МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

на правах рукописи

ТРОЙНИНА АНАСТАСИЯ СЕРГЕЕВНА

УДК 004.02 : 004.825 : 004.942

МОДЕЛИ, МЕТОДИ И ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ СОЗДАНИЯ И СОПРОВОЖДЕНИЯ ЗНАНИЕОРИЕНТИРОВАННЫХ СИСТЕМ КОНТРОЛЯ

Специальность 05.13.06 –Информационные технологии

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
Рувинская Виктория Михайловна
кандидат технических наук, доцент

Одесса – 2015

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ СОКРАЩЕНИЙ

АРМ – автоматизированное рабочее место

АСУ ПС – автоматизированная система управления программными средствами

БЗ – база знаний

ДНФ – дизъюнктивная нормальная форма

ИГ – именная группа

ИИ– искусственный интеллект

ИТ – информационная технология

КНФ – конъюнктивная нормальная форма

ЛПР – лицо принимающее решение

МДНФ – минимальная дизъюнктивная нормальная форма

НЭС – нечеткая экспертная система

ООП – объектно-ориентированное программирование

ППФ – правильно построенная формула

ПКС – персональные компьютерные системы

ПрО – предметная область

Сетевая модель OSI – Open Systems Interconnection basic reference model

СПО – системное программное обеспечение

ТДМ –техническая диагностика и мониторинг

ЦСП– цифровой сигнальный процессор

ЭС – экспертная система

ЭСММ – экспертные системы морского мониторинга

ЯПЗ – язык представления знаний

BRMS – Business Rule Management System

CLIPS – C Language Integrated Production System

CSV – Comma-Separated Values

FTP – File Transfer Protocol

HTTP – HyperText Transfer Protocol

IDEA – International Data Encryption Algorithm

MAC – Media Access Control

SADT – Structured Analysis and Design Technique

SAT – SATisfiability problem

SQL – Structured Query Language

UML – Unified Markup Language

ВВЕДЕНИЕ

Актуальность темы. Современные требования к уровню безопасности с целью безаварийного функционирования объектов обусловливают необходимость совершенствования процесса автоматизированного контроля. Следовательно, к функциям контроля относятся, кроме наблюдения, анализ параметров объекта для определения его состояния и предоставления обработанной информации лицу, принимающему решение.

Специфика систем контроля, которые рассматриваются в работе, заключается в том, что количество состояний объектов ограничено, рассматриваемые параметры могут быть различными, то есть качественными, когда числовые параметры объектов поступают с датчиков и конвертируются в нечеткие параметры, или логическими; исследуемые системы не являются системами реального времени; условия контроля могут меняться. Перечисленные свойства систем контроля характерны для таких предметных областей (ПрО), как соблюдение требований охраны труда, экология и экономика, медицина и санитария, техника и другие. Исследуемые системы контроля могут работать как автономно, так и включаться в качестве компонента в существующие системы контроля, в том числе, в системы автоматизированного мониторинга и управления.

Для принятия решений о состоянии объектов применяются правила, которые обычно встроены в систему контроля. Однако условия контроля для большинства систем меняются довольно часто, и для того, чтобы упростить сопровождение таких систем, возникает необходимость в адаптации, то есть настройке правил, а изменения требуют значительных затрат, что приводит к снижению оперативности внесения изменений и сокращению достоверности решений. Кроме того, во многих традиционных системах контроля каждый параметр оценивается отдельно без взаимосвязи с другими, а это негативно влияет на достоверность принятия решений по результатам контроля.

В связи с тем, что задача контроля не является полностью формализованной, условия контроля обычно описаны декларативно и при автоматизации требует накопленного опыта экспертов, для комплексного анализа объектов с возможностью оперативного внесения изменений предлагается использование знаниеориентированных систем.

Однако внедрение таких систем сдерживается из-за неудобного режима формирования и обновления знаний. Следовательно, существует противоречие между потенциальными возможностями интеллектуальных знаниеориентированных систем и низким уровнем их практического использования из-за несовершенства средств информационной поддержки ведения баз знаний. Поэтому **актуальной и нерешенной является научно-техническая задача** разработки новых моделей, методов и ИТ создания и сопровождения систем контроля на базе знаниеориентированного подхода.

Предложено использование такой модели правил, как И/ИЛИ-граф, которая дает возможность на ранних этапах при структурировании знаний с учетом специфики исследуемых систем контроля применять методы математической логики для анализа правил и на этой основе разработать инструмент, поддерживающий функционально достаточный набор операций для описания рассмотренных выше ПрО. Таким образом, БЗ содержит меньше ошибок и неточностей, что при ее использовании ведет к улучшению достоверности принятых решений и в конечном итоге к уменьшению количества аварийных ситуаций.

Связь работы с научными программами, планами, темами. Диссертация выполнялась в соответствии с планом НИР кафедры системного программного обеспечения ОНПУ №38-73 (№ ДР 0110U008193) «Методи, моделі та засоби аналізу та підвищення продуктивності комп’ютерних систем».

Цель и задачи исследования. *Целью* исследования является повышение достоверности контроля, а также предоставление возможности оперативного внесения изменений в правила за счет разработки новых моделей, методов и

ИТ создания и сопровождения систем контроля на базе знаниеориентированного подхода.

Для достижения этой цели решены следующие *задачи*:

– проанализированы существующие средства и системы контроля объектов, а также способы создания и поддержки правил контроля с целью определения путей их развития;

– проведена постановка задачи контроля, определены особенности структуры знаний для контроля, которая остается неизменной для различных ПрО, в целях усовершенствования средств проверки и поддержки знаний в актуальном состоянии;

– усовершенствованы модели правил в виде И/ИЛИ-графа и булевых выражений с целью построения, визуализации, интерактивной работы, создания условий для оперативного внесения изменений и проверки правил контроля;

– разработаны методы проверки правил контроля на противоречивость, полноту, достижимость состояний;

– запроектирован и разработан редактор правил;

– создана ИТ для построения и сопровождения знаниеориентированных систем контроля;

– проведена апробация разработанных методов и ИТ для нескольких ПрО, исследована их результативность при использовании для принятия решений и в обучении.

Объектом исследования является процесс создания и внедрения систем контроля, основанных на знаниях.

Предметом исследования – модели, методы и ИТ создания и сопровождения знаниеориентированных систем контроля.

Методы исследования. Для визуализации и интерактивной работы с правилами используется теория графов, при представлении И/ИЛИ-графа в виде булевых выражений и их преобразованиях применяется математическая логика, в частности булева алгебра; для определения противоречивости посылок правил используется теория выполнимости булевых формул SAT

(SATisfiability problem); для разработки редактора правил используется объектно-ориентированный подход (ООП).

Научная новизна исследования заключается в разработке новых и совершенствовании существующих моделей и методов построения и сопровождения знаниеориентированных систем контроля, в совокупности обеспечивающих повышение достоверности контроля и предоставляющие возможность оперативного внесения изменений в правила. В диссертационной работе получены следующие новые научные результаты:

– *получила дальнейшее развитие* математическая модель правил в виде И/ИЛИ-графа, которая, в отличие от существующих моделей, основывается на разделении правил на группы в зависимости от состояния объекта и содержит специальную разметку графа с учетом дополнительной информации о нечетких и взаимно противоположных параметрах, вершине-состоянии, структуре правил, что позволило визуализировать правила двумя способами, интерактивно работать с ними, обеспечивать нечеткий логический вывод при их применении, а также за счет преобразования путей в графе, соответствующих правилам, в логические выражения, разработать методы проверки правил;

– *получила дальнейшее развитие* математическая модель правил в виде логических булевых выражений в связи с применением ее для задач контроля, которая, в отличие от существующей, основывается на разделение правил на группы в зависимости от состояния объекта контроля и содержит формулы как для «прямых», так и для «инверсных» правил, что позволяет проверять правила на противоречивость, полноту, достижимость состояний на ранних этапах проектирования БЗ;

– *впервые разработан метод* проверки противоречивости посылок правил контроля с использованием модели правил в виде булевых выражений на основе задачи выполнимости булевых формул (SAT), который, в отличие от существующих, позволяет выявлять противоречия между посылками как

внутри каждого правила, так и между правилами на этапе концептуализации знаний, что позволяет уменьшить количество ошибок в БЗ;

– *впервые разработан метод* проверки правил контроля на полноту с использованием модели правил в виде булевых выражений, который в отличие от существующих основан на визуализации как «прямых», так и «инверсных» правил, в которых выводы противоположны выводам исходных правил, что позволяет выявлять и дополнять отсутствующие в правилах знания в автоматизированном режиме конструирования нового варианта БЗ;

– *получил дальнейшее развитие* метод проверки достижимости в правилах контроля состояния объектов, который, в отличие от существующих, основан на модели правил контроля в виде булевых выражений, что позволяет удалять правила, которые не используются для определении состояния объекта.

