стратегия на основе сходства [8], состоит из алгоритма подсчета степени сходства атома и предложений, определения наличия пустого дизъюнкта и основного алгоритма метода резолюций.

Таким образом, получим: сложность алгоритма подсчета степени сходства атома и предложений есть $O(1) + O(M)$, а алгоритма определения наличия пустого дизъюнкта есть $O(1)$, в итоге получаем общую сложность алгоритма $O(N^2) + O(1) + O(M) = O(M + N^2)$.

Из текущих подсчетов видно, что все три стратегии алгоритмов (стратегия в интерпретации Lee, стратегия в интерпретации Mukaidono и стратегия на основе сходства) имеют одинаковую временную сложность $O(M + N^2)$.

Это означает, что скорость работы каждого алгоритма увеличивается пропорционально количеству элементов M и квадрату количества элементов N. При этом, все три алгоритма должны показывать схожую производительность, предполагая, что константы, скрытые в нотации O, не значительно отличаются. Если они существенно различаются, один алгоритм

может быть быстрее другого несмотря на то, что их асимптотическая сложность одинакова.

Также важно отметить, что эта оценка сложности предполагает худший сценарий. В некоторых случаях, на практике алгоритмы могут работать быстрее.

3.4 Иллюстративные примеры

3.4.1 Пример решения задачи алгоритмом в интерпретации Lee

Пусть задано следующее множество предложений:

$$\{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}.$$

Докажем или опровергнем его противоречивость.

Рассматриваем данную задачу в рамках нечеткой логики, а значит каждая литера имеет степени истинности: $t(C) = 0.5,\ t(O) = 0.4, t(Z) = 0.3, t(Y) = 0.8$.

Результаты работы алгоритмов представлены в таблице 3.3 и на рис. 3.1.

Таблица 3.3 – Результаты работы алгоритма в интерпретации Lee

$k = 0$ $M = \{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}$ $S_0 = \varnothing$
S_0 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
$D_i = M_0[0] = C(x) \vee O(x),\ D_j = M_0[1] = Z(y) \vee \neg C(y)$ Резольвента существует, но не является значимой
$D_i = M_0[1] = Z(y) \vee \neg C(y),\ D_j = M_0[3] = \neg Z(A)$ Резольвента существует, и она является значимой
$S_1 = \{\neg C(A)\}$
$k = 1,\ S_1 = \varnothing$
S_1 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
$D_i = M_1[2] = Y(A),\ D_j = M_1[4] = \neg O(z) \vee \neg Y(z)$ Резольвента существует, и она является значимой
$S_2 = \{\neg C(A), \neg O(A)\}$
$k = 2, S_2 = \varnothing$
S_2 не содержит пустого дизъюнкта
В M есть унифицируемые литералы

$D_i = M_2[0] = C(x) \vee O(x),\ D_j = M_2[6] = \neg O(A)$ Резольвента существует, и она является значимой
$S_3 = \{\neg C(A), \neg O(A), C(A)\}$
$k = 3,\ S_3 = \varnothing$
S_3 не содержит пустого дизъюнкта
В M есть унифицируемые литералы
Нет дизъюнктов, которые могут образовать значимую резольвенту, алгоритм завершает работу

Заметим, что данная формула является выполнимой, тем не менее алгоритм в интерпретации резольвенты Lee определить этого не смог. Причиной тому наличие литеры, степень истинности которой равна 0.5.

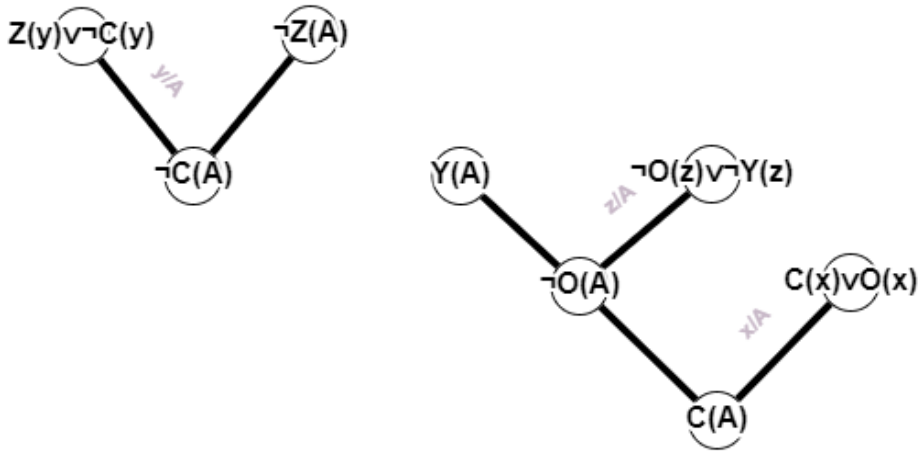


Рис. 3.1 – Дерево вывода, соответствующее алгоритму метода резолюций в интерпретации Lee

3.4.2 Пример решения задачи алгоритмом в интерпретации Mukaidono

Пусть задано следующее множество предложений:

$$\{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}.$$

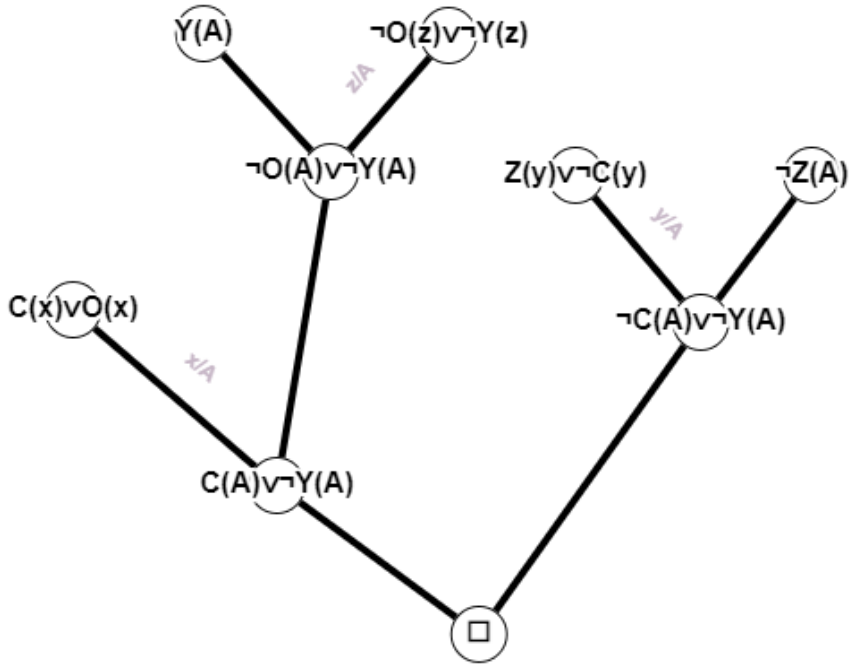
Докажем или опровергнем его противоречивость.

Рассматриваем данную задачу в рамках нечеткой логики, а значит каждая литера имеет степени истинности: $t(C) = 0.5, t(O) = 0.4, t(Z) = 0.3, t(Y) = 0.8$.

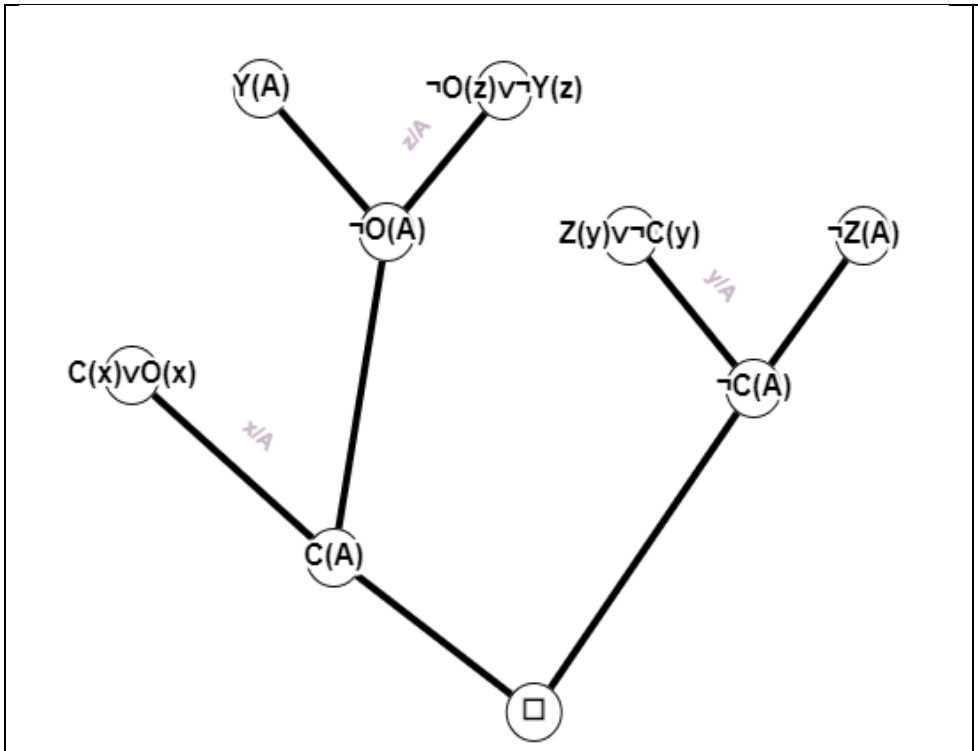
Ниже представлено решение задачи алгоритмом в интерпретации резольвенты Mukaidono (табл. 3.4, рис. 3.2).

Таблица 3.4 – Результаты работы алгоритма в интерпретации Mukaidono

$C=\{W(x)\vee O(x), Z(y)\vee\neg W(y), Y(A), \neg Z(A), \neg O(z)\vee\neg Y(z)\}$ $c_p(Y)=2\cdot(0.8-0.5)=0.6,\ c_p(Z)=2\cdot(0.7-0.5)=0.4$ $c_p(O)=2\cdot(0.6-0.5)=0.2,\ c_p(W)=2\cdot(0.5-0.5)=0$
$k=0, R_0=\varnothing;\ R_0$ не содержит пустого дизъюнкта; в C есть унифицируемые литералы
$D_i=Y(A),\ D_j=\neg O(z)\vee\neg Y(z)$
$\neg O(A)\vee\neg Y(A)=\neg O(A),\ R_1=\{\neg O(A)\}$
$k=1;\ R_1$ не содержит пустого дизъюнкта; в C есть унифицируемые литералы
$D_i=Z(y)\vee\neg W(y),\ D_j=\neg Z(A)$
$\neg W(A)\vee\neg Y(A)=\neg W(A),\ R_2=\{\neg O(A), \neg W(A)\}$
$k=2, R_2$ не содержит пустого дизъюнкта, в C есть унифицируемые литералы
$D_i=W(x)\vee O(x),\ D_j=\neg O(A),\ R_3=\{\neg O(A), \neg W(A), W(A)\}$
$k=3,\ R_3$ не содержит пустого дизъюнкта; в C есть унифицируемые литералы
$D_i=\neg W(A),\ D_j=W(A),\ R_4=\{\neg O(A), \neg W(A), W(A), \square\}$
R_4 содержит пустой дизъюнкт – множество C противоречиво.



а) *дерево до операции дизъюнкции*



б) *дерево после операции дизъюнкции*

Рис. 3.2 –Дерево вывода, соответствующее алгоритму метода резолюций в интерпретации Mukaidono

3.4.3 Пример решения задачи алгоритмом сходства

Пусть даны следующие предложения:

$$\{P_1 \vee P_2, P_3 \vee P_4, P_5 \vee P_6, P_7 \vee P_8, P_9 \vee P_{10}\}$$

при этом

$$P_1 \sim \neg P, P_2 \sim Q, P_3 \sim \neg Q, P_4 \sim R, P_5 \sim \neg R, P_6 \sim S, P_7 \sim \neg S, P_8 \sim T, P_9 \sim \neg T, P_{10} \sim P,$$

где $\sim =$ “близко”.

$$P_1 = 0.1, P_2 = 0.2, P_3 = 0.9, P_4 = 0.3, P_5 = 0.8, P_6 = 0.3, P_7 = 0.8, P_8 = 0.2, P_9 = 0.9, P_{10} = 0.9$$

Используя формулу для вычисления индекса сходства:

$$Sim(A,B) = 1 - \frac{\max\left\{\min\limits_x\left\{1-\mu_A(x), \mu_B(x)\right\}, \min\limits_x\left\{\mu_A(x), 1-\mu_B(x)\right\}\right\}}{\max\limits_x\left\{\mu_A(x), \mu_B(x)\right\}}$$

и следующее определение:

Пусть $D_1 = P \vee D_1'$, $D_2 = P' \vee D_2'$ – дизъюнкты, $\varepsilon \in [0,1]$ – пороговое значение, близкое к 1. Резольвента дизъюнктов $res(D_1, D_2) = D_1' \vee D_2'$

существует тогда и только тогда, когда сходство между $\overline{P}$ и P' не меньше заданного порога ε, т.е. $Sim(P,P') \geq \varepsilon,\ \ 0 < \varepsilon < 0.5$

Докажем или опровергнем противоречивость исходного множества (табл. 3.5, рис. 3.3).

Таблица 3.5 – Результаты работы алгоритма на основе сходства

$C = \{P_1 \vee P_2, P_3 \vee P_4, P_5 \vee P_6, P_7 \vee P_8, P_9 \vee P_{10}\}$ $P_1 = 0.1, P_2 = 0.2, P_3 = 0.9, P_4 = 0.3, P_5 = 0.8, P_6 = 0.3, P_7 = 0.8, P_8 = 0.2, P_9 = 0.9, P_{10} = 0.9$ $Sim(A,B) = 1 - \frac{\max\left\{\min\limits_x\{1-\mu_A(x), \mu_B(x)\}, \min\limits_x\{\mu_A(x), 1-\mu_B(x)\}\right\}}{\max\limits_x\{\mu_A(x), \mu_B(x)\}}$
$D_1 = P_1 \vee P_2,\ D_2 = P_3 \vee P_4$ $Sim(P_1,P_3) = 1 - \frac{\max\{\min\{1-0.1, 0.9\}, \min\{0.1, 1-0.9\}\}}{\max\{0.1, 0.9\}} = 1 - \frac{\max\{0.9, 0.1\}}{\max\{0.1, 0.9\}} = 0$ $Sim(P_1,P_4) = 1 - \frac{\max\{\min\{1-0.1, 0.3\}, \min\{0.3, 1-0.9\}\}}{\max\{0.3, 0.9\}} = 1 - \frac{\max\{0.3, 0.1\}}{\max\{0.3, 0.9\}} = \frac{2}{3}$ $Sim(P_2,P_3) = 1 - \frac{\max\{\min\{1-0.2, 0.9\}, \min\{0.2, 1-0.9\}\}}{\max\{0.2, 0.9\}} = 1 - \frac{\max\{0.8, 0.1\}}{\max\{0.2, 0.9\}} = \frac{1}{9}$ $Sim(P_2,P_4) = 1 - \frac{\max\{\min\{1-0.2, 0.3\}, \min\{0.2, 1-0.3\}\}}{\max\{0.2, 0.3\}} = 1 - \frac{\max\{0.3, 0.2\}}{\max\{0.2, 0.3\}} = 0$
$res(D_1, D_2) = res(P_1 \vee P_2, P_3 \vee P_4) = P_1 \vee P_4$ $P_1 \sim \neg P, P_4 \sim R$
$D_1 = P_5 \vee P_6,\ D_2 = P_7 \vee P_8$ $Sim(P_5,P_7) = 1 - \frac{\max\{\min\{1-0.8, 0.8\}, \min\{0.8, 1-0.8\}\}}{\max\{0.8, 0.8\}} = 1 - \frac{\max\{0.2, 0.2\}}{\max\{0.8, 0.8\}} = \frac{6}{8}$ $Sim(P_5,P_8) = 1 - \frac{\max\{\min\{1-0.8, 0.1\}, \min\{0.8, 1-0.1\}\}}{\max\{0.8, 0.1\}} = 1 - \frac{\max\{0.1, 0.8\}}{\max\{0.8, 0.1\}} = 0$ $Sim(P_6,P_7) = 1 - \frac{\max\{\min\{1-0.3, 0.8\}, \min\{0.3, 1-0.8\}\}}{\max\{0.3, 0.8\}} = 1 - \frac{\max\{0.7, 0.2\}}{\max\{0.3, 0.8\}} = \frac{1}{8}$ $Sim(P_6,P_8) = 1 - \frac{\max\{\min\{1-0.3, 0.1\}, \min\{0.3, 1-0.1\}\}}{\max\{0.3, 0.1\}} = 1 - \frac{\max\{0.1, 0.3\}}{\max\{0.3, 0.1\}} = 0$

$res(D_1, D_2) = res(P_5 \vee P_6, P_7 \vee P_8) = P_5 \vee P_8$

$P_5 \sim \neg R, P_8 \sim T$

$D_1 = P_1 \vee P_4,\ D_2 = P_5 \vee P_8$

$$Sim(P_1, P_5) = 1 - \frac{\max\left\{\min\{1-0.1, 0.3\}, \min\{0.1, 1-0.3\}\right\}}{\max\{0.1, 0.3\}} = 1 - \frac{\max\{0.3, 0.1\}}{\max\{0.1, 0.3\}} = 0$$

$$Sim(P_1, P_8) = 1 - \frac{\max\left\{\min\{1-0.1, 0.1\}, \min\{0.1, 1-0.1\}\right\}}{\max\{0.1, 0.1\}} = 1 - \frac{\max\{0.1, 0.1\}}{\max\{0.1, 0.1\}} = 0$$

$$Sim(P_4, P_5) = 1 - \frac{\max\left\{\min\{1-0.3, 0.8\}, \min\{0.3, 1-0.8\}\right\}}{\max\{0.3, 0.8\}} = 1 - \frac{\max\{0.7, 0.2\}}{\max\{0.3, 0.8\}} = \frac{1}{8}$$

$$Sim(P_4, P_8) = 1 - \frac{\max\left\{\min\{1-0.3, 0.1\}, \min\{0.3, 1-0.1\}\right\}}{\max\{0.3, 0.1\}} = 1 - \frac{\max\{0.1, 0.3\}}{\max\{0.3, 0.1\}} = 0$$

$res(D_1, D_2) = res(P_1 \vee P_4, P_5 \vee P_8) = P_1 \vee P_8$

$P_1 \sim \neg P, P_8 \sim T$

$D_1 = P_1 \vee P_8,\ D_2 = P_9 \vee P_{10}$

$$Sim(P_1, P_9) = 1 - \frac{\max\left\{\min\{1-0.1, 0.9\}, \min\{0.1, 1-0.9\}\right\}}{\max\{0.1, 0.9\}} = 1 - \frac{\max\{0.9, 0.1\}}{\max\{0.1, 0.9\}} = 0$$

$$Sim(P_1, P_{10}) = 1 - \frac{\max\left\{\min\{1-0.1, 0.8\}, \min\{0.1, 1-0.8\}\right\}}{\max\{0.1, 0.8\}} = 1 - \frac{\max\{0.8, 0.1\}}{\max\{0.1, 0.8\}} = 0$$

$$Sim(P_8, P_9) = 1 - \frac{\max\left\{\min\{1-0.2, 0.9\}, \min\{0.2, 1-0.9\}\right\}}{\max\{0.2, 0.9\}} = 1 - \frac{\max\{0.8, 0.1\}}{\max\{0.2, 0.9\}} = \frac{1}{9}$$

$$Sim(P_8, P_{10}) = 1 - \frac{\max\left\{\min\{1-0.2, 0.9\}, \min\{0.2, 1-0.9\}\right\}}{\max\{0.2, 0.9\}} = 1 - \frac{\max\{0.8, 0.2\}}{\max\{0.2, 0.9\}} = \frac{1}{9}$$

$res(D_1, D_2) = res(P_1 \vee P_8, P_9 \vee P_{10}) = P_1 \vee P_{10}$

$P_1 \sim \neg P, P_{10} \sim P$ – останов, решение найдено

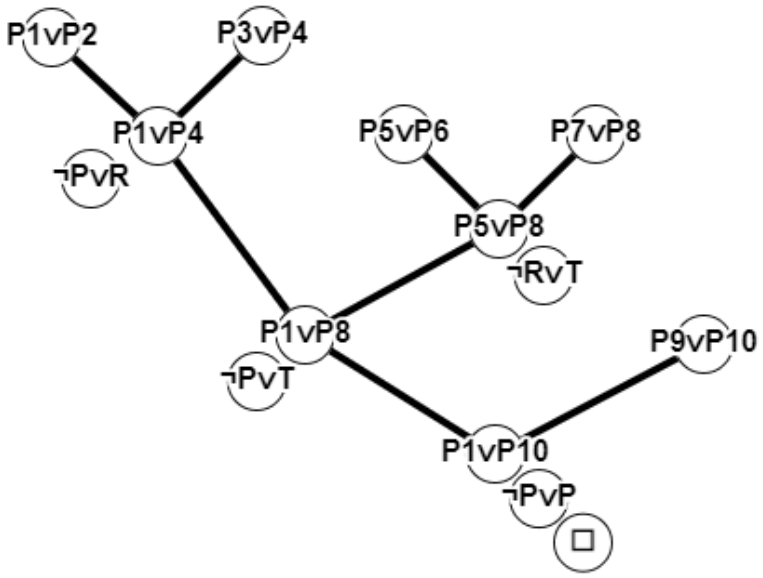


Рис. 3.3 – Дерево вывода, соответствующее алгоритму метода резолюций, основанном на сходстве

3.5 Выводы

В исследовании, посвященном анализу и применению нечеткой логики в методе резолюций, были достигнуты следующие результаты:

1. Особое внимание было уделено анализу возможностей нечеткой логики в формировании свойств и характеристиках нечеткой резольвенты, особенно в контексте использования треугольных норм и конорм. Эти элементы играют важную роль в операционализации логических операций "И" и "ИЛИ" в рамках нечеткой логики, подчеркивая значимость и сложность интеграции нечетких множеств и логических операций для глубокого понимания логического вывода в условиях неопределенности.

2. Был представлен широкий спектр алгоритмических решений, направленных на реализацию нечеткого логического вывода. Эти алгоритмы, основанные на принципах нечеткой логики, предназначены для обработки и анализа нечеткой информации, позволяя формировать резолютивный вывод на основе нечетких предпосылок и правил. В исследовании особое внимание уделено различным подходам к построению резолютивного вывода, включая адаптированные к условиям нечеткости резолюции, раскрывая новые

перспективы для оптимизации логического вывода в условиях неопределенности.

Таким образом, исследование выявило важность и эффективность интеграции нечетких множеств и логических операций в процесс логического вывода, а также разработки и применения адаптированных алгоритмов для решения сложных задач в условиях нечеткости и неопределенности.

ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СТРАТЕГИЙ ПОИСКА РЕШЕНИЙ МЕТОДОМ РЕЗОЛЮЦИЙ В КЛАССИЧЕСКОЙ И НЕЧЕТКОЙ ЛОГИКЕ

Реализация стратегий поиска решений методом резолюций является одним из важных аспектов в области искусственного интеллекта и логического программирования. Этот метод, разработанный в 1960-х годах [78], предоставляет эффективный и формально обоснованный способ нахождения решений в том случае, если для представления знаний используется логическая модель, основанная на теории предикатов первого порядка. В данном разделе описана программная реализация стратегий поиска решений методом резолюций как в классической, так и в нечеткой логике.

В классической логике метод резолюций основан на применении логического закона «дизъюнктивного силлогизма» для вывода новых утверждений из имеющихся фактов и правил [5,78]. Эти правила преобразуются в формулы логики предикатов. Программная реализация стратегий поиска решений методом резолюций в классической логике включает в себя разработку алгоритмов и структур данных, а также эффективного представления знаний и правил [55,78].

Нечеткая логика является расширением классической логики, которое позволяет работать с нечеткими или неопределенными высказываниями. В нечеткой логике метод резолюций также может быть применен для решения различных задач, связанных с нечеткими множествами и нечеткими правилами вывода [36,37,53,55]. Программная реализация стратегий поиска решений методом резолюций в нечеткой логике требует специфических алгоритмов и структур данных, а также обработки и представления нечеткой информации [7,8,60,63,67,81].

В данном разделе будут рассмотрены основные подходы к программной реализации стратегий поиска решений методом резолюций в классической и нечеткой логике. Будут приведены примеры их применения. Это позволит

лучше понять принципы работы метода резолюций и его потенциал в решении сложных логических задач как в классической, так и в нечеткой логике.

Для реализации алгоритмов был выбран язык C++. Язык программирования C++ предоставляет несколько преимуществ, которые делают его особенно подходящим для реализации алгоритмов, включая те, которые используются в методе резолюций. Несмотря на то, что выбор языка программирования в значительной степени зависит от конкретных требований проекта и личного предпочтения разработчика, есть несколько причин, по которым C++ может быть отличным выбором:

1) Производительность и эффективность использования ресурсов: C++ является одним из самых быстрых языков программирования, благодаря прямому доступу к низкоуровневым ресурсам системы и возможности ручного управления памятью. Это делает его подходящим для сложных алгоритмов, которые требуют большой производительности и эффективного использования ресурсов, таких как методы резолюций.

2. C++ поддерживает технологию объектно-ориентированного программирвания (ООП), что упрощает организацию сложных алгоритмов и структур данных, а также повышает читаемость и модульность кода. Концепции ООП, такие как наследование, инкапсуляция и полиморфизм, могут быть очень полезны при проектировании алгоритмов.

3. STL (Standard Template Library) в C++ предоставляет широкий спектр готовых к использованию структур данных (например, векторы, списки, очереди, стеки и т. д.) и алгоритмов (сортировка, поиск, и т. д.). Это упрощает разработку и сокращает время на написание кода.

4. Портативность – C++ является кроссплатформенным языком программирования, что означает, что можно написать код на одной платформе и запустить его на другой с минимальными или никакими изменениями.

4.1 Программная реализация метода резолюций в классической логике

В данном разделе рассматривается программная реализация стратегий поиска решений методом резолюций в классической логике [20]. Классическая логика, основанная на принципах истинности и ложности, является одной из основных логических систем, применяемых в информатике и искусственном интеллекте.

Разработка программ, основанных на стратегиях классической логики, позволяет решать задачи формализации и логического вывода в различных областях, таких как автоматизированное доказательство теорем, анализ и верификация программного кода, обработка естественного языка и другие. Благодаря своей математической строгости и надежности, классическая логика остается актуальным и востребованным инструментом в сфере искусственного интеллекта и информатики.

В листинге 4.1 представлены структуры для атома и дизъюнкта: каждый атом обладает или не обладает отрицанием, а также имеет наименование, каждый дизъюнкт состоит из нескольких атомов. Соответственно, программа принимает данные в вектор дизъюнктов, который состоит из некоторого количества атомов.

Листинг 4.1 – Структуры атома и дизъюнкта

```
struct Atom {
    int otr;
    string s;
    float w;
};

struct Clause {
    int si;
    vector<Atom> a;
    int rating;
};
```

В листингах 4.2-4.4 описаны функции для работы с резольвентой.

Функция 4.2 bool notEquals(Clause a, Clause b) возвращает bool-значение true, если два дизъюнкта не равны. Равенство происходит через сравнение каждого атома и его отрицания.

Функция 4.3 bool resolventExists(Clause d, Clause ad) возвращает bool-значение true, если резольвента может быть построена между дизъюнктами d и ad.

Функция 4.4 Clause resolvent(Clause d, Clause ad) возвращает резольвенту, которую образовали два дизъюнкта d и ad. В резолютирующий дизъюнкт попадают все атомы, которые не являются атомом X и атомом не X.

Листинг 4.2 – Функция, которая возвращает true, если два дизъюнкта отличаются

```
bool notEquals(Clause a, Clause b) {
    if (a.si!=b.si) return true;
    for (int i=0; i<a.si; i++) {
        if (a.a[i].s!=b.a[i].s || a.a[i].otr!=b.a[i].otr)
           return true;
    }
    return false;
}
```

Листинг 4.3 – Функция, которая возвращает true, если два дизъюнкта могут построить резольвенту

```
bool resolventExists(Clause d, Clause ad) {
    for (int i=0; i<d.si; i++) {
        for (int j=0; j<ad.si; j++) {
            if (d.a[i].s==ad.a[j].s
                && d.a[i].otr!=ad.a[j].otr)
              return true;
        }
    }
    return false;
}
```

Листинг 4.4 – Функция, строит резольвенту и возвращает ее

```
Clause resolvent(Clause d, Clause ad) {
    Clause res;
    res.si=0;
    for (int i=0; i<ad.si; i++) {
        for (int j=0; j<d.si; j++) {
            if (d.a[j].s==ad.a[i].s
                && d.a[j].otr!=ad.a[i].otr) {
                for (int i2=0; i2<ad.si; i2++) {
                    if (i2!=i) {
                        Atom a;
```

```
                        a.s=ad.a[i2].s;
                        a.otr=ad.a[i2].otr;
                        res.a.push_back(a);
                        res.si++;
                    }
                }
                for (int j2=0; j2<d.si; j2++) {
                    if (j2!=j) {
                        Atom a;
                        a.s=d.a[j2].s;
                        a.otr=d.a[j2].otr;
                        res.a.push_back(a);
                        res.si++;
                    }
                }
            }
        }
    }
    return res;
}
```

4.1.1 Реализация стратегии поиска минимального дизъюнкта

Опираясь на описание стратегии поиска минимального дизъюнкта (раздел 2.2.2) и схему реализации алгоритма (рис 4.1), продумана программная реализация (листинг 4.5). Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт, функция поиска минимального дизъюнкта, а также функции для работы с резольвентой: поиск дизъюнктов, которые могут построить резольвенту и построение самой резольвенты.

В листинге 4.6 описана функция, которая возвращает минимальный дизъюнкт в наборе m, который является вектором дизъюнктов.

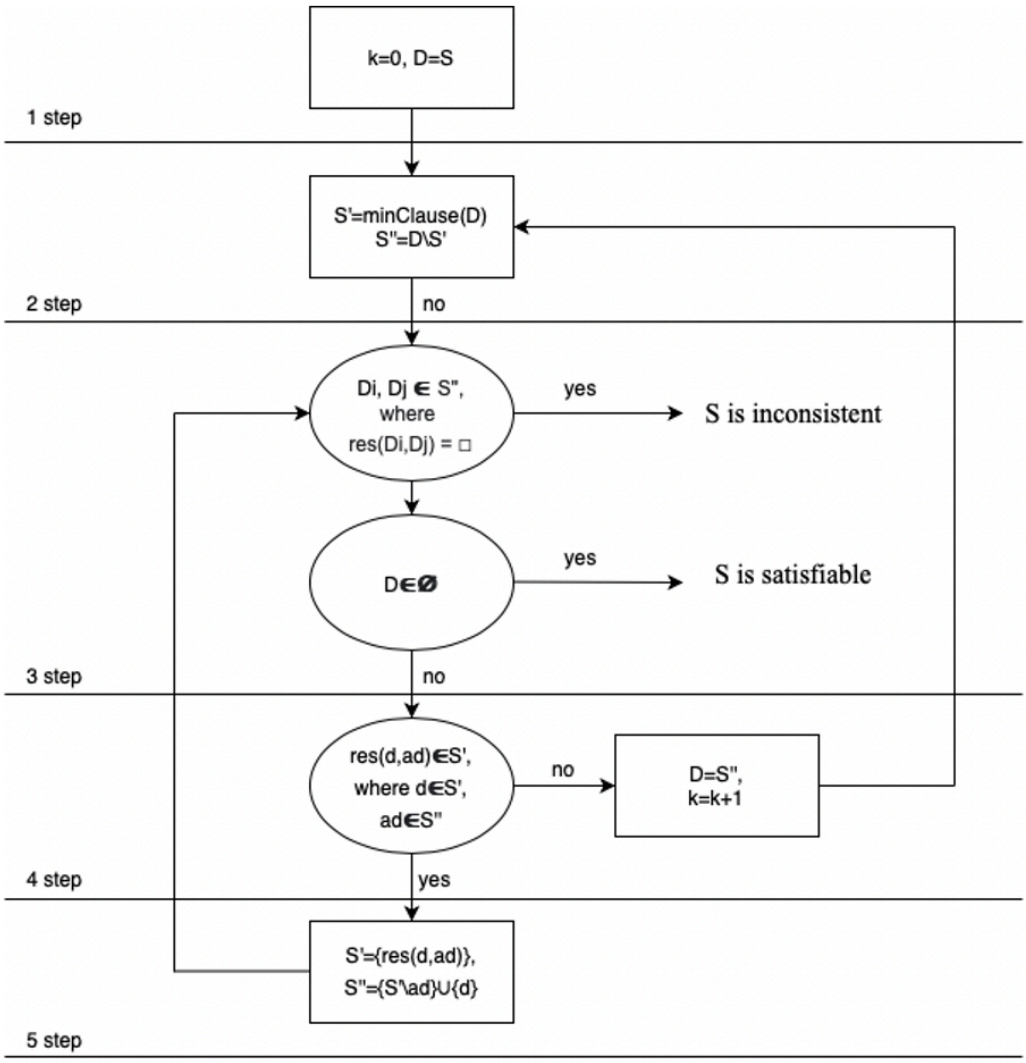


Рис. 4.1 – Схема работы алгоритма

Листинг 4.5 – Функции реализиации алгоритма на основе минимального дизъюнкта

```
Clause findMinD(vector<Clause> &m, int n);
string MinimalClause(vector<Clause> vD, int n);
int MinimalClauseWithDescription(vector<Clause> vD, int n);
```

Листинг 4.6 – Функция, которая возвращает минимальный дизъюнкт в наборе

```
Clause findMinD(vector<Clause> &m, int n){
    Clause minClause=m[0]; int minSi=minClause.si;
    for (int i=0; i<n; i++) {
        if (m[i].si<minSi) {
            minSi=m[i].si;
            minClause=m[i];
        }
    }
    return minClause;
}
```

Последовательно имплементируются шаги алгоритма: выполняем цикл до тех пор, пока, не найдем пустой дизъюнкт (а значит множество противоречиво) или множество, в котором ищем минимальный дизъюнкт, не станет пустым (а значит множество выполнимо).