Практическое значение полученных результатов. На основе созданных моделей и методов автором разработана ИТ построения знаниеориентированных систем контроля с использованием созданного инструментального средства в виде редактора правил для конструирования и проверки правил. На основе предложенной ИТ и редактора правил были созданы действующие прототипы ЭС контроля: первый, предназначенный для контроля за безопасной работой с электроустановками, был введен в эксплуатацию в коммунальном предприятии «Одесскомунтранс». Использование системы позволило увеличить достоверность принятых диспетчером решений на 12% и уменьшить количество несчастных случаев при выполнении работ в течение нескольких месяцев наблюдения на 32%. Второй прототип ЭС для контроля за работой компьютерной сети внедрен в ООО «Свитеко», использовался для системных администраторов и позволил уменьшить время их обучения на 20% без ухудшения качества. При модификации правил контроля необходимое время для внесения изменений в систему уменьшается до 3-4 часов, а внесение изменений в программный код занимает не менее нескольких суток.

При применении разработанного редактора правил ошибки, связанные с противоречивостью, появлялись и исправлялись в среднем в 10% правил,

благодаря проверке достижимости в правилах состояний объекта удалось извлечь в среднем 5% правил, база знаний для пополнения ее недостающими правилам была увеличена в среднем на 12%. Результаты исследований также внедрены в учебном процессе подготовки студентов по направлению 6.050103 "Программная инженерия" на кафедре «Системное программное обеспечение» ОНПУ.

Личный вклад соискателя заключается в разработке моделей, методов и технологии разработки знаниеориентированных систем контроля и мониторинга.

Автором предложена модель для построения, визуализации, интерактивной работы с правилами контроля [70, 22, 74, 76]; разработаны методы для проверки качества правил [72, 22]; проведен анализ систем для принятия решений [69, 24,]; спроектирован и реализован редактор правил, который поддерживает основные этапы предложенной информационной технологии [71, 72]; разработаны и опробированы ЭС для двух ПрО [73, 75, 77, 78].

Апробация научных результатов диссертации. Результаты исследования докладывались на Всеукраинской научно-практической конференции «Сучасні тенденції розвитку інформаційних технологій в науці, освіті та економіці» (г. Луганск, 2011 г.), на Международной научно-технической конференции «Обчислювальний інтелект (OI-2011)» (г. Черкассы), на международных научно-практических конференциях «Современные информационные и электронные технологии» (г. Одесса, 2011 г., 2012 г.), на международной конференции «Dependable systems, Services & Tehnologies»(г. Севастополь, 2012), на научно–практической конференции «Інформаційні технології в освіті та управлінні (ITEM)» (г. Новая Каховка, 2013 г.), на международной конференции студентов и молодых ученых «Современные информационные технологии 2014» (г. Одесса, 2014 г.), на Международной научно технической конференции «Информационные технологи в металлургии и машиностроении» (г. Днепропетровск, 2015 г.), на Международной научно-практической Интернет-конференции «Молодежь в технических науках: ис-

следования, проблемы, перспективы» (г. Винница, 2015 г.), на XII научно-технической конференции «LOGISTYKA SYSTEMY TRANSPORTOWE BEZPIECZEŃSTWO W TRANSPORCIE» (г. Szczyrk, 20 - 23 апреля 2015 г.), на научном семинаре в Чехии г.Брно в рамках стажировки в университете «Masaryk», факультет «Информационных технологий», в программе ERASMUS MUNDUS BACKIS, на научном семинаре кафедры СПО ОНПУ «Интеллектуальные системы и современные информационные технологии».

Публикации. По результатам исследований опубликовано 12 научных трудов, в том числе пять статей в журналах из перечня специализированных изданий МОН Украины, 1 статья в зарубежном издании, 6 – в сборниках научных трудов конференций.

РАЗДЕЛ 1
АНАЛИЗ АВТОМАТИЗИРОВАННЫХ СИСТЕМ КОНТРОЛЯ И ЗНАНИЕОРИЕНТИРОВАННЫХ СИСТЕМ

1.1 Понятие контроля объектов и процессов. Средства контроля

Данная работа посвящена исследованию подходов к построению знание-ориентированных систем контроля и созданию методов их разработки.

Под контролем понимается наблюдение и проверка процесса функционирования и фактического состояния объектов [34]. Контроль является процессом, обеспечивающим достижение поставленных целей путём сравнения фактического состояния объекта с желаемым. Необходимость совершенствования процесса автоматизированного контроля связана с современными требованиями к уровню безопасности с целью безаварийного функционирования объектов. К функциям контроля относят непосредственное наблюдение, а также анализ параметров объекта для определения его состояния, которое представляет собой нормальное функционирование или аварию, и, при необходимости, предоставление обработанной информации лицу, принимающему решение (ЛПР).

Для реализации контроля необходимо выполнение следующих условий:

– установление стандартов деятельности, подлежащих проверке, в частности, с помощью нормативов;

– корректировка процессов, если достигнутые результаты существенно отличаются от установленных стандартов;

– контроль не должен быть чрезмерным, то есть, выполняться слишком часто, так как срабатывание контрольных механизмов при несущественных отклонениях фактического состояния объекта от заданного делают систему неэкономичной.

Рассмотрим далее различные классификации процессов контроля. Контроль может осуществляться по множеству различных направлений: технический контроль, энергетический, экологический, санитарный, контроль и

надзор за соблюдением требований охраны труда, финансовый контроль и т.д. В зависимости от видов объектов контроль бывает регулярным или нерегулярным, также может проводиться в виде специальных проверок. Относительно времени реализации контролируемых решений и действий контроль разделяют на предварительный, текущий и последующий.

Для принятия решений о состоянии объектов применяются правила, которые для большинства систем меняются довольно часто.

В данной работе исследование проводится на примере систем технического контроля и систем надзора за соблюдением требований электробезопасности. Первая ПрО выбрана в связи с тем, что технические объекты активно развиваются и по мере их усложнения требуют развитого контроля во время работы. Для второй ПрО нормативные документы присутствуют лишь в текстовом виде, то есть систем контроля по электробезопасности не существует.

Специфика рассмотренных в работе видов контроля заключается в следующем:

– контроль осуществляется обычно нерегулярно, то есть при специальных проверках, например, при необходимости проведения работ или проверки состояния объекта;

– по отношению ко времени реализации представляет собой предыдущий контроль и текущий при мониторинге работы объектов;

– количество состояний объектов ограничено (нормальное состояние и аварийные);

– анализируемые параметры объектов могут быть различными, то есть качественными, когда числовые параметры объектов поступают с датчиков и конвертируются в нечеткие параметры, или логическими;

– исследуемые системы не являются системами реального времени;

– условия контроля могут меняться.

Перечисленные свойства систем контроля характерные для таких предметных областей (ПрО), как соблюдение требований охраны труда, экология

и экономика, медицина и санитария, техника и другие. Исследуемые системы контроля могут работать как автономно, так и включаться в качестве компонента в существующие системы контроля, а также, в частности, в системы автоматизированного мониторинга и управления.