Внутри функции собираем множество S1 и множество S2, которые в программе являются векторами дизъюнктов. Перебираем значения из векторов, если d из вектора S1 и ad из вектора S2 могут построить резольвенту (**if** (resolventExists(s1[i1], s2[i2])), то она строится (Clause res = resolvent(s1[i1], s2[i2])), если был получен пустой дизъюнкт, то программа завершает свою работу, получаем ответ (cout<<"\n\n ANSWER = INCONSISTENT\n";). Иначе изменяем вектора S1 и S2.

Переменная LIMIT указана для предотвращения проблемы зацикливания, ограничение «по умолчанию» стоит на 100 итераций.

4.1.2 Реализация рейтинговой стратегии

Опираясь на описание рейтинговой стратегии (раздел 2.2.1) и схему реализации алгоритма (рис 4.2), продумана программная реализация (листинг 4.7). Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт с рейтингом.

Набор функций `Rating Search` (листинг 4.7) реализует стратегию поиска по рейтингу, которая приоритизирует применение резолюций к наиболее часто встречающимся атомам (или литералам). Функция `compareAtoms` сравнивает два атома на основе их частоты, а `sortByFrequency` возвращает вектор атомов, отсортированный по частоте встречаемости. `compareClauseRatings` и `sortByRating` выполняют аналогичные операции, но для оценок дизъюнктов.

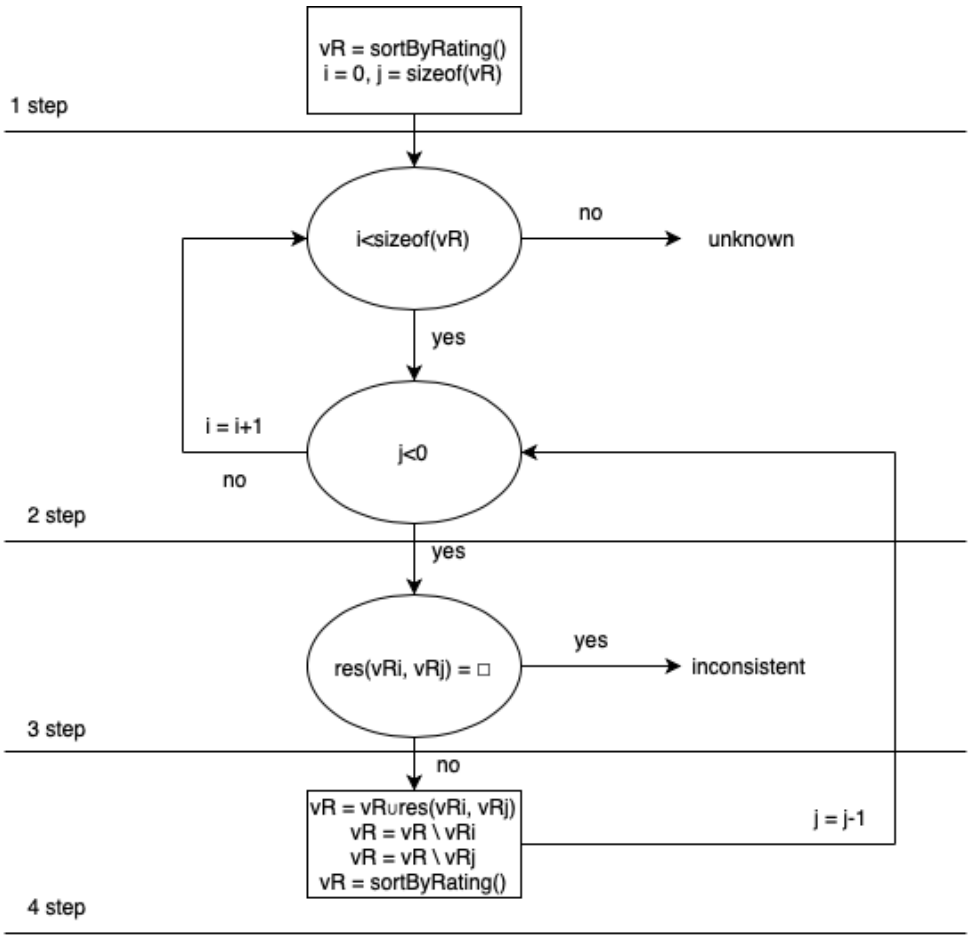


Рис. 4.2 – Схема работы алгоритма рейтинговой стратегии

Листинг 4.7 – Функции алгоритма на основе рейтинга

```
bool compareAtoms(const Atom& a1, const Atom& a2);
vector<Atom> sortByFrequency();
bool compareClauseRatings(const ClauseRating& r1,
                          const ClauseRating& r2);
vector<Clause> sortByRating(vector<Clause> vD,
                            vector<Atom> vC);
string RatingSearch(vector<Clause> vC);
```

Функция `compareAtoms` (листинг 4.8) служит для сравнения двух атомов по полю `otr`. Функция возвращает `true`, если значение поля `otr` первого атома больше, чем у второго, и `false` в противном случае, и используется в сортировке.

Функция `sortByFrequency` (листинг 4.9) подсчитывает частоту встречаемости каждого атома всех предложений и сортирует объекты типа `Atom` по частоте их встречаемости внутри списка `Clause`.

Функции `compareClauseRatings` (листинг 4.10) и `sortByRating` (листинг 4.11) необходимы для работы с рейтингом. Функция `sortByRating` предназначена для сортировки вектора дизъюнктов по рейтингу. Рейтинг каждого дизъюнкта вычисляется исходя из его свойств и свойств его вложенных `Atom`. После того, как рейтинги для всех предложений вычислены, они сортируются в порядке убывания.

Листинг 4.8 – Имплементация функции `compareAtoms`

```
bool compareAtoms(const Atom& a1, const Atom& a2) {
    return a1.otr > a2.otr;
}
```

Листинг 4.9 – Имплементация функции `sortByFrequency`

```
vector<Atom> sortByFrequency(vector<Clause> vD) {
    map<int, int> frequencyMap;
    for (auto& clause : vD) {
        for (auto& atom : clause.a) {
            frequencyMap[atom.otr]++;
        }
    }
    vector<pair<int, int>> frequencyPairs(frequencyMap.begin(),
                                          frequencyMap.end());
    sort(frequencyPairs.begin(), frequencyPairs.end(),
        [](const auto& p1, const auto& p2) {
        return p1.second > p2.second; });
    vector<Atom> vC;
    for (auto& pair : frequencyPairs) {
        for (auto& clause : vD) {
            for (auto& atom : clause.a) {
                if (atom.otr == pair.first) {
                    vC.push_back(atom);
                }
            }
        }
    }
    sort(vC.begin(), vC.end(), compareAtoms);
    return vC;
}
```

Листинг 4.10 – Имплементация функции `compareClauseRatings`

```
bool compareClauseRatings(const ClauseRating& r1, const
ClauseRating& r2) {
    return r1.rating > r2.rating;
}
```

Листинг 4.11 – Имплементация функции `sortByRating`

```
vector<Clause> sortByRating(vector<Clause> vD, vector<Atom> vC)
{
    vector<ClauseRating> clauseRatings(vD.size());
    for (int i = 0; i < vD.size(); ++i) {
        ClauseRating& rating = clauseRatings[i];
        rating.clause = vD[i];
        rating.rating = 0;
        for (auto& atom : rating.clause.a) {
            if (atom.otr == 1) {
                rating.rating += 2;
            }
            for (int j = 0; j < vC.size()
                && j < priorityRating; ++j) {
                if (atom.otr == vC[j].otr) {
                    rating.rating += 1;
                    break;
                }
            }
        }
        if (rating.clause.si == 1) {
            rating.rating += 1;
        }
    }
    sort(clauseRatings.begin(), clauseRatings.end(),
         compareClauseRatings);
    vector<Clause> vR;
    for (auto& rating : clauseRatings) {
        vR.push_back(rating.clause);
    }
    return vR;
}
```

4.1.3 Реализация стратегии, основанной на весах предложений

Опираясь на описание стратегии, основанной на весах предложений (раздел 2.2.4) и схему реализации алгоритма (рис 4.3), продумана программная реализация (листинг 4.12). Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт с весом.

Набор функций `Weight Search` (листинг 4.12) реализует стратегию поиска похожих предложений, где каждому предложению присваивается вес. `getWeightClause` (листинг 4.13) возвращает вес для данного предложения в наборе vD. `getWeightD` (листинг 4.14) возвращает вектор весов для всех предложений в наборе.

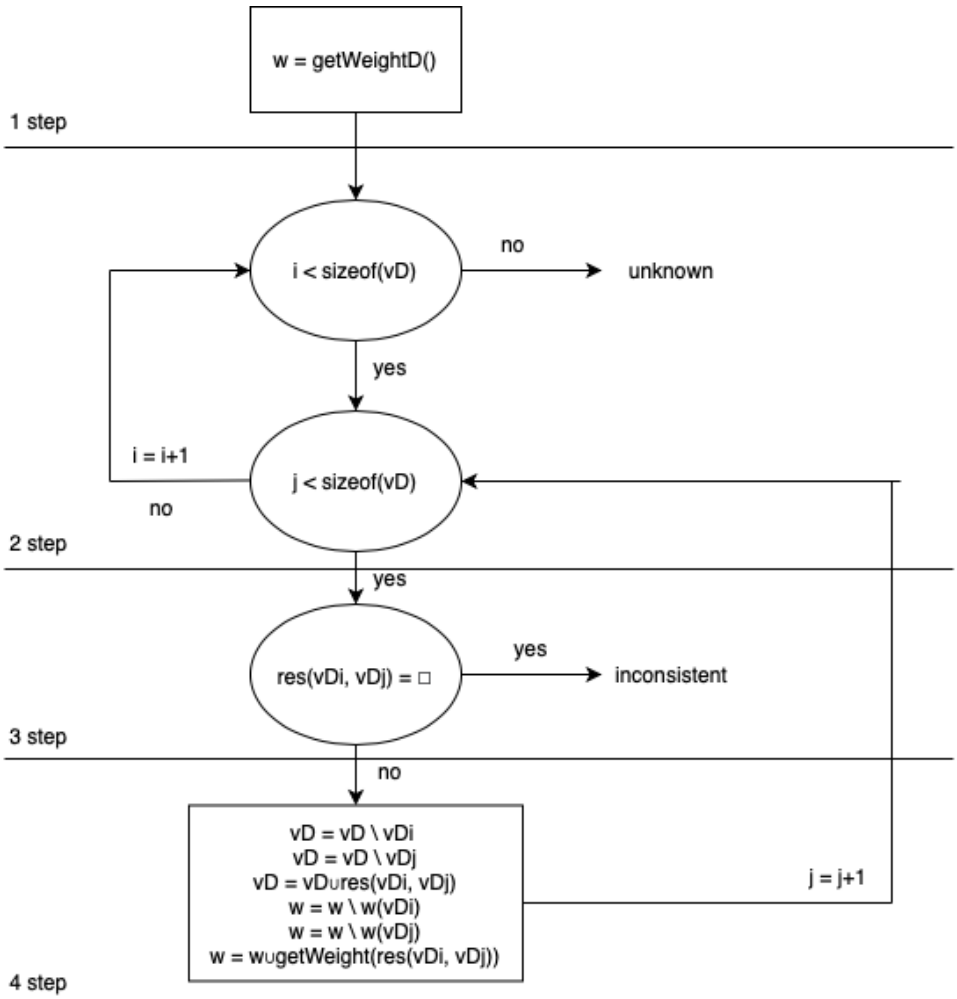


Рис. 4.3 – Схема работы алгоритма стратегии поиска близких по весу дизъюнктов

Листинг 4.12 – Функции алгоритма стратегии поиска близких по весу дизъюнктов

```
int getWeightClause (vector<Clause> &vD, int i);
vector<int> getWeightD (vector<Clause> &m);
string WeightSearch(vector<Clause> vD);
```

Листинг 4.13 – Имплементация функции `getWeightClause`

```
int getWeightClause (vector<Clause> &vD, int i) {
    double s=0.0;
    int n = vD[i].si;
    for (int j=0; j<n; j++) s+=vD[i].a[j].w;
    return s;
}
```

Листинг 4.14 – Имплементация функции `getWeightD`

```
vector<int> getWeightD (vector<Clause> &m) {
    vector<int> w;
    int n = m.size();
    for (int i=0; i<n; i++) {
        w.push_back(getWeightClause(m, i));
    }
    return w;
}
```

4.2 Программная реализация метода резолюций в нечеткой логике

Создание программ, которые базируются на стратегиях нечеткой логики, облегчает решение задач, связанных с формализацией и логическим выводом в различных сферах, таких как автоматическое доказательство теорем, анализ и верификация кода программ, обработка естественного языка и т.д. Благодаря своей способности управлять нечеткостью и неопределенностью, нечеткая логика сохраняет свою значимость и спрос в области искусственного интеллекта и информатики.

В листинге 4.15 представлены структуры для атома и дизъюнкта: каждый атом обладает или не обладает отрицанием, а также имеет наименование, каждый дизъюнкт состоит из нескольких атомов. Соответственно, программа принимает данные в вектор дизъюнктов, который состоит из некоторого количества атомов.

Листинг 4.15 – Структуры атома и дизъюнкта

```
struct Atom {
    int otr;
    string s;
    float w;
};

struct Clause {
    int si;
    vector<Atom> a;
    float w;
};
```

4.2.1 Реализация стратегии, основанной на интерпретации Lee

Опираясь на описание стратегии, основанной на интерпретации Lee, в нечеткой логике (раздел 3.2.1) и схему реализации алгоритма (рис 4.4), продумана программная реализация (листинг 4.16-4.17). Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт.

Листинг 4.16 – Функции алгоритма стратегии в интерпретации Lee

```
string LeeSearch(vector<Clause> vD);
bool mClause(Clause d);
```

Листинг 4.17 – Имплементация функции` mClause`

```
bool mClause(Clause d) {
    int size = d.si;
    double mw = d.a[0].w;
    for (int i=1; i<size; i++)
        if (d.a[i].w > mw)
            mw = d.a[i].w;
    return mw>0.5;
}
```

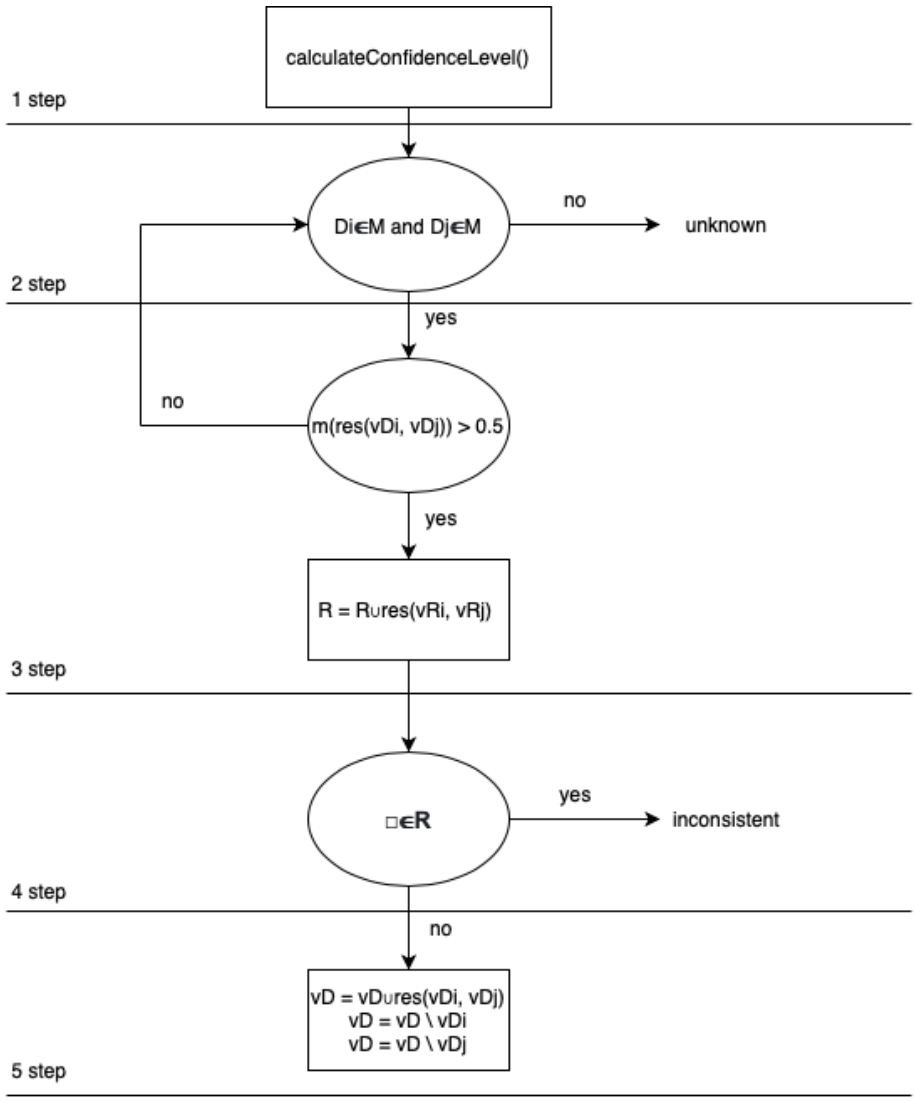


Рис. 4.4 – Схема работы алгоритма стратегии в интерпретации Lee

4.2.2 Реализация стратегии, основанной на интерпретации Mukaidono

Опираясь на описание стратегии, основанной на интерпретации Mukaidono, в нечеткой логике (раздел 3.2.2) и схему реализации алгоритма (рис 4.5), продумана программная реализация (листинг 4.18).

Рассмотрим более подробно резолютивный вывод на основе М-резольвенты. Здесь C – множество дизъюнктов, невыполнимость (противоречивость) которого исследуется. Данный алгоритм направлен в сторону просмотра литер с наиболее высокой степенью доверия. Для компактного изложения алгоритма введем следующие функции:

1) функция *calculateConfidenceLevel(cp)* вычисляет степень доверия каждой литеры (результат в c_p);

2) функция *sortConfidenceLevels(cp)* сортирует последовательность степеней доверия литер в порядке убывания (после применения данной функции массив c_p обновляется);

3) функция *searchAndChoose(Di, Dj, cp)* осуществляет поиск таких дизъюнктов, которые содержат первую по порядку литеру из c_p и могут использоваться для построения значимой резольвенты; если подходящих дизъюнктов нет, то осуществляется поиск дизъюнктов, которые содержат следующую литеру c_p и так далее.

Ниже представлена реализация алгоритма построения резольвенты в интерпретации Mukaidono.

Листинг 4.18 – Алгоритмический язык реализованного алгоритма в интерпретации Mukaidono

```
  Input: C - множество дизъюнктов, невыполнимость
(противоречивость) которого исследуется, R - множество
построенных резольвент
  Input: k = 0, R_0 = ∅
  Output: Результат работы алгоритма – значение из множества {C
is inconsistent, C is unknown}
   // вычисляем степени доверия каждой литеры и сортируем степени
доверия литер по убыванию значения
  calculateConfidenceLevel(c_p); sortConfidenceLevels(c_p);
```

```
  // продолжаем алгоритм пока в C есть дизъюнкты, которые образуют резольвенту
  while (∃ D_i ∈ M and ∃ D_j ∈ M) do
     // если существует значимая резольвента, то добавляем ее в R
     if (∃ res(D_i, D_j) is significant) do
        searchAndChoose (D_i, D_j, c_p); R_{k+1} = R_k ∪ {res(D_i, D_j)}
        // если в R есть □, то C - противоречиво, конец программы
        if (□∈ R_{k+1}) C is inconsistent; end program;
          // если в R нет □, то меняем множество C и идем дальше
          else C = C \ D_i ∪ C \ D_j ∪ res(D_i, D_j); k=k+1;
    end if
end while
// если в C закончились дизъюнкты, которые образуют резольвенту, считаем, что противоречивость C неизвестена, конец программы
C is unknown;
end program
```

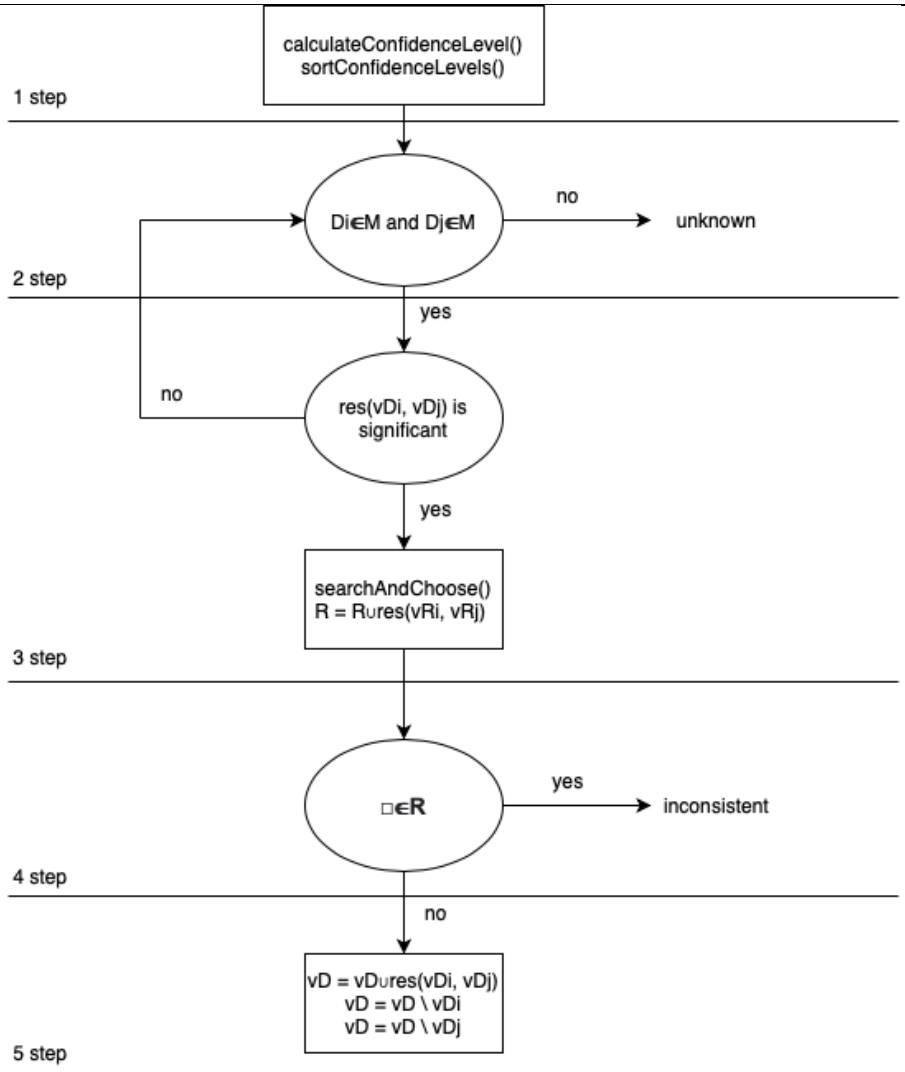


Рис. 4.5 – Схема работы алгоритма стратегии в интерпретации Mukaidono

4.2.3 Реализация стратегии, основанной на сходстве

Опираясь на описание стратегии, основанной на сходстве, в нечеткой логике (раздел 3.2.3) и схему реализации алгоритма (рис 4.6), продумана программная реализация (листинг 4.19). Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт.

Функция `simAtom` (листинг 4.20) считает индекс сходства по формуле из главы 3.2.2.3:

$$Sim_1(A,B) = 1 - \frac{\max\left\{\min_x\left\{1-\mu_A(x),\mu_B(x)\right\},\min_x\left\{\mu_A(x),1-\mu_B(x)\right\}\right\}}{\max_x\left\{\mu_A(x),\mu_B(x)\right\}}$$

Функция `resEmpty` (листинг 4.21) определяет, образуют ли дизъюнкты d и ad пустой дизъюнкт.

Листинг 4.19 – Функции алгоритма на основе сходства

```
string SimilaritySearch(vector<Clause> vD);
bool simAtom(Atom a1, Atom a2);
bool resEmpty(Clause d, Clause ad);
```

Листинг 4.20 – Имплементация функции `simClause`

```
bool simAtom(Atom a1, Atom a2) {
    double sim = 1 - (max(min(1-a1.w,a2.w),
                      min(a1.w,1-a2.w)))/max(a1.w,a2.w);
    return sim>0.5;
}
```

Листинг 4.21 – Имплементация функции `resEmpty` (EPS = 0.01)

```
bool resEmpty(Atom a1, Atom a2) {
    return abs(a1.w-a2.w)<EPS?1:0;
}
```

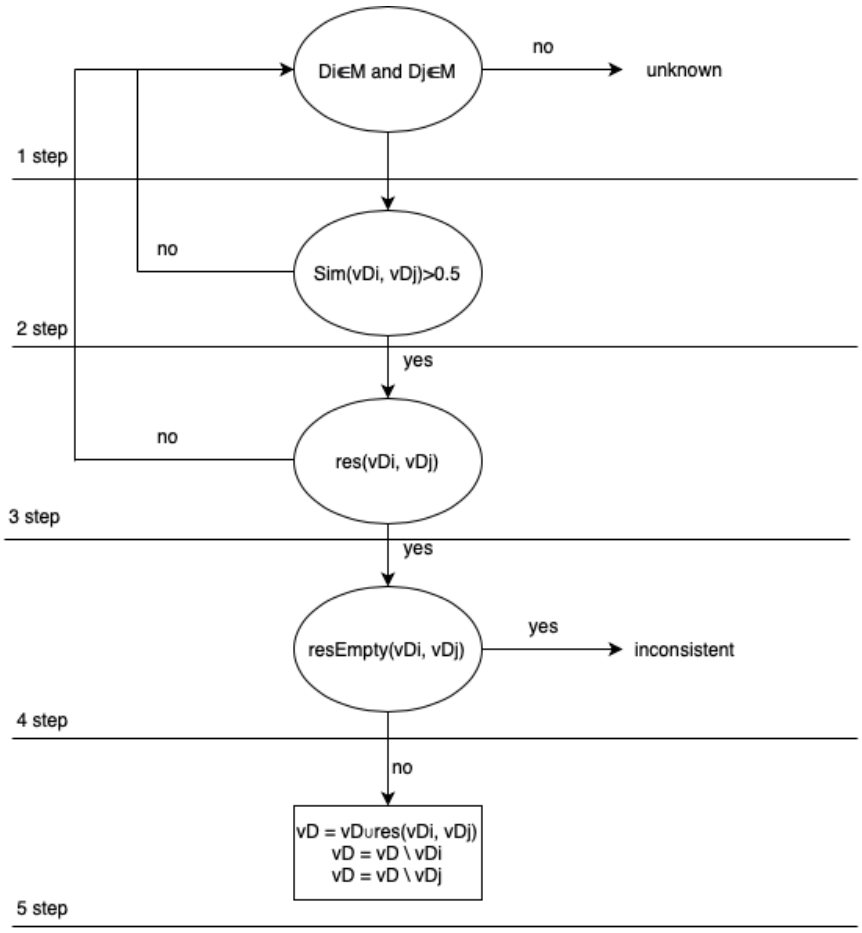


Рис. 4.6 – Схема работы алгоритма стратегии на основе сходства

4.3 Вычислительный эксперимент, анализ результатов и разработка рекомендаций

В данной главе представлены практические исследования, демонстрирующие применение разработанных алгоритмов в различных условиях и сценариях. В рамках диссертации было проведено около 50 экспериментов, каждый из которых направлен на изучение эффективности, стабильности и применимости алгоритмов в разнообразных вычислительных задачах. Далее подробно описаны наиболее значимые и показательные эксперименты, которые позволяют оценить реальные возможности и ограничения предложенных алгоритмических решений.

Выборка экспериментов для включения в эту главу основывалась на их репрезентативности и способности демонстрировать ключевые характеристики и преимущества разработанных алгоритмов. Описания

экспериментов содержат детализированные данные о условиях проведения, входных параметрах, используемых методиках и полученных результатах. Анализ этих данных позволяет глубже понять принципы работы алгоритмов и их практическую ценность.

Цели эксперимента, следующие:

1. Изучить эффективность разработанных алгоритмов.

Оценить производительность алгоритмов в решении заданных вычислительных задач, сравнить полученные результаты с существующими алгоритмическими подходами: стратегией полного перебора.

2. Проанализировать стабильность алгоритмов.

Определить устойчивость алгоритмов к разным входным данным, выявить любые ограничения или условия, при которых эффективность алгоритмов снижается.

3. Оценить применимость алгоритмов в различных сценариях.

Оценить универсальность алгоритмов путем их применения в разнообразных вычислительных задачах, исследовать способность алгоритмов адаптироваться к специфическим требованиям различных задач.

4. Подтвердить теоретические положения.

Доказать практическую эффективность алгоритмов в соответствии с теоретическими предположениями, изложенными в предыдущих разделах диссертации.

4.3.1 Эксперименты со стратегиями классической логики

Эксперимент со стратегией поиска минимального дизъюнкта

Рассмотрим следующее множество предложений:

$$S=\left\{C(x)\vee O(x),B(y)\vee \bar{C}(y),W(A),\bar{B}(A),\bar{O}(z)\vee \bar{W}(z),C(v)\vee B(v)\vee \bar{W}(v)\right\}$$

при этом данные для ввода будут представлены в следующем виде:

6

2 0 C 0 O

2 0 B 1 C

1 0 W

1 1 B

2 1 O 1 W

3 0 C 0 B 1 W

Анализ примера работы алгоритма, основанного на поиске минимального дизъюнкта, показывает важные аспекты эффективности и направленности предложенного подхода.

На рисунке 4.7 (слева) демонстрируется процесс, при котором алгоритм успешно определяет противоречивость множества S, достигая цели - нахождения пустого дизъюнкта - через построение всего лишь четырех резольвент. Отсутствие неиспользованных резольвент указывает на высокую целенаправленность и экономичность алгоритма, где каждый шаг вносит вклад в приближение к решению, минимизируя лишние вычисления и потребление ресурсов.

На рис. 4.7 (справа) отображено время работы алгоритма поиска минимального дизъюнкта и полного перебора. Видно, что новая стратегия работает эффективнее по времени в 2 раза.

Ключ к эффективности алгоритма на основе минимального дизъюнкта лежит не только в его способности быстро достигать решения, но и в том, как он оптимизирует процесс поиска, исключая неэффективные пути и фокусируясь на наиболее перспективных направлениях для достижения цели.