Современный уровень систем контроля подразумевает использование развитых средств контроля, в частности, таких как сенсорные сети и машинное зрение. Далее рассмотрим **средства контроля объектов на основании данных, полученных от узлов сенсорной беспроводной сети.** Беспроводная сенсорная сеть (WSN) – это распределенная сеть из множества датчиков (датчики температуры, давления, освещенности, уровня вибрации, местоположения [2] и т.п.) и исполнительных устройств, объединенных между собой посредством радиоканала, обладает способностью к самоорганизации в сеть для передачи информации. Это позволяет устройству проводить измерения, самостоятельно проводить начальную обработку данных и поддерживать связь с внешней информационной системой. Причем, зона покрытия подобной сети может быть от нескольких метров до нескольких километров за счет способности ретрансляций сообщений от одного элемента к другому [20].

Гибкая архитектура и снижение затрат при монтаже выделяют беспроводные сети интеллектуальных датчиков среди других беспроводных и проводных интерфейсов передачи данных, особенно когда речь идет о большом количестве соединенных между собой устройств. Сенсорная сеть позволяет подключать до 65000 устройств [2].

Технология ретрансляции ближней радиосвязи 802.15.4/ZigBee [26], известная как «Сенсорные сети», является одним из направлений развития самоорганизующихся отказоустойчивых распределенных систем наблюдения и управления ресурсами и процессами. Сегодня технология беспроводных сенсорных сетей является единственной беспроводной технологией, с помощью которой можно решить задачу контроля и мониторинга [25], которые критичны к времени работы датчиков. Основной областью применения является контроль и мониторинг измеряемых параметров физических сред и объектов.

Сейчас беспроводные сенсорные сети используются для наблюдения за состоянием стационарных объектов в следующих областях [12]: своевременное выявление возможных отказов исполнительных механизмов по контролю таких параметров, как вибрация, температура, давление и т.п.; контроль доступа в режиме реального времени до отдельных систем объекта мониторинга; автоматизация инспекционного обслуживания промышленных активов; управление коммерческими активами; применение как компоненты в энерго и ресурсосберегающих технологиях; контроль эко-параметров окружающей среды.

Инновационной является идея использования сенсорных сетей для отслеживания именно подвижных узлов [21], это открывает совершенно новую область применения данной технологии и позволяет использовать сенсорные сети для навигации, в частности, внутри зданий.

Основой для создания такой системы является беспроводная сенсорная сеть, которая устанавливается по всему зданию. Сама идея отслеживания заключается в следующем. Сеть состоит из стационарных узлов, положение которых заранее известно, и подвижных узлов, которые нужно отслеживать. Стационарные узлы служат как опорные точки и инфраструктура для передачи информации. Каждый стационарный узел, кроме сетевого интерфейса, имеет датчики, способные получать информацию об окружающей среде, в частности о расстоянии до подвижного узла. Стационарные узлы организуются в сеть с произвольной топологией [3], все зависит от реализации. Динамические узлы не требуют наличия датчиков, а имеют только сетевой интерфейс.

Подобные системы могут найти применение в различных областях: на объектах с повышенной опасностью можно координировать движение и действия работников зависимости от состояния объекта или окружающей среды; в больницах – наблюдение за больными, у которых есть проблемы с памятью, это позволит не только наблюдать за их поведением, но и в необходимых случаях координировать их движение по зданию больницы, предостав-

ляя подсказки; в больших домах со сложной инфраструктурой, таких как торговые центры, развлекательные центры, бизнес-центры, гостиницы целесообразна разнообразная навигация (прокладка маршрутов) для посетителей и в других областях.

Машинное зрение как средство контроля — это применение компьютерного зрения для промышленности и производства [5].

Обработка изображений, или машинное зрение, представляет собой важный инструмент оптимизации и автоматизации контроля и мониторинга производственных процессов. Промышленные камеры обеспечивают сбор, хранение и архивирование важной видеоинформации, на основе которой пользователи могут принимать решения, которые было бы невозможно принять в отсутствие данных, полученных камерой.

Системы машинного зрения, оборудованные промышленными камерами, используются для решения таких задач, как измерение габаритов и подсчет транспортных средств или изделий, вычисление их веса или объема, а также визуальный контроль предварительно определенных состояний объекта продукции на максимальной скорости. Кроме того они обеспечивают автоматическое извлечение небольших фрагментов ключевой информации из огромных объемов данных и помогают экспертам интерпретировать изображения благодаря средствам фильтрации, оптимизации и дополнения данных, а также их быстрого поиска и извлечения.

Хотя машинное зрение — это процесс применения компьютерного зрения для промышленного применения, полезно перечислить часто используемые аппаратные и программные компоненты. Типовое решение системы машинного зрения включает в себя следующие компоненты: одна или несколько цифровых или аналоговых камер (черно-белые или цветные) с подходящей оптикой для получения изображений; программное обеспечение для обработки изображений (для аналоговых камер – это оцифровщик изображений); процессор (современный ПК с многоядерным процессором или встроенный процессор, например – ЦСП); программное обеспечение машинного

зрения, которое предоставляет инструменты для разработки отдельных приложений; оборудование ввода-вывода или каналы связи; «умная» камера – одно устройство, которое включает в себя все вышеперечисленные пункты; очень специализированные источники света (светодиоды, люминесцентные и галогенные лампы и т. д.); специфичные приложения программного обеспечения для обработки изображений и обнаружения соответствующих свойств; датчик для синхронизации частей обнаружения (часто оптический или магнитный датчик) для захвата и обработки изображений; приводы определенной формы, используемые для сортировки или отбрасывания части информации.

Применение машинного зрения разнообразно, оно охватывает различные области деятельности, включая, но не ограничиваясь следующими: крупное промышленное производство; ускоренное производство уникальных продуктов; системы безопасности в промышленных условиях; контроль предварительно изготовленных объектов (например, контроль качества, исследование допущенных ошибок); системы визуального контроля и управления (учет, считывание штрих-кодов); контроль автоматизированных транспортных средств; контроль качества и инспекция продуктов питания; в автомобильной промышленности системы машинного зрения используются в качестве руководства для промышленных роботов, а также для проверки поверхности окрашенного автомобиля, сварных швов, блоков цилиндров и многих других компонентов на наличие дефектов.

1.2 Особенности контроля для различных предметных областей

1.2.1 Контроль безопасности выполнения работ на электроустановках

Работа в действующих электроустановках требует от обслуживающего персонала выполнения правил безопасности [58]. В данных Правилах изложены основные требования безопасности при эксплуатации электроустано-

вок. Требования правил распространяются на работников, обслуживающих действующие электроустановки потребителей и являются обязательными для всех потребителей и производителей электроэнергии независимо от их ведомственной принадлежности и форм собственности на средства производства. Среди основных вопросов, описанных в правилах, такие как: пункт 3 – организация безопасной эксплуатации электроустановок; пункт 4 – требования к работникам, осуществляющим оперативные обслуживание электроустановок; пункт 6 – организационные мероприятия, обеспечивающие безопасность работ в электроустановках, и так далее.

Особенно это актуально для электрических сетей и оборудования крупных промышленных комплексов, электростанций, торговых центров и тому подобное. Однако при обзоре литературы было выявлено, что существующие автоматизированные системы диспетчеров энергосистем существуют, но они направлены, в основном, на решение задач управления энергосистемой в целом. Так в [65] описаны разработанные с помощью методов искусственного интеллекта Советчики диспетчера для разных служб, в частности, Советчик по экономичному ведению режима, Советчик по оптимальным переключениям, Советчик диспетчера по ликвидации перегрузок и другие. Они используются для помощи в выполнении диспетчерами сложных задач управления энергосистемой: планирования, оперативного управления, моделирования, оптимизации в режиме реального времени, В [55] рассмотрены особенности режимных тренажеров диспетчера энергосистем, включающие моделирование длительных переходных процессов, возможность представить в модели большую часть оборудования реальной энергосистемы, наличие механизма исполнения сценариев. Программных систем для помощи диспетчеру в решении вопросов, связанных с безопасностью при эксплуатации электроустановок, не найдено.