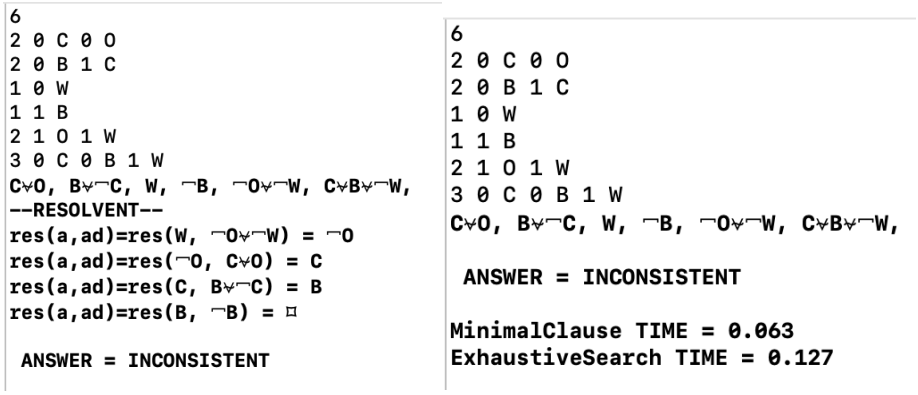


Рис. 4.7 – Пример работы алгоритма на Примере 1: на основе минимального дизъюнкта.

Теперь рассмотрим другое множество предложений

$$S = \begin{Bmatrix} C(x) \vee O(x) \vee \bar{B}(x), B(y) \vee C(y), W(A) \vee \bar{B}(A), \bar{O}(z) \vee \bar{W}(z), \\ \bar{C}(v) \vee B(v) \vee \bar{W}(v) \end{Bmatrix}.$$

По сравнению с предыдущим примером множество S не содержит одночленов. Данное множество выполнимо, и □ получить нельзя.

На рис. 4.8 представлен пример работы алгоритма, основанного на поиске минимального дизъюнкта. Видим, что множество S – выполнимо, а значит можно сделать следующий вывод: успешное определение выполнимости множества S без зацикливания алгоритма указывает на его способность эффективно обрабатывать сложные логические структуры и принимать верные решения даже в условиях, когда найти конкретное решение не представляется возможным. Это подчеркивает одно из ключевых преимуществ данного подхода: способность корректно завершать поиск при сохранении высокой точности в оценке выполнимости логических множеств.

```
5
3 0 C 0 O 1 B
2 0 B 0 C
2 0 W 1 B
2 1 O 1 W
3 1 C 0 B 1 W

C∨O∨¬B, B∨C, W∨¬B, ¬O∨¬W, ¬C∨B∨¬W,
--RESOLVENT--
res(a,ad)=res(B∨C, C∨O∨¬B) = C∨O∨C
NOres(a,ad)=res(B∨C, W∨¬B)
NOres(a,ad)=res(B∨C, ¬O∨¬W)
res(a,ad)=res(B∨C, ¬C∨B∨¬W) = B∨¬W∨B
NOres(a,ad)=res(B∨C, B∨C)
NOres(a,ad)=res(B∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, W∨¬B)
res(a,ad)=res(C∨O∨C, ¬O∨¬W) = ¬W∨C∨C
NOres(a,ad)=res(C∨O∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, C∨O∨C)
res(a,ad)=res(B∨¬W∨B, W∨¬B) = ¬B∨B∨B∨W∨¬W∨B∨W∨B∨¬W
NOres(a,ad)=res(B∨¬W∨B, B∨C)
NOres(a,ad)=res(B∨¬W∨B, B∨C)
NOres(a,ad)=res(B∨¬W∨B, C∨O∨C)
NOres(a,ad)=res(B∨¬W∨B, B∨¬W∨B)

 ANSWER = MAYBE SATISFIABLE
```

Рис. 4.8 – Пример работы алгоритма на Примере 2: на основе минимального дизъюнкта.

Такой исход эксперимента также демонстрирует, что алгоритм обладает значительной преимущественной стабильностью и надёжностью, что является критически важным аспектом для алгоритмов, применяемых в вычислительных системах с высокими требованиями к точности и безопасности.

Далее рассмотрим следующее множество предложений:

$$S = \left\{ C(x) \vee O(x), B(y) \vee \overline{C}(y), W(A), \overline{B}(A), \overline{O}(z) \right\}.$$

В данном примере по сравнению с предыдущими уже на начальных итерациях потребуется изменение множества D. На рис 4.9 представлен пример работы алгоритма, основанного на поиске минимального дизъюнкта. Видим, что множество S – противоречиво. В процессе работы алгоритма было построено 3 резольвенты и алгоритм снова хорошо себя показывает, как и в предыдущих примерах.

```
5
2 0 C 0 O
2 0 B 1 C
1 0 W
1 1 B
1 1 O

C∨O, B∨¬C, W, ¬B, ¬O,
--RESOLVENT--
NOres(a,ad)=res(W, C∨O)
NOres(a,ad)=res(W, B∨¬C)
NOres(a,ad)=res(W, ¬B)
NOres(a,ad)=res(W, ¬O)
--RESOLVENT--
NOres(a,ad)=res(¬B, C∨O)
res(a,ad)=res(¬B, B∨¬C) = ¬C
NOres(a,ad)=res(¬B, ¬O)
NOres(a,ad)=res(¬B, ¬B)
res(a,ad)=res(¬C, C∨O) = O
NOres(a,ad)=res(¬C, ¬O)
NOres(a,ad)=res(¬C, ¬B)
NOres(a,ad)=res(¬C, ¬C)
res(a,ad)=res(O, ¬O) = □

 ANSWER = INCONSISTENT
```

Рис. 4.9 – Пример работы алгоритма на Примере 3: на основе минимального дизъюнкта.

Таким образом, практически показана работа стратегии, алгоритм, которой основан на поиске и работе с минимальным дизъюнктом, и получены корректные результаты.

Эксперимент с рейтинговой стратегией

Пусть дано следующее множество предложений:

$$S=\left\{C(x)\vee O(x),Z(y)\vee \overline{C}(y),Y(A),\overline{Z}(A),\overline{O}(z)\vee \overline{Y}(z)\right\},$$

при этом данные для ввода:

5

2 0 C 0 O

2 0 Z 1 C

1 0 Y

1 1 Z

1 1 O 1 Y

На рис. 4.10 представлен полный процесс работы алгоритма, который в итоге нашел решение, дойдя до пустого дизъюнкта, построив 5 резольвент. Несмотря на то, что алгоритм и не предложил самого эффективного пути к решению задачи в плане минимизации количества шагов или выбора оптимальных резольвент, он достиг цели. Важно также отметить, что отсутствие лишних резольвент в процессе поиска говорит о том, что алгоритм способен избегать ненужных вычислений и направляет свои ресурсы на продвижение к цели.

```
CVO, ZV¬C, Y, ¬Z, ¬OV¬Y,


res(a,ad)=res(¬OV¬Y, Y) = ¬O
res(a,ad)=res(ZV¬C, CVO) = OVZ
–
ZV¬C, ¬Z, ¬O, CVO, OVZ,


res(a,ad)=res(¬Z, OVZ) = O
res(a,ad)=res(¬O, OVZ) = Z
–
ZV¬C, ¬Z, CVO, O, Z,


–
ZV¬C, ¬Z, CVO, O, Z,


–
ZV¬C, ¬Z, CVO, O, Z,


res(a,ad)=res(Z, ¬Z) = □

 ANSWER = INCONSISTENT
```

Рис. 4.10 – Процесс решения задачи алгоритмом на основе рейтинга дизъюнктов

Эксперимент со стратегией, основанной на весах предложений

Пусть дано следующее множество предложений:

$$S=\left\{C(x)\vee O(x),Z(y)\vee \bar{C}(y),Y(A),\bar{Z}(A),\bar{O}(z)\vee \bar{Y}(z)\right\},$$

На рис. 4.11 представлен полный результат работы алгоритма, который в итоге нашел решение. Особенно значимым является обнаружение решения через построение всего лишь двух резольвент, что указывает на высокую эффективность алгоритма в выборе пути поиска решения. Это демонстрирует не только способность алгоритма эффективно управлять процессом резолюции, но и его потенциал для минимизации вычислительных затрат за счет выбора оптимальных путей поиска решений.

```
5
2 0 C 0.7 0 O 0.3
2 0 Z 0.4 1 C 0.3
1 0 Y 0.6
1 1 Z 0.6
2 1 O 0.6 1 Y 0.4

CVO, ZV¬C, Y, ¬Z, ¬OV¬Y,


res(a,ad)=res(CVO, ZV¬C) = ¬C
res(a,ad)=res(¬Z, ZV¬C) = □

 ANSWER = INCONSISTENT
```

Рис. 4.11 – Процесс решения задачи алгоритмов на основе весов дизъюнктов

Сравнение времени работы новых алгоритмов классической логики

На рис. 4.12 представлено сравнение среднего времени работы новых алгоритмов классической логики на нескольких десятках примеров. Таким образом, фактически алгоритмы, основанные на работе с весами предложений и поиске минимального дизъюнкта, является самыми быстрыми для большинства примеров задач.

1. Из таблицы видно, что все четыре алгоритма поиска привели к решению, пустой дизъюнкт был найден

2. Время поиска различается между алгоритмами. Самым быстрым оказался алгоритм на основе минимального дизъюнкта (0.063 с), перед ним следует алгоритм на основе весов дизъюнктов (0.022 с). Это указывает на то,

что эти алгоритмы могут быть более эффективными в отношении времени исполнения, по крайней мере в этом конкретном случае.

3. Алгоритм полного перебора показал время поиска 0.114 с, а Рейтинговый алгоритм - 0.354 с, что делает их медленнее в сравнении с остальными двумя. В некоторых случаях, где время выполнения критично, эти алгоритмы могут оказаться менее предпочтительными.

Search Algorithm	Search Time (s)	Answer
Exhaustive	0.114	INCONSISTENT
Rating	0.354	INCONSISTENT
Minimal Clause	0.063	INCONSISTENT
Weight	0.022	INCONSISTENT

Рис. 4.12 – Сравнение времени работы новых алгоритмов классической логики

4.3.2 Эксперименты со стратегиями нечеткой логики

Пусть даны следующие предложения:

$$\{C(x) \vee O(x), Z(y) \vee \neg C(y), Y(A), \neg Z(A), \neg O(z) \vee \neg Y(z)\}$$

Найдем решение и применим все предложенные стратегии нечеткой логики, тогда в рис. 4.13 получим значения времени работы алгоритмов.

Search Algorithm	Search Time (s)	Answer
Lee Search	0.063	INCONSISTENT
Mukaidano Search	0.058	INCONSISTENT
Similarity Search	0.074	INCONSISTENT

Рис. 4.13 – Время и результат работы алгоритмов нечеткой логики

Пусть теперь даны следующие предложения:

$$\{P_1 \vee P_2, P_3 \vee P_4, P_5 \vee P_6, P_7 \vee P_8, P_9 \vee P_{10}\}$$

при этом

$$P_1 \sim \neg P, P_2 \sim Q, P_3 \sim \neg Q, P_4 \sim R, P_5 \sim \neg R, P_6 \sim S, P_7 \sim \neg S, P_8 \sim T, P_9 \sim \neg T, P_{10} \sim P,$$

где символ $\sim$ означает “близко”.

И пусть даны значения истинности каждого атома:

$$P_1 = 0.1, P_2 = 0.2, P_3 = 0.9, P_4 = 0.3, P_5 = 0.8, P_6 = 0.3, P_7 = 0.8, P_8 = 0.2, P_9 = 0.9, P_{10} = 0.9$$

Найдем решение (рис. 4.14) и видим, что стратегии на основе интерпретации Lee и Mukaidono не смогли найти решение, тогда как стратегия на основе сходства завершила успешный поиск решения уже через 0.059 сек.

```
| Search Algorithm  | Search Time (s)| Answer
+-------------------+----------------+---------------------+
| Lee Search        | 0.102          | I CAN'T RESOLVE IT |
| Mukaidano Search  | 0.114          | I CAN'T RESOLVE IT |
| Similarity Search | 0.059          | INCONSISTENT |
+-------------------+----------------+---------------------+
```

Рис. 4.14 – Время и результат работы алгоритмов нечеткой логики

Рассмотрим следующий пример:

$$\{\overline{P} \vee Q, \overline{Q} \vee R, \overline{R} \vee S, \overline{S} \vee T, \overline{T} \vee P\}$$

И пусть даны значения истинности каждого атома:

$$P = 0.9, Q = 0.2, R = 0.3, S = 0.3, T = 0.2$$

Можно заметить, что стратегия на основе интерпретации Lee не смошла найти решение, тогда как другие две стратегии справились с решением успешно (рис. 4.15).

```
| Search Algorithm  | Search Time (s)| Answer
+-------------------+----------------+---------------------+
| Lee Search        | 0.103          | I CAN'T RESOLVE IT |
| Mukaidano Search  | 0.074          | INCONSISTENT |
| Similarity Search | 0.059          | INCONSISTENT |
+-------------------+----------------+---------------------+
```

Рис. 4.15 – Время и результат работы алгоритмов нечеткой логики

Из заданных примеров, можно сделать следующие выводы:

1. Все алгоритмы поиска приводят к результату за примерно одно время, если результат решения находится достаточно быстро.

2. Время поиска различается между алгоритмами. Самым быстрым оказался алгоритм на сходства (0.059 с). После следуют алгоритмы на основе интерпретации Lee и Mukaidono (0.074 - 0.114 с), это указывает на то, что эти алгоритмы плохо работают на задачах, где заданы близкие атомы, а не четко различные атомы, как привычно в классической логике.

3. Некоторые задачи не решаются с помощью алгоритмов стратегии на основе интерпретации Lee или Mukaidono, и это говорит о том, что их можно улучшать.

Таким образом, анализ представленных результатов экспериментов позволяет выделить несколько важных моментов: во-первых, алгоритмы, основанные на классической логике, показали высокую скорость обработки и применимость в различных ситуациях, подтверждив свою эффективность. Во-вторых, предложенные решения демонстрируют универсальность и способность адаптироваться к разным условиям, что открывает широкие перспективы для их использования. Кроме того, подтверждена устойчивость алгоритмов к различным входным данным, что критически важно для решения задач в условиях неопределенности. Несмотря на это, были зафиксированы случаи, когда некоторые стратегии нечеткой логики не нашли решение, а значит открываются возможности в дальнейшей оптимизации и усовершенствовании предложенных подходов.

В целом, результаты и выводы исследования позволяют достичь поставленных перед экспериментом целей и подчеркивают значимость разработанных алгоритмов для научного сообщества и их практическую ценность, а также намечают направления для будущих улучшений в области.

4.4 Разработка мобильного приложения для получения эффективных рекомендаций по улучшению качества тестирования

Известно, что реализация каждого этапа жизненного цикла IT-проекта является необходимым условием для появления качественного программного продукта. Поэтому имеет смысл говорить о важности тестирования в общем процессе разработки ПО, при этом можно выделить следующие тенденции: растет понимание необходимости промышленных методов тестирования, в частности с применением специальных средств автоматизации; идет поиск возможностей для оптимизации затрат на выполнение данных работ с точки зрения общей организации процесса; начинает широко использоваться

исследовательское тестирование, которое обусловливает важность опыта работы тестировщика. Разработанная программа была положена в основу мобильного приложения, которое предназначено для поддержки принятия решений при оценке качества тестирования ПО в рамках разработки различных IT-проектов. Основное функциональное назначение приложения заключается в формировании рекомендаций по улучшению качества тестирования на основе оценки ситуации, которая имеет место на текущий момент времени.

В приложении имеется ряд параметров, которые характеризуют ситуацию, сложившуюся на конкретный момент времени и влияющую на анализ и оценку качества тестирования: *имеется общий план проекта*, *пересмотрена оценка задач спринта*, *составлен план тестирования/таблица статуса тестирования на спринт*, *определены зоны ответственности в команде тестирования*, *пересмотрен процесс тестирования*, *сделано ревью требований для задач спринта*, *сделано ревью дизайна*, *настроена тестовая среда/написаны моки*, *завершено создание тест-кейсов/апи-тестов*, *функциональное тестирование закончено*, *повторное тестирование закончено*, *итоговое тестирование завершено*, *проверена миграция*; *приемочное тестирование завершено*; *составлен отчет о прогрессе тестирования/итоговый отчет о тестировании*. Заметим, что некоторые параметры считаются обязательными для обеспечения качества ведения проекта. Особенностью приложения является возможность интерактивной оценки общего критерия качества тестирования.

Метод резолюции используется для разрешения возможных противоречий между установленными правилами проведения тестирования и работами-операциями, которые осуществляются на текущий момент времени.

Изложение реализации мобильного приложения способствует углублению знаний в области вычислительной техники и программной инженерии, а также стимулирует дальнейшие научные разработки. Включение разнообразных сценариев и условий экспериментов обеспечивает

комплексное понимание применения алгоритмов, подчеркивая их универсальность и адаптивность к различным вычислительным задачам и окружающей среде. Таким образом, данная глава не только подтверждает теоретическую значимость исследования, но и является ключевым элементом, демонстрирующим его практическую применимость и эффективность.

4.4.1 Инструменты и технологии для разработки приложения

При разработке мобильного приложения для данной научной диссертации были использованы разнообразные современные инструменты и технологии. Выбор этих инструментов был обусловлен необходимостью создания высококачественного, функционального и легко масштабируемого программного продукта, который бы демонстрировал применение новых алгоритмов в реальных условиях [32,33,35,61,62].

Одним из основных инструментов разработки, который был использован в процессе создания приложения, является Xcode. Xcode – это интегрированная среда разработки (IDE), которую Apple предоставляет для создания приложений для своих платформ, включая iOS, MacOS, watchOS и tvOS. Xcode обладает рядом уникальных свойств и инструментов, которые значительно облегчают процесс разработки. Он включает в себя современный редактор кода, инструменты для проектирования пользовательского интерфейса, а также мощные средства для тестирования, отладки и оптимизации приложений. Одним из ключевых преимуществ Xcode является его тесная интеграция с iOS SDK, что позволяет разработчикам полноценно использовать все возможности операционной системы и библиотек.

Для разработки пользовательского интерфейса в проекте использовался SwiftUI. SwiftUI – это относительно новый декларативный фреймворк от Apple для построения пользовательских интерфейсов на всех их платформах. В SwiftUI, вместо того чтобы писать детальные инструкции о том, как создать интерфейс, разработчики указывают, что должно быть на экране, а сам фреймворк обрабатывает большую часть реализации. Это облегчает

разработку и обновление пользовательского интерфейса, а также делает его более единообразным и стабильным.

В разработке использовались новые стратегии метода резолюции. Применение метода резолюции позволило применять стратегический подход к определению и исправлению возникающих проблем в коде или структуре приложения. Этот метод охватывает все - от выявления и устранения багов до оптимизации алгоритмов и улучшения производительности. Такой подход упрощает процесс выявления и устранения ошибок, увеличивая общую надежность и производительность приложения.

Мобильное приложение было разработано с целью демонстрации внедрения новых алгоритмов и их применения в реальных условиях. Целью было показать, как теоретические исследования, проведенные в ходе работы над диссертацией, могут быть переведены в практику и как они могут быть использованы для решения реальных проблем. Процесс разработки приложения включал не только создание кода, но и тестирование его в различных сценариях, чтобы убедиться в его эффективности и применимости в реальной жизни.

4.4.2 Интеграция C++ библиотеки в SwiftUI

В современной разработке программного обеспечения часто возникает необходимость в интеграции кода, написанного на различных языках программирования, для повышения производительности, повторного использования существующих алгоритмов или доступа к специфическим для платформы API. Данная глава представляет собой пошаговое руководство по интеграции библиотеки C++ в мобильное приложение, разработанное с использованием SwiftUI для платформы iOS. Методика основывается на использовании механизма "моста", сочетающего возможности Objective-C++ и Swift для создания гибридных приложений.

Инструкция реализации интеграции приведена ниже.

1. Подготовка библиотеки C++

Первым шагом является подготовка библиотеки C++, включающая создание заголовочных файлов (.h) для объявления интерфейсов, которые будут доступны из Swift кода (листинг 4.22).

Листинг 4.22 – Заголовочный файл SearchAlgorithms.hpp, который объявляет функции алгоритмов классической логики

```
// SearchAlgorithms.hpp
#include <string>
#include <vector>

class SearchAlgorithms {
public:
    static string exhaustiveSearch(vector<Clause> vD, int n);
    static vector<Atom> sortByFrequency();
    static vector<Clause> sortByRating(vector<Clause> vD,
                                       vector<Atom> vC);
    static string RatingSearch(vector<Clause> vC);
    static Clause findMinD(vector<Clause> &m, int n);
    static string MinimalClause(vector<Clause> vD, int n);
    static int MinimalClauseWithDescription(vector<Clause> vD,
                                            int n);
    static int getWeightClause (vector<Clause> &vD, int i);
    static vector<int> getWeightD (vector<Clause> &m);
    static string WeightSearch(vector<Clause> vD);
};
```

2. Добавление библиотеки в проект Xcode

Для интеграции библиотеки C++ файлы добавляются в проект Xcode и включаются в фазу сборки "Compile Sources".

3. Создание Objective-C++ мостового файла

Создается файл Objective-C++ класса с расширением .mm, который выступает в роли моста, обеспечивая взаимодействие между Swift и C++. В этом файле реализуется обертка над функциями или классами C++, делая их доступными для вызова из кода на Swift (листинг 4.23).

Листинг 4.23 – Реализация Objective-C обертки

```
// SearchAlgorithmsWrapper.mm
#import "SearchAlgorithms.hpp"
#import <Foundation/Foundation.h>

@interface SearchAlgorithmsWrapper: NSObject
+     (NSString     *)exhaustiveSearchWithVD:(vector<Clause>)vD
n:(int)n;
```

```
// Аналогично для остальных функций
@end

@implementation SearchAlgorithmsWrapper
+ (NSString *)exhaustiveSearchWithVD:(vector<Clause>)vD n:(int)n
{
    std::string result = SearchAlgorithms::exhaustiveSearch(vD,
n);
    return [NSString stringWithUTF8String:result.c_str()];
}
// Аналогично для остальных функций
@end
```

4. Настройка мостового заголовка для Swift

Для доступа к объявленным в Objective-C++ классам или функциям из Swift необходим мостовой заголовок, который автоматически предлагается создать Xcode при добавлении первого файла Objective-C в проект Swift. В этот заголовок импортируются все необходимые объявления.

5. Использование C++ кода в Swift

После настройки мостового заголовка возможно обращение к функциям и классам C++ напрямую из Swift кода, что позволяет эффективно интегрировать логику, написанную на C++, в SwiftUI приложения (листинг 4.24).

Листинг 4.24 – Использование функций

```
#import "SearchAlgorithmsWrapper.h"

let result = SearchAlgorithmsWrapper.exhaustiveSearch(withVD:
vD, n: n)
```

4.4.3 Архитектура и процесс работы приложения

В данном разделе рассматривается архитектура и процесс работы приложения, которое воплощает в себе передовые подходы и методологии в области управления проектами и качества программного обеспечения. В рамках разработки приложения были заложены ключевые критерии, обеспечивающие высокий уровень организации процесса разработки и тестирования продукта.

В основу приложения заложены следующие принципиально важные критерии (листинг 4.25):

1. Наличие общего плана проекта гарантирует целостное видение и стратегическое направление разработки, обеспечивая эффективное руководство и координацию между участниками проекта.

2. Регулярный пересмотр оценок задач спринта позволяет адаптироваться к изменяющимся условиям и требованиям, поддерживая актуальность планов и предотвращая возможные задержки в выполнении проекта.

3. Составление плана тестирования и таблицы статуса тестирования на каждый спринт обеспечивает систематический подход к верификации качества продукта и позволяет отслеживать прогресс в работе над ошибками.

4. Описание зон ответственности в команде тестирования уточняет роли и обязанности участников, способствуя повышению эффективности взаимодействия внутри команды.

5. Пересмотр процесса тестирования является ключом к непрерывному улучшению качества и эффективности проверки продукта.

6. Ревью требований и дизайна для задач спринта гарантирует, что все изменения и нововведения учитываются на ранних этапах разработки, минимизируя риск возникновения ошибок в будущем.

7. Настройка тестовой среды и подготовка моков обеспечивает эффективное и реалистичное тестирование функционала в изолированной среде.

8. Завершение создания тест-кейсов и API-тестов служит фундаментом для комплексного тестирования приложения и верификации его соответствия требованиям и спецификациям.

9. Проведение функционального и повторного тестирования, а также итогового тестирования и проверка миграции данных, подтверждает готовность продукта к релизу и его соответствие высоким стандартам качества.

10. Приемочное тестирование и составление итогового отчета о тестировании предоставляют полную картину о состоянии продукта перед его запуском в эксплуатацию.

11. Оценка качества выполнения работ на основе нечеткого параметра, принимающего значения от 0 до 1, позволяет гибко оценить соответствие продукта установленным критериям качества.

Первые 15 критериев могут принять значение либо TRUE, либо FALSE, а последний параметр является нечетким и может принимать значение от 0 до 1.

Листинг 4.25 – Базовые критерии

```
   baseCriterias = [
        "Присутствует общий план проекта",
        "Пересмотрена оценка задач спринта",
        "Составлен план тестирования/таблица статуса тестирования
на спринт",
        "Зоны ответственности в команде тестирования",
        "Пересмотрен процесс тестирования",
        "Сделано ревью требований для задач спринта",
        "Сделано ревью дизайна",
        "Настроена тестовая среда/написаны моки",
        "Завершено создание тест-кейсов/апи-тестов",
        "Функциональное тестирование закончено",
        "Повторное тестирование закончено",
        "Итоговое тестирование завершено",
        "Проверена миграция",
        "Приемочное тестирование завершено",
        "Составлен отчет о прогрессе тестирования/итоговый отчет
о тестировании"
    ]
```

В рамках архитектуры и процесса работы приложения предусмотрена сложная система правил и рекомендаций, направленная на обеспечение высокого уровня качества разработки и тестирования. Например, если не выполняются параметры "Итоговое тестирование завершено" или "Проверена миграция", то необходимо "Пересмотреть процесс итогового тестирования".

Система правил реализует комплексный подход к управлению качеством, включая следующие основные направления:

1. В случае одновременного несоответствия нескольких критериев в области тестирования предусматривается проведение комплексного аудита

процесса тестирования. Цель аудита - выявление и устранение системных недостатков, влияющих на качество и эффективность тестирования.

2. Невыполнение критериев, связанных с планированием и оценкой задач, инициирует ретроспективу спринта с детальным анализом ошибок и недочетов в процессах планирования.

3. При необходимости пересмотра процесса тестирования на различных этапах разработки предлагается разработка и внедрение новой стратегии тестирования, основанной на передовых практиках и стандартах отрасли.

4. Одновременное несоответствие критериев планирования и отчетности требует улучшения системы управления проектом через внедрение более продвинутых инструментов и методик планирования и отчетности.

5. Выявление проблем с качеством на различных этапах реализации проекта акцентирует внимание на необходимости комплексной оптимизации процессов качества.

6. При обнаружении проблем, связанных с тестированием и приемочными испытаниями, актуализируется важность пересмотра подходов к тестированию и укрепления сотрудничества с заказчиком для точного уточнения требований и ожиданий.

Соответственно, рекомендации по улучшению процессов включают в себя:

1) пересмотреть задачи спринта;

2) пересмотреть процесс итогового тестирования;

3) пересмотреть планирование задач;

4) пересмотреть зоны ответственности;

5) пересмотреть процесс проектирования;

6) провести анализ и оптимизацию текущего процесса тестирования;

7) провести все запланированные функциональные тесты;

и тд.

Таким образом, внедрение данных правил и рекомендаций обеспечивает глубокий анализ и комплексный подход к управлению качеством на всех

этапах разработки и тестирования продукта, что способствует повышению его надежности и соответствия высоким стандартам.

Архитектура мобильного приложения и процесс работы включает несколько этапов (рис. 4.16). На первом этапе пользователь вводит данные о проекте на стартовом экране (рис. 4.17а-б). Эти данные (D1, D2, … Dn) проходят процесс обработки и затем совместно с множеством правил Rules и множеством рекомендаций Recommendations формируются в необходимый формат взодных данных для передачи их в библиотеку, которая реализует алгоритмы метода резолюций. Данное приложение использует алгоритм поиска решений методом резолюций с использованием стратегий минимального дизъюнкта и М-резольвенты. Выбор данных стратегий обусловлен их быстродействием. В процессе работы алгоритма осуществляется проверка характеристик каждой предлагаемой рекомендации на соответствие совокупности фактов и правил. После завершения работы основной функции, в структуру данных recommendations включаются рекомендации, которые успешно прошли проверку (рис. 4.17в).

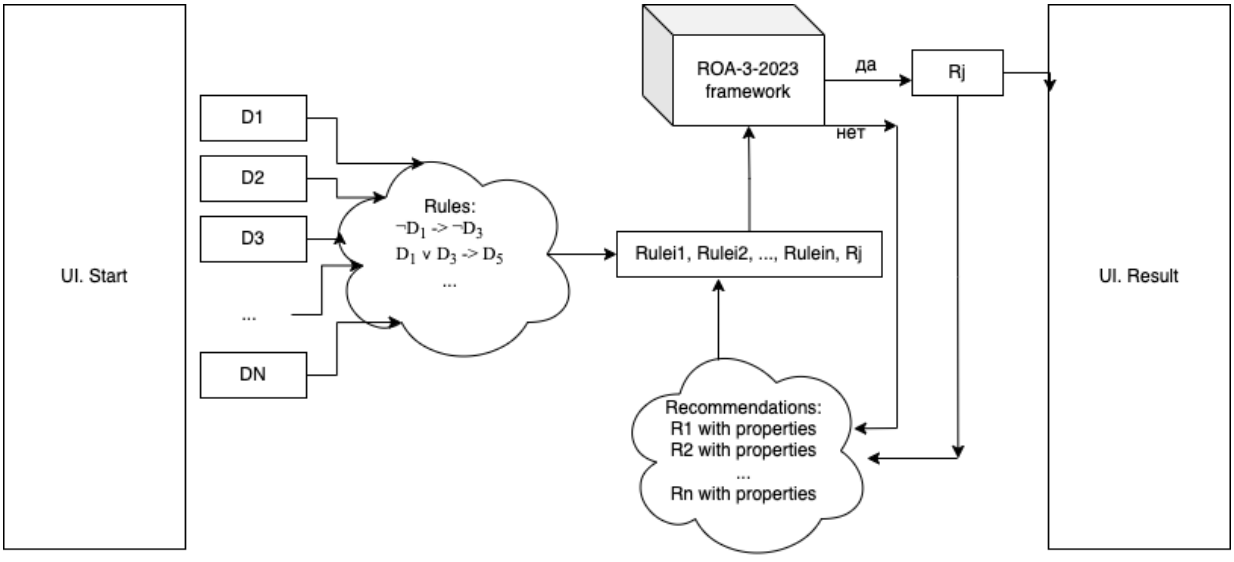
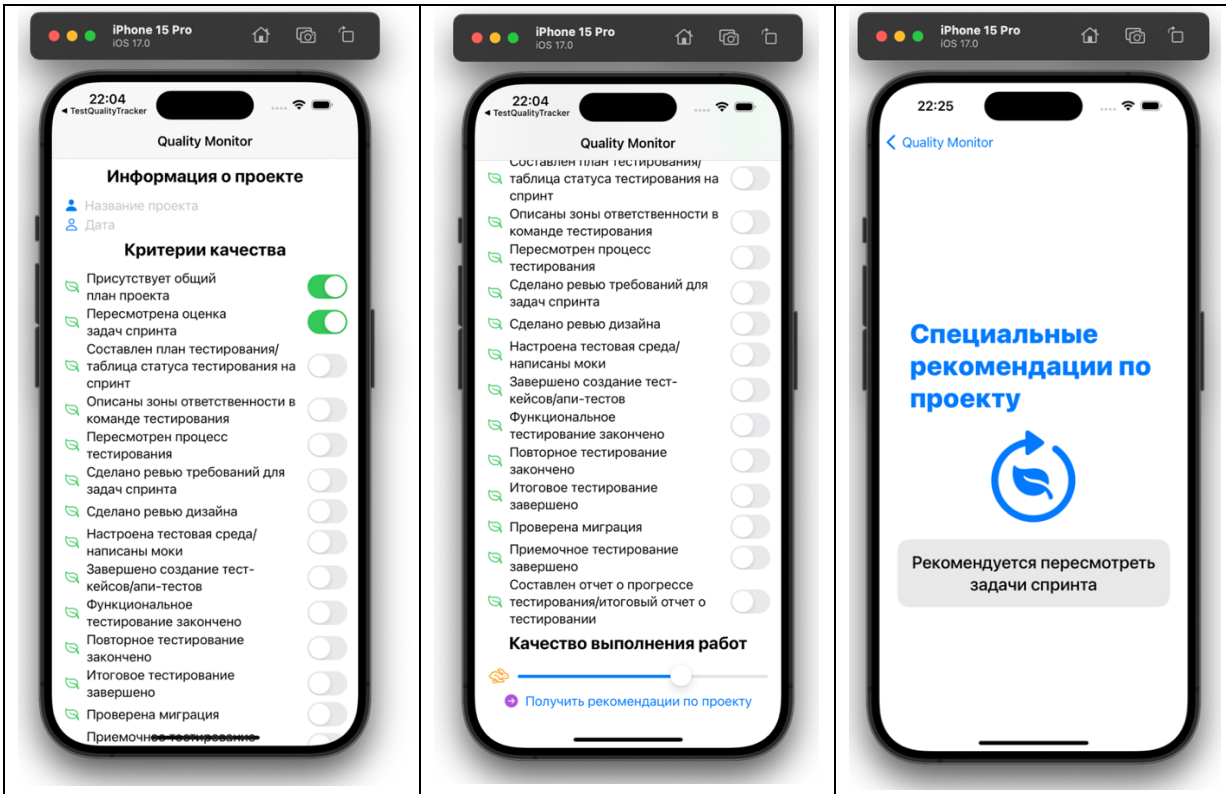


Рис. 4.16 – Архитектура мобильного приложения

Таким образом, архитектура и процесс работы приложения ориентированы на обеспечение высокого уровня качества разработки и тестирования, что

подтверждается комплексным подходом к планированию, реализации и контролю за ходом выполнения проекта. Реализация данных подходов и рекомендаций способствует непрерывному совершенствованию процессов разработки и тестирования, гарантируя создание качественного и надежного программного продукта.