Целью в данном случае является контроль и улучшение безопасности работы на электрооборудовании. Лицо, принимающее решение, – это диспетчер, который следит за техникой безопасности. Принятие решений – это

вид деятельности, целью которого является найти возможность замены существующей ситуации на желаемую [68]. Тип контроля по видам объекта – специальные проверки при необходимости проведения работ, то есть предварительный контроль, осуществляемый бригадиром, а затем диспетчером перед выходом бригады на работу; текущий контроль, если бригада уже работает, а в это время изменились условия, например, в помещении температура стала высокой.

Система контроля призвана повысить безопасность на предприятии путем предоставления рекомендаций диспетчеру. Используются простые альтернативы: разрешить или запретить выполнение работ на электрооборудовании работникам с некоторыми полномочиями при данных условиях. Критериями для принятия решений являются правила безопасности при работе с электрооборудованием. Однако в настоящее время правила присутствуют лишь в текстовом документе, и необходимо автоматизировать использование их при анализе ситуаций и выдачи консультаций диспетчеру.

1.2.2 Контроль при работе компьютерной сети

Мониторинг и контроль компьютерной сети – постоянное наблюдение за сетью с целью выявления отклонений от допустимых значений эксплуатационных показателей, позволяющий оценить ее состояние. По методологии этот вид мониторинга является комплексным, поскольку сочетает в себе черты и динамического мониторинга, так как изменения показателей сети анализируются в течение некоторого промежутка времени для выявления закономерностей, и сравнительного мониторинга – полученные результаты сравниваются с некоторыми эталонными значениями, означающими нормальное функционирование системы. Относительно классификации по объектам контроля используются специальные проверки, например, при требованиях пользователя (при медленной связи).

Контроль сети обычно осуществляется системным администратором. Возможно осуществление контроля в ручном режиме с применением утилит

ping, traceroute, ipconfig (аналоги этих утилит доступны в любой операционной системе, поддерживающей работу в сети). Но такой способ является крайне трудоемким и неэффективным, поэтому был разработан ряд автоматизированных средств контроля и мониторинга, которые могут собирать данные, представлять их в удобной и наглядной форме, а в некоторых случаях и анализировать.

Рассмотрим далее существующие автоматизированные системы контроля и мониторинга для сетей. На сегодняшний день разработано большое количество подобных инструментов. Наиболее распространенными и применяемыми из них являются RRDTool [18], Cacti [4, 11, 24], Zabbix [8, 16], Monit (M/Monit) [6], Nagios[14], Icinga [9].

Система контроля и мониторинга сети способна выполнять надзор за сетью в поисках проблем, вызванных перегруженными и неисправными серверами, устройствами коммутации и каналами связи. Например, для определения состояния веб-программа, осуществляющая мониторинг, периодически выполняет HTTP-запрос для получения некоторой страницы, размещенной на нем, а для определения состояния почтового сервера может выполняться отправка тестового сообщения по протоколу IMAP и его принятие через протоколы POP3 или IMAP.

Ситуация отказа, возникающая при этом (в случае, например, невозможности установления соединения, завершения операции через таймаут, невозможности доставки сообщения) вызывает следующие реакции: отправляется сигнал тревоги системному администратору либо автоматически активируется система защиты от сбоев (в случае ее наличия), временно выводит проблемный узел сети из эксплуатации, направляя нагрузку на другие узлы.

Сравнительная характеристика систем контроля компьютерных сетей приведена в таблице 1.2. Сравнивая рассмотренные системы контроля и мониторинга, стоит отметить, что все современные продукты, разработка и использование которых еще продолжается, предоставляют широкие возможности наблюдения за узлами сети по различным протоколам (SNMP, ICMP,

FTP, HTTP, SMTP, POP3 и другими) и различными способами (путем передачи простых запросов, с помощью агентов, устанавливаемых на узлы), указывая при этом исчерпывающую информацию о текущих показателях сети и отдельных ее узлов (объем сетевого трафика, время обработки запросов, загрузки центрального процессора, оперативной памяти, дисковой подсистемы и так далее).

Таблица 1.2 – Сравнительная характеристика систем контроля и мониторинга компьютерных сетей

Система	RRDtool	Cacti	Zabbix	M/Monit	Nagios	Ichinga	NetXMS	Разработанная система
Возможность использования для обучения	нет	нет	нет	нет	да	нет	нет	да
Выявление неполадок	нет	нет	да	да	нет	да	да	да
Анализ	параметров по отдельности	параметров по отдельности	параметров по отдельности	параметров по отдельности	параметров по отдельности	параметров по отдельности	параметров по отдельности	комплексный
Добавление собственных правил пользователем	нет	нет	нет	нет	нет	нет	нет	да

Также большинством систем контроля и мониторинга предоставляются возможности визуализации полученных данных, уведомления системного администратора об определенных ситуациях, построения карты сети и дру-

гие. К сожалению, лишь немногие системы контроля и мониторинга способны анализировать собранные данные и идентифицировать на основе полученных результатов ряд неполадок, которые могут возникать в сети. При этом возможности анализа ограничены, поскольку не позволяют идентифицировать все возможные неполадки, а также не предоставляют пользователям возможности создания собственных правил.

Возможность автоматизированного обнаружения неполадок сети [19] является полезной, поскольку позволяет снизить нагрузку на системного администратора, также повысить вероятность раннего обнаружения и исправления. Возможность определения собственных правил при этом позволяет улучшить качество такой автоматизации, а в некоторых случаях просто необходима, особенно если в сети применяется нестандартное аппаратное обеспечение или архитектурное решение. Таким образом, целесообразно применение в системах контроля и мониторинга компьютерных сетей развитой подсистемы анализа, которая может изменяться пользователем.

Интерес представляет современная программно-инструментальная среда (СЛМ-технология) [85], представляющая собой визуальное инструментальное средство построения систем, основанных на знаниях в виде ЛВС-сети. Примечательно, что это средство ориентировано на работу эксперта со знаниями при вводе, редактировании, структуризации и проверки разрабатываемой сети.

1.3 Знаниеориентированные системы для анализа и принятия решений по результатам контроля

1.3.1 Знаниеориентированные системы, их особенности и типы

К главным недостаткам традиционных автоматизированных систем относятся слабая адаптируемость к информационным потребностям пользователя, а также невозможность решать плохо формализуемые задачи, которые характеризуются и качественным, и количественным описанием.

Перечисленные недостатки устраняются при использовании интеллектуальных систем, основанных на знаниях и имеющих следующие характерные признаки: развитые коммуникативные способности; умение решать сложные плохо формализуемые задачи, способность к развитию и самообучению.

Знаниеориентированная система – это интеллектуальная компьютерная программа, содержащая знания и аналитические способности одного или нескольких экспертов в отношении некоторой области применения и обладающая способностью делать логические выводы на основе этих знаний, тем самым обеспечивая решения специфических задач (консультирование, обучение, диагностика, тестирование, проектирование и т.д.) без привлечения эксперта (специалиста в конкретной предметной области) [40].

Знаниеориентированные системы предназначены для решения задач, для которых требуется проведение экспертизы человеком-специалистом экстра – класса [1]. В отличие от программ, использующих процедурный анализ, экспертные системы решают проблемы в узкой предметной области (конкретном участке экспертизы) на основе логических рассуждений. Такие системы часто могут найти решение задач, которые неструктурированные и неточно определены. Они с помощью эвристик компенсируют отсутствие структурированности, полезны в ситуациях, когда недостаточное количество необходимых данных или времени исключает возможность проведения полного анализа.

Типичные задачи, решаемые знаниеориентированными системами, следующие [40]: извлечение информации из первичных данных; диагностика неисправностей (как в технических системах, так и в человеческом организме); структурный анализ сложных объектов; выбор конфигурации сложных многокомпонентных систем; планирование последовательности выполнения операций, приводящих к заданной цели, и другие.