а) Экран ввода данных	б) Экран ввода данных	в) Экран рекомендаций

Рис. 4.17 – Основные этапы работы мобильного приложения

Анализ качества тестирования вручную в рамках одной итерации в среднем занимает около 4 часов (по данным компании, где осуществлялось внедрение), что является проблемой в компаниях, где различных IT-проектов от 20 и больше. С помощью данного приложения время составляет в среднем от около 1,5 часов, а то и меньше. Использование приложения позволяет: сократить время тестирования, необходимое для анализа текущей ситуации, выявление проблем и разработку рекомендаций для их решения, в 2.5 раза; повысить точность предоставляемых рекомендаций по улучшению качества тестирования и степень обоснованности принимаемых решений; снизить

затраты ресурсов на проведение работ; обеспечить современный уровень автоматизации процедур тестирования.

4.5 Выводы

В разработке приложений для автоматического доказательства теорем с использованием метода резолюций, особое внимание уделяется структуре приложения, которая играет ключевую роль в обеспечении гибкости и адаптации к различным задачам. Такая структура включает в себя модуль ввода для приема исходных данных, ядро с реализацией алгоритмов метода резолюций и его вариантов, модуль управления памятью для эффективного хранения данных и графический пользовательский интерфейс для удобства использования. Эти компоненты вместе формируют многогранную систему, способную адаптироваться к разнообразным требованиям и обеспечивать высокую эффективность решения задач логического вывода.

В результате диссертационного исследования были достигнуты следующие ключевые результаты:

1. Основной фокус был направлен на разработку и внедрение алгоритмов для новых стратегий метода резолюций. Эти алгоритмы применимы как в рамках классической логики, так и в контексте нечеткой логики, что позволяет значительно расширить возможности логического вывода. Это позволило расширить возможности автоматического доказательства теорем, обеспечив высокую адаптируемость и эффективность в различных областях логического вывода и искусственного интеллекта.

2. В результате проведенных экспериментов и анализа полученных результатов можно сформулировать следующие ключевые выводы:

– Стратегии классической логики, работающие с весами предложений и на основе поиска минимального дизъюнкта, продемонстрировали наивысшую скорость обработки большинства задач, подтверждая свою эффективность и применимость в разнообразных вычислительных сценариях.

– Предложенные алгоритмы классической логики показали универсальность и способность адаптироваться к различным требованиям и условиям задач, что свидетельствует о их широком потенциале применения.

– В экспериментах классической логики подтверждена устойчивость алгоритмов к различным входным данным, что является важным качеством для решения вычислительных задач в условиях неопределённости.

– Временные показатели различных алгоритмов классической и нечеткой логики варьируются, но большинство из них работают быстрее, чем известный алгоритм стратегии полного перебора, за исключением рейтинговой стратегии.

– Некоторые стратегии нечеткой логики не находят решение, что показывают результаты экспериментов. Таким образом, открываются возможности для дальнейшей оптимизации и улучшения предложенных алгоритмов.

Эти выводы подчёркивают значимость проведённых исследований и предложенных алгоритмических решений для научного сообщества и практического применения, а также намечают пути для дальнейших разработок и улучшений в данной области.

3. Еще одним важным результатом является создание тщательно спроектированной структуры приложения, поддерживающей разнообразие подходов к резолюции и стратегии вывода, стало важным результатом исследования. Структура включает в себя модуль ввода для приема исходных данных, ядро с алгоритмами резолюций, модуль управления памятью и хранения данных для оптимизации ресурсов, а также графический пользовательский интерфейс для удобства использования. Эта комплексная и многогранная система обеспечивает гибкость и высокую эффективность приложения, адаптируемого к широкому спектру задач логического вывода.

ЗАКЛЮЧЕНИЕ

В диссертационной работе решена актуальная научная задача повышения эффективности метода резолюций в классической и нечеткой логике за счет разработки новых стратегий управления резолютивным выводом, при этом под эффективностью понимается быстродействие, оптимизация структуры графа вывода, способность обрабатывать неопределенные суждения. Совершенствование метода резолюций позволит расширить его применимость для разработки систем искусственного интеллекта, основанных на логической модели представления знаний.

В соответствии с целями и задачами исследования получены следующие результаты:

1. Принцип резолюций составляет основу методов машинной логики и применяется в интерактивных системах, базирующихся на задаче автоматического доказательства теорем. Данная задача является NP-полной и в общем случае может быть решена полным перебором. Однако особенности знаний позволяют управлять логическим выводом с помощью эвристических правил – стратегий, что приводит к повышению быстродействия соответствующих систем и способствует их эффективности. В диссертации проведен анализ существующих стратегий управления выводом и предложены новые критерии, учитывающие структурные, временные и сложностные аспекты их алгоритмической реализации.

2. Процесс работы с базами знаний, которые используют логические модели является ресурсозатратным, поэтому одним из основных требований к алгоритмам логического вывода в этом случае является необходимость обеспечения их быстродействия. Предложен комплекс эвристических стратегий для выбора дизъюнктов при построении резолютивного вывода, который учитывает особенности множества предложений на каждом шаге резолютивного вывода. Установлено, что самые быстрые стратегии позволяют повысить эффективность поиска решения в 1.8 раза, при этом не затрачивая

память на сохранение промежуточных решений (построенных дизъюнктов – резольвент), которые впоследствии оказываются лишними.

3. Неопределенность аргументации и выводов, характерных для неформальных суждений, значительно ограничивает применимость классического метода резолюций и обусловливает использование нечеткого логики. В диссертации получены условия, при выполнении которых нечеткая резольвента приводит к значимому логическому следствию. Исследована возможность использования для ее построения обобщений логических операций – треугольных норм и конорм. Разработаны стратегии управления выводом в нечетком методе резолюций.

4. В контексте современной разработки программного обеспечения, мониторинг качества проектной деятельности приобретает особое значение. Разработанные алгоритмы использовались при создании программного приложения, которое обеспечивает возможность оперативного и своевременного определения статуса тестирования и принятия соответствующих решений для обеспечения его качества в рамках IT-проектов по разработке мобильных и веб-приложений. В результате использования приложения достигается снижение затрат ресурсов на проведение аналитических работ на 60%, что способствует более высокому уровню автоматизации процедур тестирования и улучшению общей продуктивности труда. Такая оптимизация процесса позволяет не только экономить время в масштабах крупных компаний с 20 и более IT-проектами, но и повышает точность и обоснованность предлагаемых решений по улучшению качества тестирования.

5. По результатам исследований опубликовано 9 статей, среди которых 3 статьи без соавторов, 3 статьи опубликованы в рецензируемых журналах из Перечня ВАК, также получено свидетельство о регистрации программы для ЭВМ. Результаты диссертационной работы в форме алгоритмов, реализующих различные стратегии управления выводом, используются в учебном процессе и при выполнении выпускных квалификационных работ. Также разработано

мобильное приложение для отслеживания качества тестирования разрабатываемого программного обеспечения в IT-компании «Сёрф».

В целом совокупность полученных в диссертации теоретических и практических результатов позволяет сделать вывод о том, что цель исследований достигнута, сформулированные задачи решены.

Рекомендации по использованию. Научные результаты, полученные в рамках диссертационного исследования, имеют практическое значение и могут быть рекомендованы для разработки интеллектуальных информационных систем, в которых база знаний формируется на основе логической модели, основанной на исчислении предикатов первого порядка, постановка задачи сводится к проверке правильности рассуждений, а основу обработки знаний составляет метод резолюций.

Перспективы развития результатов исследования связаны с дальнейшим совершенствованием стратегий управления выводом, в частности, с динамическим выбором стратегий на каждой итерации метода резолюций. Также представляет интерес исследование нечеткого метода резолюций в зависимости от конкретного вида треугольных норм, поскольку в других задачах, связанных с их использованием, различия в получаемых результатах наблюдаются. Кроме того, имеет смысл рассмотреть случай, когда «правило дизъюнктивного силлогизма» (именно он формализует принцип резолюции) обобщается на другие типы информации (например, когда степень истинности высказываний оценивается интервальным числом или лингвистическим термом).

СПИСОК ЛИТЕРАТУРЫ

1. Вагин, В.Н. Алгоритмы параллельного логического вывода и исследование их эффективности на компьютерных системах / В.Н. Вагин, А.В. Деревянко, В.П. Кутепов // М.: Московский энергетический институт, 2017. – № 1. – С. 3-9.

2. Вагин, В.Н. Логический вывод в искусственном интеллекте / В.Н. Вагин // М.: Мир, 1992. – С. 358.

3. Вагин, В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В. Достоверный и правдоподобный вывод / Под ред. В.Н. Вагина, Д.А. Поспелова // М.: Физматлит, 2008 – С. 712.

4. Воронков, А. Автоматическое доказательство теорем. Энциклопедия искусственного интеллекта / А. Воронков // М.: Физматлит, 2002 – С. 456-467.

5. Леденева, Т.М. Формальные аксиоматические теории. Исчисление предикатов. Часть 2. / Т.М. Леденева. // Воронеж: Издательский дом ВГУ, 2020. – С. 45.

6. Леденева, Т.М. Формальные аксиоматические теории. Исчисление предикатов. Часть 1. / Т.М. Леденева, Е.М. Аристова. // Воронеж: Издательский дом ВГУ, 2016. – С. 34.

7. Леденева, Т.М., Лещинская, М.В. Анализ подходов к определению нечеткой резольвенты / Т.М. Леденева, М.В. Лещинская // Информатика и ее применения, 2024. – Т. 18. – Вып. 1. – С. 76-83. DOI: 10.14357/ 19922264240113

8. Леденева, Т.М., Лещинская, М.В. Вычисление резольвенты на основе отношения сходства в нечетком методе резолюций / Т.М. Леденева, М.В. Лещинская // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, 2023. – № 3. – С. 143-155. DOI: https://doi.org/10.17308/sait/1995-5499/2023/3/143-155

9. Леденева, Т.М., Лещинская, М.В. Метод резолюций и стратегии поиска опровержений / Т.М. Леденева, М.В. Лещинская // Вестник Воронежского государственного университета. Серия: Системный анализ и

информационные технологии, 2021. – № 1. – С. 98-111. DOI: https://doi.org/10.17308/sait.2021.1/3374

10. Лещинская М.В. Мобильное приложение для предоставления рекомендаций по питанию на основе метода резолюций / М.В. Лещинская // Сб. тр. Междунар. науч. конф. «Актуальные проблемы прикладной математики, информатики и механики», 2023. – С. 209-215.

11. Лещинская, М.В. Стратегия управления выводом в методе резолюций с использованием весов дизъюнктов / М.В. Лещинская // Сб. тр. Междунар. науч. конф. «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 12-14 декабря 2022 г.), 2022. – С. 1407-1411.

12. Лещинская, М.В. Реализация стратегии управления выводом на основе минимального дизъюнкта / М.В. Лещинская // Сб. тр. науч. конф. «Математика, информационные технологии, приложения», 2023. – С. 256.

13. Лещинская, М.В., Леденева, Т.М. Новая стратегия поиска опровержений методом резолюций / М.В. Лещинская, Т.М. Леденева // Сб. тр. Междунар. науч. конф. «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 13-15 декабря 2021 г.), 2021. – С. 1628-1633.

14. Ловленд, Д. Автоматическое доказательство теорем: Перспективы систем резолюций / Д. Ловленд // М.: Наука, 1978.

15. Нильсон Н. Искусственный интеллект. Методы поиска решений / Н. Нильсон // М.: Мир, 1973. – С. 272.

16. Новак, В., Перфильева, И., Мочкорж, И. Математические принципы нечеткой логики / В. Новак, И. Перфильева, И. Мочкорж // М.: Физматлит, 2006 – С. 252.

17. Поспелов, Д.А. Искусственный интеллект. Модели и методы: Справочник / Д. А. Поспелов // Радио и связь, 1990. – С. 304.

18. Тейз, А. Логический подход к искусственному интеллекту. От классической логики к логическому программированию / Тейз А. // М.: Мир, 1990. – С. 450.

19. Харрисон, Дж. Введение в логику и автоматическое доказательство теорем / Дж. Харрисон // М.: Техносфера, 2009.

20. Свидетельство о государственной регистрации программы для ЭВМ 2023664399 Российская Федерация. Библиотека алгоритмов метода резолюций «ROA-3-2023» / М.В. Лещинская; заявитель и правообладатель Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет». – № 2023664399; заявление 22.06.2023 ; опубл. 04.07.2023.

21. Baader, F. The Description Logic Handbook / Baader F. // New York: Cambridge University Press, 2003.

22. Bellman, R.E., Zadeh, L.A. Decision making in a fuzzy environment / R.E. Bellman, L.A. Zadeh // Management Science, 1970 – 17(4) – B141-B164.

23. Boden, M.A. Artificial Intelligence and Natural Man / M.A. Boden // Basic Books, 1977.

24. Bolignano, D. Automatic verification of cryptographic protocols with SETHEO / D. Bolignano // Journal of Computer Security, 1997 – 5(3) – P.191-212.

25. Boyer, R.S., Ait-Kaci, H., Lincoln, P., Nasr, R. The Efficient Implementation of Lattice Operations / R.S. Boyer, H. Ait-Kaci, P. Lincoln, R. Nasr // ACM Transactions on Programming Languages and Systems, 1988 – 11(1) – P. 115-146.

26. Boyer, R.S., Green, M.W., Moore, J.S. The Use of a Formal Simulator to Verify a Simple Real Time Control Program / R.S. Boyer, M.W. Green, J.S. Moore // Beauty is Our Business, Springer-Verlag, 1990 – P. 54-66.

27. Boyer, R.S., Moore, J.S. A Fast String Searching Algorithm / R.S. Boyer, J.S. Moore // Communications of the Association for Computing Machinery, 1977 – 20(10) – P. 762-772.

28. Boyer, R.S., Moore, J.S. A Theorem Prover for a Computational Logic / R.S. Boyer, J.S. Moore // In Automated Deduction - CADE-10, Lecture Notes in Computer Science 449, Springer-Verlag, 1990 – P. 1-15.

29. Boyer, R.S., Moore, J.S. MJRTY - A Fast Majority Vote Algorithm. / R.S. Boyer, J.S. Moore // In Automated Reasoning: Essays in Honor of Woody Bledsoe, Kluwer Academic Publishers, 1987 – P. 105-117.

30. Chang, C.L., Slagle, J.R. An Admissible and Optimal Algorithm for Searching AND/OR Graphs / C.L. Chang, J.R. Slagle // Artificial Intelligence. – 1971. – Vol. 2, No. 2. – P. 117-128. DOI: 10.1016/0004-3702(71)90006-3.

31. Chang, C.L., Slagle, J.R. Using Rewriting Rules for Connection Graphs to Prove Theorems / C.L. Chang, J.R. Slagle // Artificial Intelligence. – 1979. – Vol. 12, No. 2. – P. 159-178. DOI: 10.1016/0004-3702(79)90015-8.

32. Davis, M., Putnam, H. A computing procedure for quantification theory / M. Davis, H. Putnam // J. Assoc. Comput. Mach. – 1960. – No7. – P. 201-215.

33. Dietrich, E., Markman, A. Cognitive dynamics: Computation and representation regained / E. Dietrich, A. Markman // In: D. Scarborough, S. Sternberg (eds.) Invitation to Cognitive Science – MIT Press, 1998 – Vol. 4.

34. Dubois, D., Prade, H. Necessity and Resolution Principle / D. Dubois, H. Prade // IEEE Transactions on Systems, Man, and Cybernetics, 1987 – 17(3) – P. 474-478. DOI:10.1109/TSMC.1987.4309063

35. Ehrig, H., Kowalski, R.A., Levi, G., Montanari, U. TAPSOFT'87: Proceedings of the International Joint Conference on Theory and Practice of Software Development / H. Ehrig, R.A. Kowalski, G. Levi, U. Montanari // Pisa, Italy, March 23-27, 1987 – Volume 1: Advanced Seminar on Foundations of Innovative Software Development I and Colloquium on Trees in Algebra and Programming, Lecture Notes in Computer Science – Vol. 249.

36. Filev, D., Yager, R.R. Fuzzy models induced by alternative defuzzification methods / D. Filev, R.R. Yager // Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, New Orleans, 1996 – P. 457-462.

37. Guller, D. Hyperresolution for Gödel logic with truth constants / D. Guller // Fuzzy Sets and Systems 363, 2019 – P. 1-65.

38. Gunning, D. Explainable Artificial Intelligence / D. Gunning // Defense Advanced Research Projects Agency (DARPA), 2017.

39. Habiballa, H. Resolution principle and fuzzy logic. / H. Habiballa // In: E. Dadios (ed.), Fuzzy Logic Algorithms, Techniques, and Implementations, 2012 – Ch. 3, IntechOpen, London – P. 55–74.

40. Haugeland, J. Artificial Intelligence: The Very Idea / J. Haugeland // MIT Press, 1985.

41. Hillenbrand, T. Citius altius fortius: Lessons learned from the Theorem Prover WALDMEISTER / T. Hillenbrand // Electronic Notes in Theoretical Computer Science, 2003 – 86.1 – P. 1-13.

42. Klement E.P., Mesiar R., Pap E. Triangular norms. Position paper I: basic analytical and algebraic properties / E.P. Klement, R. Mesiar, E. Pap // Fuzzy Set and Systems 143, 2004 – P. 5-26.

43. Klement E.P., Mesiar R., Pap E. Triangular norms. Position paper II: general constructions and parameterized families / E.P. Klement, R. Mesiar, E. Pap // Fuzzy Set and Systems 145, 2004 – P. 439-454.

44. Klement E.P., Mesiar R., Pap E. Triangular norms. Position paper III: continuous t-norms / E.P. Klement, R. Mesiar, E. Pap // Fuzzy Set and Systems 145, 2004 – P. 439-454.

45. Kowalski, R.A. Directions for Logic Programming / R.A. Kowalski // Wissensbasierte Systeme, 1987 – P. 128-146.

46. Kowalski, R.A. The Limitation of Logic / R.A. Kowalski // ACM Conference on Computer Science, 1986 – P. 7-13.

47. Kowalski, R.A., Sergot, M.J. A Logic-based Calculus of Events. / R.A. Kowalski, M.J. Sergot // New Generation Computing, 1986 – 4(1) – P. 67-95.

48. Lawrence, C. P. Isabelle: The Next 700 Theorem Provers. / C.P. Lawrence // P. Odifreddi, Logic and Computer Science, 1990 – P. 361-386.

49. Lawrence, C.P. The Foundation of a Generic Theorem Prover / C.P. Lawrence // Journal of Automated Reasoning 5, 1989 – P. 363-397.

50. Ledeneva, T., Leshchinskaya, M. On the New Inference Control Strategy in the Resolution Method / T. Ledeneva, M. Leshchinskaya // 4th International Conference on Control Systems, Mathematical Modeling, Automation and Energy

Efficiency (SUMMA), Lipetsk, Russian Federation. – 2022. – P.18, DOI: 10.1109/SUMMA57301.2022.9973441.

51. Ledeneva, T. Additive Generators of Fuzzy Operation in the Form of Linear Fractional Function / T. Ledeneva // Fuzzy Set and Systems, 2020 – P. 1-24. DOI: http://dx.doi.org/10.1016/j.fss.2019.03.005

52. Ledeneva, T. New family of triangular norms for decreasing generators in the form of a logarithm of a linear fractional functions / T. Ledeneva // Fuzzy Sets and Systems, 2022 – P. 37-54. DOI: https://doi.org/10.1016/j.fss.2020.11.020

53. Lee, R.C.T. Fuzzy Logic and the Resolution Principle / R.C.T Lee // Journal of the ACM, 1972 – 19(1) – P. 109-119. DOI:10.1145/321679.321688

54. Lee, S.-J., Plaisted, D.A. Problem solving by searching for models with a theorem prover / S.-J. Lee, D.A. Plaisted // Artificial Intelligence, 1994 – 69(1-2) – P. 205-233.

55. Leitsch, A. The resolution calculus / A. Leitsch // Texts in Theoretical Computer Science. An EATCS Series. Springer, 1997. ISBN 978-3-642-60605-2.

56. Letz, R., Mayr, K., Goller, C. SETHEO: A high-performance theorem prover / R. Letz, K. Mayr, C. Goller // Journal of Automated Reasoning, 1994 – 8(2) – P. 183-212.

57. Lifschitz, V. Formal theories of action (preliminary report) / V. Lifschitz // In: Proc. IJCAI, 1987 – P. 966-972.

58. Luckham, D., Sankar, S., Takahashi, S. (1991). Two-dimensional pinpointing: debugging with formal specifications / D. Luckham, S. Sankar, S. Takahashi // IEEE Software, 1991 – 8 – P. 74-84. DOI:10.1109/52.62935

59. Luckham, D.C. Rapide: A language and toolset for causal event modelling of distributed system architectures / D.C. Luckham // Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1998 – 1368 – P. 88-96. DOI:10.1007/3-540-64216-1_42

60. Luckham, D.C., Vera, J., Bryan, D., Augustin, L., Belz, F. Partial orderings of event sets and their application to prototyping concurrent, timed systems / D.C.

Luckham, J. Vera, D. Bryan, L. Augustin, F. Belz // The Journal of Systems and Software, 1993 – 21 – P. 253-265. DOI:10.1016/0164-1212(93)90027-U

61. Marconi, D. The Interaction of Humans and Machines in the Age of Artificial Intelligence / D. Marconi // AI & Society, 2020. – T. 35. – P. 547-558.

62. Marr, D. A Computational Investigation into the Human Representation and Processing of Visual Information / D. Marr // MIT Press, 1982.

63. McCarthy, J., Hayes, P.J. Some philosophical problems from the standpoint of artificial intelligence / J. McCarthy, P.J. Hayes // In: B. Meltzer, D. Michie (eds.) Machine Intelligence 4. Edinburgh University Press, 1969.

64. McCune, W. Experiments with Discrimination-Tree Indexing and Path Indexing for Term Retrieval / W. McCune // Journal of Automated Reasoning, 1992 – 9(2) – P. 147–167. DOI:10.1007/BF00245458.

65. McCune, W., Wos, L. Otter: The CADE-13 Competition Incarnations / W. McCune, L. Wos // Journal of Automated Reasoning, 1997 – 18(2) – P. 211–220. DOI:10.1023/A:1005843632307.

66. Miller, T. Explanation in artificial intelligence: Insights from the social sciences / T. Miller // Artificial Intelligence, 2019 – Vol. 267 – P. 1-38.

67. Mondal, B., Raha, S. Approximate reasoning in fuzzy resolution / B. Mondal, S. Raha // International Journal of Intelligence Science, 2013 – 3(2) – P. 86–98. DOI: 10.4236/ijis.2013.32010

68. Moser, M., Ibens, O., Letz, R., Steinbach, J., Goller, C., Schumann, J., Mayr, K. SETHEO and E-SETHEO - The CADE-13 Systems / M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, K. Mayr // Journal of Automated Reasoning, 1996 – 18(2) – P. 237-246.

69. Mukaidono, M. Fuzzy Inference of Resolution Style / M. Mukaidono // In: R. R. Yager, Ed., Fuzzy Set and Possibility Theory, New York: Pergamon Press, 1998 – P. 224-231.

70. NewBorn, M. Automated Theorem Proving: Theory and Practice. / M. NewBorn // Springer Verlag. 2000 – P. 231.

71. Newell, H.A. Human Problem Solving / H.A. Newell // Prentice-Hall, 1972.

72. Nguyen, T.M.T, Tran, D.A.K Resolution Method in Linguistic Propositional Logic / T.M.T. Nguyen, D.A.K. Tran // International Journal of Advanced Computer Science and Applications, 2016 – 7(1) – P. 672-678.

73. Nie, X., Plaisted, D.A. Refinements to depth-first iterative-deepening search in automatic theorem proving / X. Nie, D.A. Plaisted // Artificial Intelligence, 1989 – 41(2) – P. 223-235.

74. Plaisted, D.A. Complete problems in the first-order predicate calculus / D.A. Plaisted // Urbana, Ill.: Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1979.

75. Plaisted, D.A. Inference rules for unsatisfiability / D.A. Plaisted // Urbana: Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1979.

76. Raha, S., Ray, K.S. 1994. Approximate Reasoning Based on Generalised Disjunctive Syllogism / S. Raha, K.S. Ray // Fuzzy Sets and Systems, 1994 – 61(2) – P. 143-151.

77. Riazanov, A., Voronkov, A. The design and implementation of VAMPIRE / A. Riazanov, A. Voronkov // AI Communications, 2002 – 15(2–3/2002) – P. 91-110.

78. Robinson, J.A. A Machine Oriented Logic Based on the Resolution Principle // Journal of the ACM, 1965 – 12(1) – P. 23-41. DOI:10.1145/321250.321253

79. Samokhvalov, Y. Proof of Theorems in Fuzzy Logic Based on Structural Resolution / Y. Samokhvalov // Cybernetics and Systems Analysis, 2019 – 55 – P. 1-13. DOI:10.1007/s10559-019-00125-8.

80. Schumann, J. E-SETHEO: Design, Configuration and Use of a Parallel Automated Theorem Prover / J. Schumann // Journal of Automated Reasoning, 1997 – 18(2) – P. 237-246.

81. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T. The British Nationality Act as a Logic Program / M.J. Sergot, F. Sadri, R.A.

Kowalski, F. Kriwaczek, P. Hammond, H.T. Cory // Communications of the ACM, 1986 – 29(5) – P. 370-386.

82. Sowa, J.F. Semantic networks / J.F. Sowa // In: S.C. Shapiro (ed.) Encyclopedia of Artificial Intelligence. John Wiley, Sons, 1987 – P. 1493-1511.

83. Suttner, C.B., Sutcliffe, G. SPTHEO - A Parallel Theorem Prover / C.B. Suttner, G. Sutcliffe // Journal of Automated Reasoning, 1998 – 20(3) – P. 317-337.

84. Tammet, T. A Resolution Theorem Prover for Intuitonistic Logic. / T. Tammet // 1996 – P. 2-16. DOI:10.1007/3-540-61511-3_65.

85. Viedma, M.A.C., Morales, R.M., Sanchez I.N. Fuzzy Temporal Constraint Logic: A Valid Resolution Principle / M.A.C. Viedma, R.M. Morales, I.N. Sanchez // Fuzzy Sets and Systems, 2001 – 117(2) – P. 231-250.

86. Voronkov, A. The anatomy of vampire / A. Voronkov // Journal of Automated Reasoning, 1995 – 15(2) – P. 237–265. DOI:10.1007/BF00881918.

87. Wos, L., Pieper, G.W. 3.11 OTTER and Earlier Automated Theorem-Proving Programs / L. Wos, G.W. Pieper // A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning. World Scientific, 1999. ISBN 978-9810239107.

88. Yager, R.R. Fuzzy logic control with discrete outputs / R.R. Yager // Proceedings of World Congress on Neural Networks, Washington, DC, 1995 – II – P. 595-601.

89. Yager, R.R., Kelman, A. Fuzzy decision making in hierarchical environments / R.R. Yager, A. Kelman // Proceedings of the World Automation Congress, Montpellier, 1995 – P. 717-724.

90. Zadeh, L.A. A computational approach to fuzzy quantifiers in natural languages / L.A. Zadeh // Computers and Mathematics with Applications: Special Issue on Computational Linguistics, 1983 – 9(1) – P. 149-184.

91. Zadeh, L.A. A fuzzy-algorithmic approach to the definition of complex or imprecise concepts / L.A. Zadeh // International Journal of Man-Machine Studies, 1976 – 8(3) – P. 249-291.

92. Zadeh, L.A. Fuzzy logic and the calculus of fuzzy if-then rules / L.A. Zadeh // In Proceedings of the 22nd International Symposium on Multiple-Valued Logic, IEEE Computer Society Press, 1992 – P. 480.

93. Zadeh, L.A. Fuzzy sets / L.A. Zadeh // Information and Control, 1965 – Vol. 8 – No. 3 – P. 338-353.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023664399

ROA-3-2023

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

Автор(ы): ***Лещинская Мария Владимировна (RU)***

Заявка № **2023663031**
Дата поступления **22 июня 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***04 июля 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

УТВЕРЖДАЮ
Декан факультета прикладной математики, информатики и механики
_____Медведев С. Н.
«17» января 2024 г.

АКТ

о внедрении (использовании) результатов

кандидатской диссертационной работы

Лещинской Марии Владимировны

Комиссия в составе:

председатель к.ф.-м.н., доц. С.Н. Медведев

члены комиссии: д.т.н., проф. Ю.В. Бондаренко, к.ф.-м.н., доц. Е.М. Аристова

составили настоящий акт о том, что результаты диссертационной работы Лещинской Марии Владимировны «Разработка новых стратегий управления выводом в классическом и нечетком методе резолюций», представленной на соискание ученой степени кандидата технических наук, использованы в деятельности кафедры вычислительной математики и прикладных информационных технологий факультета прикладной математики, информатики и механики при проведении лекционные и практических занятий по дисциплинам «Математическая логика и теория алгоритмов», «Интеллектуальные информационные технологии», а также при выполнении выпускных квалификационных работ.

Результаты диссертационной работы Лещинской Марии Владимировны:

– комплекс стратегий управления выводом в форме эвристических правил для выбора дизъюнктов при построении резольвент в классическом методе резолюций;

– критерии для сравнения эффективности различных стратегий управления выводом, отличающиеся учетом структурных характеристик графа вывода, а также временных и сложностных характеристик алгоритмической реализации стратегий;

– алгоритмы построения резолютивного вывода для нечеткого метода резолюций, отличающиеся различными подходами к определению нечеткой резольвенты.

Председатель комиссии:

Декан факультета прикладной математики, информатики и механики, к.ф-м.н, доцент — С.Н. Медведев

Члены комиссии:

Председатель научно-методического совета факультета ПММ, д.т.н., проф. кафедры математических методов исследования операций — Ю.В. Бондаренко

Зам. зав. кафедрой вычислительной математики и прикладных информационных технологий, к.ф-м.н., доцент — Е.М. Аристова

УТВЕРЖДАЮ
Директор ООО «Сёрф»
Макеев Владимир Геннадьевич
«28» 01 2024 г.

АКТ

о внедрении (использовании) результатов

кандидатской диссертационной работы

Лещинской Марии Владимировны

Комиссия в составе:

председатель Пластинин А.А.,

члены комиссии: Пастухов В.М., Чаусова Д.В.

составили настоящий акт о том, что результаты диссертационной работы Лещинской Марии Владимировны «Разработка новых стратегий управления выводом в классическом и нечетком методе резолюций», представленной на соискание ученой степени кандидата технических наук, использованы в деятельности отдела качества IT-компании ООО “Сёрф” при разработке инструмента для отслеживания качества тестирования на проекта в виде программного комплекса, в основе которого лежит мобильное приложение на iOS для определения качества проекта с использованием алгоритмов метода резолюций.

Заключение: Использование программного комплекса позволяет: сократить время разработки рекомендаций на выявленные проблемы в 2.5 раза; повысить удобство и точность предоставляемых советов по улучшению качества тестирования на проектах; сократить затраты на проведение работ.

Председатель комиссии:

Операционный директор Пластинин А.А.

Члены комиссии:

Руководитель отдела качества Пастухов В.М.

Инженер по тестированию Чаусова Д.В.