Применение знаниеориентированных систем дает следующие преимущества по сравнению с человеком-экспертом: повышенная доступность зна-

ний; возможность получения экспертных знаний из многих источников; возможность объяснения полученного заключения.

Знаниеориентированные системы разделяют на следующие классы:

– Системы с интеллектуальным интерфейсом, которые обладают коммуникативными способностями. Это интеллектуальные БД, естественно-языковой интерфейс, гипертекстовые системы, контекстные системы, когнитивная графика.

– Экспертные системы, предназначенные для решения сложных задач. Это классифицирующие, доопределяющие, трансформирующие и многоагентные системы.

– Самообучающиеся системы. К ним относятся: индуктивные системы, нейронные сети, системы, основанные на прецедентах и другие.

Основой знаниеориентированной системы является совокупность знаний, которая структурируется для упрощения процесса принятия решения. Знание [32] – это целостная и систематизированная совокупность понятий о закономерностях природы, общества и мышления, накопленная человечеством в процессе активной преобразующей деятельности и направленная на дальнейшее познание и изменение объективного мира.

База знаний содержит структурированную информацию, покрывающую некоторую предметную область, и предназначена для использования человеком или машиной с конкретной целью.

Модели представления знаний и вывод на знаниях. В настоящее время разработаны десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам: продукционные модели; семантические сети; фреймы; формальные логические модели. В свою очередь это множество классов можно разбить на две большие группы: модульные и сетевые. Модульные языки оперируют отдельными (не связанными) элементами знаний, будь то правила-продукции или аксиомы в формально-логических моделях. Сетевые

языки предоставляют возможность связывать эти элементы или фрагменты знаний через отношения в семантические сети или сети фреймов.

Рассмотрим подробнее наиболее популярные у разработчиков языки представления знаний (ЯПЗ).

Продукционная модель. ЯПЗ, основанные на правилах (rule-based), являются наиболее распространенными, и более 80% ЭС используют именно их. Продукционная модель, или модель, основанная на правилах, позволяет представить знания в виде предложений типа "Если (условие), то (действие)".

Семантические сети. Термин *"семантическая" означает "смысловая*", а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т. е. наука, определяющая смысл знаков. Модель на основе семантических сетей была предложена американским психологом Куиллианом. Основным ее преимуществом является то, что она более других соответствует современным представлениям об организации долговременной памяти человека. *Семантическая сеть* – это ориентированный граф, вершинами которого являются понятия, а дуги – отношения между ними. В качестве понятий обычно выступают абстрактные или конкретные объекты, а *отношения* – это связи типа: "это" ("АКО — A-Kind-Of, "is", или "элемент класса"), "имеет частью" ("has part"), "принадлежит", "любит" и другие.

Фреймы. Термин *фрейм* (от англ. *frame* – "каркас" или "рамка") был предложен Марвином Минским, одним из пионеров ИИ, в 70-е годы для обозначения структуры знаний для восприятия пространственных сцен. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование. Фрейм – это абстрактный образ для представления стереотипа объекта, понятия или ситуации.

Формальные логические модели. Традиционно в представлении знаний выделяют *формальные логические модели, основанные на классическом* исчислении предикатов 1-го порядка, когда предметная область или задача описывается в виде набора аксиом. Эта логическая модель применима в ос-

новном в исследовательских системах, так как предъявляет очень высокие требования и ограничения к предметной области. В промышленных же экспертных системах используются различные ее модификации и расширения [28].

Знаниеориентированные системы не только хранят знания человека, но и способны обрабатывать их как для получения новых знаний [36], так и для использования существующих при решении определенных задач. В любом случае при обработке используется так называемый логический вывод, а компонент, осуществляющий собственно вывод, а также управление этим процессом называется *машиной вывода,* Таким образом, можно сказать, что машина вывода – это программа, имитирующая логический вывод эксперта, пользующегося данной базой знаний для интерпретации поступивших в систему данных.

Вывод существенно зависит от модели представления знаний.

При *продукционной модели* машина вывода работает циклически и выполняет две функции.

Первая функция – собственно, *вывод*, то есть просмотр существующих данных (фактов) из рабочей памяти (базы данных) и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов. Таким образом, правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.

Вторая функция - *управление выводом*, то есть определение порядка просмотра и применения правил, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных. От выбранного метода поиска, т. е. стратегии вывода, зависит порядок применения и срабатывания правил.

Процедура выбора сводится к определению *направления поиска* и *способа* его осуществления (*стратегии*).

Направление поиска может быть обратным и прямым. При *обратном* порядке вывода сначала выдвигается некоторая гипотеза, а затем механизм вывода возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу. Ес6ли она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчиненной гипотезы. В системах с *прямым выводом* по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удается найти, то оно заносится в рабочую память.

Стратегии управления выводом позволяют минимизировать время поиска решения и тем самым повысить эффективность вывода. К числу таких стратегий относятся: поиск в глубину, поиск в ширину, разбиение на подзадачи и альфа-бета-алгоритм.

Вывод для *семантической сети* – это сопоставление некоторой подсети, отражающей поставленный запрос к базе, с фрагментом общей сети предметной области. В настоящее время популярны представления знаний в виде онтологий – это семантические сети для описания web, так называемый, семантический web.

Вывод на *фреймах* состоит в поиске значения свойства для некоторого объекта (с учетом наследования).

Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации.

В [85] предложена новая современная модель представления знаний – логико-вычислительная семантическая сеть (ЛВС-сеть), устраняющая многие проблемы, присущие традиционным моделям, в частности, связанные с автоматизацией получения, структурирования и проверки знаний. Для ЛВС-сети предложена совокупность методов обработки знаний, в том числе: ме-

тод прямого вывода, метод построения начального состояния процесса прямого вывода, методы верификации и тестирования.

Таким образом, существуют разнообразные модели представления знаний и способы логического вывода и из рассмотренных вариантов далее необходимо выбрать и, при необходимости, усовершенствовать те, которе в наибольшей степени применимы для задачи контроля.

1.3.2 Контроль и знаниеориентированные системы

Современные требования к уровню безопасности с целью безаварийного функционирования объектов обусловливают необходимость совершенствования процесса автоматизированного контроля, следовательно, к функциям контроля обычно относятся, кроме непосредственного наблюдения, еще и анализ параметров объекта для определения его состояния, который представляет собой нормальное функционирование или аварию, и, при необходимости, предоставление обработанной информации лицу, принимающему решение.

Для принятия решений о состоянии объектов применяются правила, которые обычно встроены в систему контроля. Однако условия контроля для большинства систем меняются довольно часто, и для облегчения сопровождения таких систем возникает необходимость в настройке правил, а адаптация традиционных систем требует значительных затрат, что приводит к снижению оперативности внесения изменений и/или сокращение достоверности решений. Кроме того, во многих системах контроля каждый параметр оценивается отдельно без связи с другими, и это негативно влияет на достоверность принятия решений по результатам контроля.

Таким образом, для комплексного анализа объектов с возможностью оперативно менять правила контроля предлагается использовать знаниеориентированных системы, потому что задача контроля не является полностью формализованной, условия контроля обычно описаны декларативно и при автоматизации требуют накопленного опыта экспертов. Выбрана модель пред-

ставления знаний в виде правил потому, что, во-первых, по своей сути такие знания определяют связь между состояниями контролируемого объекта и условиями, при которых эти состояния могут достигаться, и имеют следующий вид: "Если условия, то состояние объекта", а во-вторых, такая модель выгодно отличается низким уровнем сложности, универсальностью, высокой модульностью, небольшим объемом при хранении в памяти, близостью к естественному языку.

Привлечение технологий знаниеориентированных систем к построению систем контроля в различных предметных областях представляется целесообразным, так как они зарекомендовали себя на практике как эффективное средство поддержки процесса принятия решений с точки зрения накопления опытных знаний экспертов и предоставления их менее квалифицированным специалистам. Ранее системы контроля создавались для решения узких классов задач, вытекающих, как правило, из конкретных аварийных ситуаций, и их развитие носило характер модификации или модернизации, и в большинстве случаев такие системы узкоспециализированы, что снижало эффективность их использования. Поэтому разработка подхода к созданию систем контроля на основе «гибких технологий» таких, как знаниеориентированные системы, позволит расширить область охвата решаемых задач и повысить качество управления за счет привлечения экспертных знаний.

Ниже представлены описанные в литературе подходы к созданию знаниеориентированных систем технического контроля, оценивающие качество, надежность, а также состояния устройств.

В статье [84] освещаются вопросы выбора вариантов реализации знаниеориентированных систем контроля, встраиваемых в сеть передачи данных для улучшения качества обслуживания. Рассматривается обобщенная структура сети передачи данных, содержащая основные элементы знаниеориентированных систем контроля. Анализируются проблемы взаимодействия знаниеориентированных систем контроля с сетью передачи данных.

При создании экспертной системы контроля и оценки состояния и условий эксплуатации воздушных линий (ВЛ) электропередач [56] на этапе структуризации было проведено расчленение общей проблемы надежности воздушных линий на четырнадцать основных направлений, включающих в себя анализ полного спектра условий их эксплуатации и ремонта. Каждое из основных направлений предусматривает контроль, оценку качества и достаточности технической документации, а так же инструкций для соответствующего персонала, уровня исполнения регламентов, что позволило обеспечить методологический подход к выявлению и анализу причин повреждения ВЛ и перейти к формированию концепции повышения их надежности.

Знаниеориентированная система для контроля за текущим техническим состоянием рельсовых цепей [62] является Web-приложением и позволяет получать и оценивать данные о работоспособности рельсовых цепей, а также давать рекомендации по повышению надежности их работы.

Разработана знаниеориентированная система мониторинга, диагностики и управления для контроля технического состояния трансформаторного оборудования [82].

В [81] представлены подходы, реализующие контроль динамического состояния станков по виброакустическим колебаниям. Представлен образец знаниеориентированной системы контроля динамического состояния шлифовального станка.

В [50] описана знаниеориентированная система для автоматизации контроля температуры в стекловаренной печи с применением собственных радиотепловых сигналов от расплава стекломассы и алгоритм оптимального управления процессом. За счет применения четырехканального метода измерения мощности радиотепловых сигналов с использованием приемных СВЧ-антенн обеспечена точность контроля температуры стекломассы в пределах нескольких градусов.

Знаниеориентированная система используются также при обучении для контроля знаний. В статье [43] рассмотрен вариант реализации балльно-

рейтинговой системы оценивания ключевых компетенций. Одним из основных видов такого контроля является тестирование, экспертная система RExpert позволяет добиться максимальной объективности при минимальных затратах времени.

Таким образом, в ходе исследования открытых печатных источников, посвященных данному вопросу, был сделан вывод о том, что существуют системы контроля, реализующие анализ информации для различных предметных областей, однако не был найден общепризнанный и проверенный на практике общий подход к построению систем контроля со встроенной компонентой экспертного анализа.

Далее в работе предложенные модели и методы построения знаниеориентированных систем контроля рассматриваются на примере двух предметных областей: контроль безопасности работы с электроустановками и контроль, и мониторинг работы компьютерной сети. Рассмотрим целесообразность применения знаний в этих системах контроля.

1.3.3 Целесообразность использования знаниеориентированных систем при контроле

Перед тем, как перейти к разработке системы, необходимо рассмотреть вопрос о целесообразности применения знаниеориентированной системы для решения поставленной задачи. Для того, чтобы применение знаниеориентированной системы было возможно, нужно, чтобы выполнялись следующие требования [67]: в данной предметной области существуют эксперты; эксперты пришли к согласию в оценке возможности решения поставленной задачи; эксперты способны вербализовать и объяснить используемые для решения задач подходы и действия; в предметной области могут быть выделены основные понятия, отношения между ними и методы решения поставленной задач.

Использование знаниеориентированных системы может быть возможным, но неоправданным. Для оправдания ее применения требуется выполне-

ние одного из следующих условий: применение знаниеориентированной системы имеет значительный эффект (например, экономический); привлечение человека–эксперта при решении задач невозможно из-за недостаточного их количества; применение знаниеориентированной системы позволяет предотвратить недопустимые потери времени и информации при коммуникации с экспертом; существует необходимость решения задачи в окружении, опасном для человека.

1.3.3.1 Целесообразность использования знаниеориентированных систем для контроля безопасности работы с электроустановками

Работа в действующих электроустановках требует от обслуживающего персонала выполнения Правил безопасности, которые в настоящее время хранятся в текстовом документе, в том числе и в электронном виде. И диспетчер, отвечающий за безопасность выполнения работ, при допуске бригад к работе и в процессе работы руководствуется Правилами, то есть читает этот документ.

Однако знания, хранящиеся в Правилах, можно записать не просто в тексте, а так, чтобы на их основе автоматизированная система могла сама делать выводы и помогать принимать решения. В наибольшей степени для контроля безопасности выполнения работ на электроустановках подходят знаниеориентированные системы, позволяющие с помощью знаний описать и сам объект, и ситуации, возникающие в объекте, для которой производится контроль.

1.3.3.2 Целесообразность использования знаниеориентированных систем для задачи контроля и мониторинга сети

Состояние компьютерной сети зависит от большого количества различных факторов и их сочетаний. В таких условиях выявление причины возникающих проблем требует не только наличия необходимых знаний и квалификации, но и определенных специфических навыков, умения быстро и безошибочно оценить любую ситуацию, которая может возникнуть. Приобрете-

ние необходимых знаний, навыков и развитие их до нужного уровня требует значительного времени, поэтому остро ощущается нехватка специалистов с необходимым уровнем квалификации.

Другим важным фактором является время обнаружения и исправления неисправности. Возникновение неполадок в компьютерной сети могут приводить к снижению уровня функционирования (вплоть до полного отказа в обслуживании) через недоступность тех или иных ресурсов, что сказывается на функционировании зависимых от них программ и сервисов. По этой причине важно минимизировать время недоступности сетевых ресурсов, то есть времени на диагностирование причины и исправления неисправности обычно мало.

Таким образом, применение экспертных систем для контроля и мониторинга компьютерных сетей позволяет достичь значительного экономического эффекта путем минимизации затрат на обслуживающий персонал и потерь, вызванных пониженным уровнем функционирования или недоступностью сетевых ресурсов. Кроме этого, применение знаниеориентированных систем дает возможность использования опыта лучших специалистов без необходимости их непосредственного привлечения, а также снижение потерь времени при обнаружении и исправлении неполадок, что позволяет говорить о его целесообразности.

1.4 Автоматизация создания и поддержки баз правил

1.4.1 Структура знаний для контроля

Для получения знаний инженер по знаниям должен реализовать три этапа: получение знаний, структурирование знаний и представление знаний. *Получение знаний* с применение компьютера может быть реализовано с помощью специальных инструментальных средств, помогающих эксперту и инженеру по знаниям в работе, а также на основе примеров при наличии репрезентативной (то есть достаточно представительной) выборки примеров при-

нятия решений в предметной области. *Структурирование (или концептуализация) знаний* [35] – разработка неформального наглядного описания знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области.

В качестве модели представления знаний наиболее целесообразной представляется модель фактов и правил, так как правила естественным образом могут показать, в каком случае возникает то или иное состояние объекта. Например, производится контроль безопасности выполнения работ на электроустановках. Если количество работников в бригаде меньше 2 и тип работ переключение и помещение особо опасное, то работы выполнять нельзя. Состояние объекта здесь – «работы выполнять нельзя». Таким образом, правило в выводах содержат состояние объекта, а в посылках – условия, при которых возникает это состояние. В общем случае, посылки – это состояния некоторых сущностей. Например, сущность – «степень опасности помещения», а состояния этой сущности: «помещение неопасное», «помещение с повышенной опасностью», «помещение особо опасное».

1.4.2 Словарь предметной области

Одним из начальных шагов при создании любой автоматизированной системы является изучение предметной области и, в частности, терминов, наиболее часто встречающихся при составлении текстов и формирующих так называемый *словарь предметной области* [53]. Термин – это слово, устойчивое словосочетание или сокращение, которое выражает и в известной мере классифицирует в данной предметной области определённое понятие или сущность, отражая в своей смысловой структуре характеристические признаки объекта и взаимосвязи этого объекта с другими. При разработке знаниеориентированных систем словарь терминов незаменим при построении знаний, и, прежде всего, поля знаний; так как термины сами по себе являются знаниями о предметной области, словарь позволяет напомнить экс-

перту о терминах, а инженеру по знаниям более целостно представить общую картину предметной области.

В настоящее время существуют методы и системы для автоматического и/или автоматизированного создания словарей [49].

При *создании словаря* значимы следующие *этапы*:

– подбор текстов для анализа;

– выделение терминов;

– обработка результатов выделения терминов [60];

– конечное построение словаря предметной области.

Процесс построения словаря предметной области состоит в последовательной обработке некоторого множества текстов и выделения из него слов и именных групп, являющихся терминами [7]. При последовательной обработке текстов из заданного множества на каждой итерации обрабатывается очередной текст. Для каждого текста создаются выделенные из этого текста слова и именные группы, и они включаются в список слов и именных групп общего словаря. Таким образом, после обработки каждого текста словарь предметной области увеличивается. Процесс его формирования останавливается после обработки очередного текста, в результате которой прирост количества терминов не произойдет.

1.4.3 Графическое отображение знаний и интерактивная обработка

В процессе построения поля знаний необходимо отображать результаты в наглядной форме, в наибольшей степени для этого подходят графики, диаграммы, то есть графическое представление. Для разных типов знаниеориентированных систем подходят разные представления, к примеру, для *диагностических* наиболее приемлемо дерево, в корне которого находится компонент системы, который диагностируется (см. рис. 1.1), в примере – это двигатель автомобиля. На нижних уровнях дерева находятся части диагностируемого компонента (в примере – это наличие топлива, состояние аккумулято-

ров, проблемы в системе зажигания), а в листьях дерева располагаются способы решения проблем (в примере они обозначены цифрами).

Необходимо, аналогично задаче диагностики, предложить способы графической визуализации и интерактивной работы с правилами контроля. Такой подход позволит упростить разработку систем и возможность оперативного внесения изменений [35].

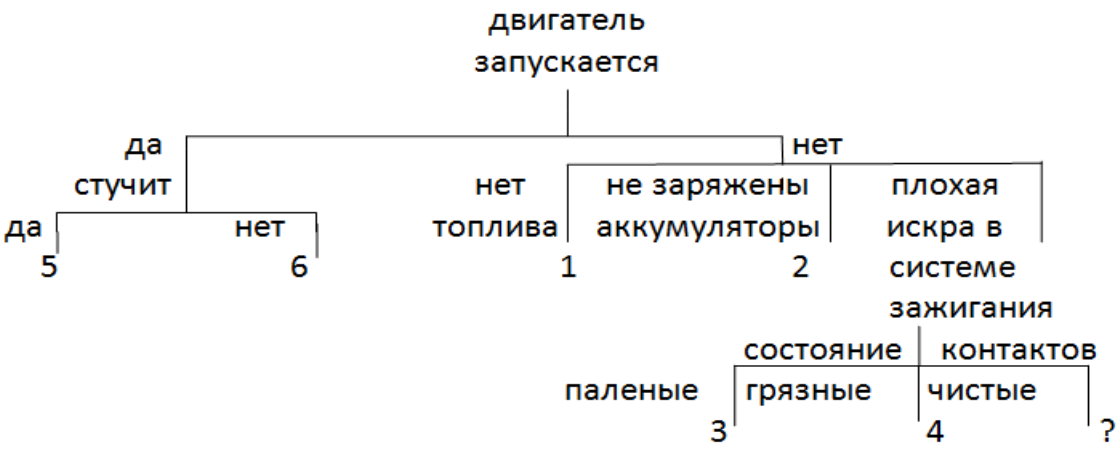

Рисунок 1.1 – Дерево решений для диагностики неисправностей двигателя

1.4.4 Тестирование и проверка качества баз правил

Вопросы обеспечения надежности программного обеспечения (ПО) относятся к области компетенции специального направления исследований, получившего название *валидации и верификации* (V&V) [64].

Вследствие необходимости обеспечения надежности знаниеориентированных систем, данные исследования затронули и проблемы валидации и верификации знаниеориентированных систем. Таким образом, если при разработке обычного ПО валидация направлена на то, чтобы показать, что система *правильна* относительно предъявляемых к ней требований, то при построении знаниеориентированных систем лучшее, чего можно добиться – это показать, что система *достаточно компетентна*.

В рамках решения проблем валидации и верификации продукционных знаниеориентированных систем выделились следующие доминирующие

подходы [85]: осмотр, статическая верификация, эмпирическое тестирование, эмпирическая оценка.

Два подхода из перечисленных (*осмотр* и *эмпирическая оценка*) основываются на привлечении разработчиков знаниеориентированных систем для выполнения валидации и верификации. Так описанный в [13] *осмотр* нацелен на изучение БЗ знаниеориентированных систем экспертами и обнаружение ими семантически некорректных знаний, а *эмпирическая оценка* – на оценку разработчиками эксплуатационных показателей работы систем в целом. Таким образом, данные подходы основываются на «ручных», а не на автоматизированных процедурах выполнения валидации и верификации.

Другие два подхода (*статическая верификация* и *эмпирическое тестирование*) ориентированы на автоматизацию валидации и верификации систем. Методы, разрабатываемые в рамках данных подходов, принципиально разнятся по характеру использования механизма логического вывода. Так, методы статической верификации, в отличие от методов эмпирического тестирования, не опираются на использование механизма логического вывода.

Отметим, что проверки БЗ наиболее эффективны на ранних этапах, когда создается описание основных понятий и взаимосвязей ПрО на естественном языке, потому что на этом этапе со знаниями может работать эксперт. Поэтому далее остановимся на методах статической верификации.

При статической верификации обнаруживаются так называемые аномалии – статический образец в БЗ, который предполагает присутствие ошибки в представленных знаниях. Типы аномалий: избыточность, противоречивость, зацикливание, неполнота.

На сегодняшний день наибольшую популярность получила классификация методов статической верификации для модели представления знаний виде правил, предложенная А.Присом (A.Preece) [17]:

– *Методы, основанные на таблицах решений* – каждая строка формализует отношения, существующие между посылкой и выводом каждого правила. Это наглядней, чем правила в текстовом виде, и они могут показать толь-

ко аномалии, но для дальнейшего анализа используются метазнания, предварительно задаваемые экспертом, и механизмы логического вывода. Существуют и расширенные таблицы решений [64] для обнаружения статических и динамических аномалий в знаниях, предусматривающие совместную обработку неопределенных, неточных и нечетких знаний для обнаружения аномалий нарушений целостности и статической несогласованности.

– *Методы, основанные на генерации меток*. Предполагают построение из базы правил аналитического представления всех возможных цепочек вывода с формированием множества окружений (меток), приводящих к выводимости релевантных целей, то есть метки – это ограничения, задаваемые пользователем для проверки аномалий.

– *Методы, основанные на графах*. Преобразование правил в графовое представление, которое показывает отношения между посылками и выводами правил, а также между правилами. Используются ориентированные графы, проверка аномалий производится на основе заранее введенных пользователем ограничений, Применяются и раскрашенные сети Петри, в которых переходы – это номера правил, а позиции – атрибуты. Проверяется достижимость конечных вершин из начальных. Для поиска противоречий используются заранее заданные ограничения.

Существуют и методы, объединяющие представленные выше подходы. Так в [44] используется модель в виде гиперграфа специального вида, объединяющего в себе все сущности ПрО, а также зависимости, заданные в виде ограничений.

Недостатками большинства существующих методов статической верификации правил является то, что они требуют от пользователя задания ограничений, а также предполагают необходимость привлечения эксперта к анализу выявленных аномалий. Таким образом, необходимо предложить усовершенствованные модели и методы статической верификации для модели

представления знаний в виде правил с учетом их применимости для задачи контроля.

1.5 Инструментальные средства для подготовки баз правил

Поскольку базы знаний могут иметь большой объем и достаточно сложную структуру, логичной и оправданной является наличие специальных средств для их создания. Для разработки баз знаний, представляемых в виде правил, существует специальный класс инструментов – системы управления бизнес-правилами (Business rule management system, BRMS). На сегодняшний день существует несколько подобных систем, которые успешно применяются в бизнес-ориентированных продуктах, логика которых представляет собой сложную систему правил. Существуют разнообразные формы представления знаний в подобных программах: системы продукционных правил, таблицы и деревья принятия решений.

Система управления бизнес-правилами (Business Rule Management System, BRMS) – программная система, предназначенная для определения, развертывания, исполнения, контроля и поддержки сложных правил и процедур принятия решений (эти правила и процедуры также называются бизнес-правилами). Несмотря на название, эти системы могут быть применены не только при разработке бизнес-программ, но и для других систем, где используются правила.

Однако практика *широкого* применения систем управления бизнес-правилами в системах общего назначения неизвестна. Причиной этого является то, что формы представления знаний, которые применяются в этих системах, не предназначены для систем в других предметных областях, в том числе и для систем контроля. Кроме того, методы проверки знаний, применяемые в имеющихся продуктах, не является полными и достаточными для проверки знаний в других предметных областях.

В таблице 1.3 приведена сравнительная характеристика рассмотренных средств разработки баз знаний.

Таблица 1.3 – Сравнение инструментов разработки баз знаний

	Jess	DTRules	OpenL Tablets	IBM JRules	JBoss Drools	Разрабатываемый редактор правил
1	2	3	4	5	6	7
Форма представления знаний	Правила и факты	Правила и факты	Таблица решений	Правила и факты	Правила та факты	Правила и факты
Инструменты редактирования базы знаний	Текстовый редактор	Текстовый редактор	Microsoft Word, Excel	Среда Eclipse, Microsoft Word, Excel	Среда Eclipse, текстовый редактор	Визуальный редактор для графа, текстовый редактор
Визуализация знаний	Нет	Нет	Нет	Да	Да	Да
Контроль качества знаний	Нет	Нет	Огранич.	Огранич.	Огранич.	Да
Проверка противоречивости знаний	Нет	Нет	Да	Да	Да	Да
Проверка полноты знаний	Нет	Нет	Нет	Да	Нет	Да
Проверка избыточности знаний	Нет	Нет	Нет	Нет	Нет	Да
Открытый исходный код	Нет	Да	Нет	Нет	Нет	Да

Таким образом, необходимо предложить в качестве редактора правил контроля визуальную среду, облегчающую для инженера по знаниям и эксперта работу с правилами и позволяющую представлять правила как на естественном языке, так и в графическом виде с возможностью предоставления условий для гибкой работы по вводу, редактирования и проверки правил.

1.6 Выводы по первому разделу

В данной главе проведен анализ существующих решений в области контроля объектов и процессов для различных ПрО, знаниеориентированных систем для анализа и принятия решений по результатам контроля, автомати-

зации создания и поддержки баз правил, инструментальных средств для подготовки баз правил.

Сделаны следующие основные выводы:

1. Рассмотрено понятие контроля объектов и процессов, проведена классификация видов контроля, описаны средства контроля и мониторинга, а также возможности их использования в системах контроля. Современные средства контроля могут быть применены в широком диапазоне областей, начиная от крупного производства, систем безопасности и кончая сферой обслуживания.

2. Рассмотрены существующие системы контроля безопасности выполнения работ на электроустановках. Критериями для принятия решений являются Правила безопасности при работе с электрооборудованием, однако в настоящее время правила присутствуют лишь в текстовых документах, и сделан вывод, что необходимо автоматизировать их использование при анализе ситуаций и выдаче консультаций диспетчеру. Тип контроля – предварительный контроль и специальные проверки при необходимости проведения работ с электроустановками, а также текущий контроль.

3. Проведен анализ существующих систем контроля и мониторинга сети, в связи с выявленными недостатками сделан вывод о том, что необходимы усовершенствованные методы анализа результатов объектов контроля, позволяющие проводить комплексный анализ параметров, а также дающие возможность оперативно менять правила контроля. Контроль и мониторинг компьютерной сети – это постоянные контроль и наблюдение за сетью с целью выявления отклонений от допустимых значений эксплуатационных показателей, позволяющий оценить ее состояние, а также специальные проверки, в частности, по требованию пользователя.

4. Для автоматизации анализа ситуаций и принятия решения по результатам контроля целесообразно использование знаниеориентированные систем, так как они зарекомендовали себя на практике как эффективное средство поддержки процесса принятия решений, и задача контроля является слабо

формализованной, условия контроля обычно описаны декларативно, и при автоматизации используется накопленный опыт экспертов.

6. Проведен анализ существующих экспертных систем контроля и мониторинга, различающихся по назначению, предметной области, методам представления знаний, динамичности и сложности. Сделан вывод, что каждая из них направлена на свою область, поэтому актуальной есть задача разработки ИТ, которая обобщает процессы создания и сопровождения знанняориентированных систем контроля для разных ПрО.

7. Показана целесообразность использования знаниеориентированных систем для двух вышерассмотренных областей. На основе проведенного анализа моделей представления знаний решено для контроля использовать правила-продукции. Предлагается в выводах правил описывать состояния объектов (нормальное состояние или аварийное), для которых проводится контроль, а в посылках – условия, при которых возникают эти состояния.

8. Отмечено, что «узким местом» при разработке знаниеориентированных систем является этап структурирования знаний для ПрО, поэтому актуальным является автоматизация этого процесса. При подготовке правил разработчики должны изучать и применять специальные языки для представления знаний, что предопределяет проблемы как при создании, так и, что важнее, при сопровождении систем. В связи с этим актуальной является разработка такой визуальной среды, в которой появляется возможность представлять ПрО с помощью естественного языка. Однако работа с правилами в текстовом виде затруднена при большом количестве правил, в частности, крайне тяжело отслеживать взаимосвязи между правилами. Поэтому необходимо предложить способы для графической визуализации и интерактивной работы с правилами контроля, такой подход приводит к облегчению разработки систем и возможности оперативного внесения изменений.

9. Актуальной является задача проверки правил, этими проблемами занимались некоторые отечественные и зарубежные исследователи. Поспеловым И.Г. предложенный способ поиска противоречий в правилах на этапе

тестирования БЗ с использованием машины вывода. Однако сделан вывод о том, что такого рода проверки наиболее эффективны на ранних этапах, когда создается описание основных понятий и взаимосвязей ПрО, так как на этом этапе со знаниями в состоянии работать эксперт.

Наиболее популярной является классификация А. Приса методов статической верификации для правил-продукций без применения машины вывода. Однако большинство существующих методов статической верификации правил требуют от пользователя задания ограничений, а также эти методы не выполняются автоматически и требуют необходимости привлечения эксперта к анализу выявленных аномалий, связанных с противоречиями и неполнотой БЗ. Для других моделей представления знаний, в частности, логико-вычислительной семантической сети (ЛОС-сети), которая разработана Яловцем А.Л., предложены методы поиска аномалий на основе структурного анализа фрагментов ЛОС, в которых эти недостатки частично решены. Поэтому и для правил контроля необходимо предложить усовершенствованные модели и методы статической верификации.

Результаты исследований раздела 1 опубликованы в работах автора [23, 69].

РАЗДЕЛ 2
РАЗРАБОТКА МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ ПРАВИЛ ДЛЯ ЗНАНИЕОРИЕНТИРОВАННЫХ СИСТЕМ КОНТРОЛЯ

2.1 Создание схемы знаниеориентированных систем контроля

Предлагается следующая структурно-функциональная схема системы контроля на основе знаний (рис. 2.1), состоящая их двух частей.

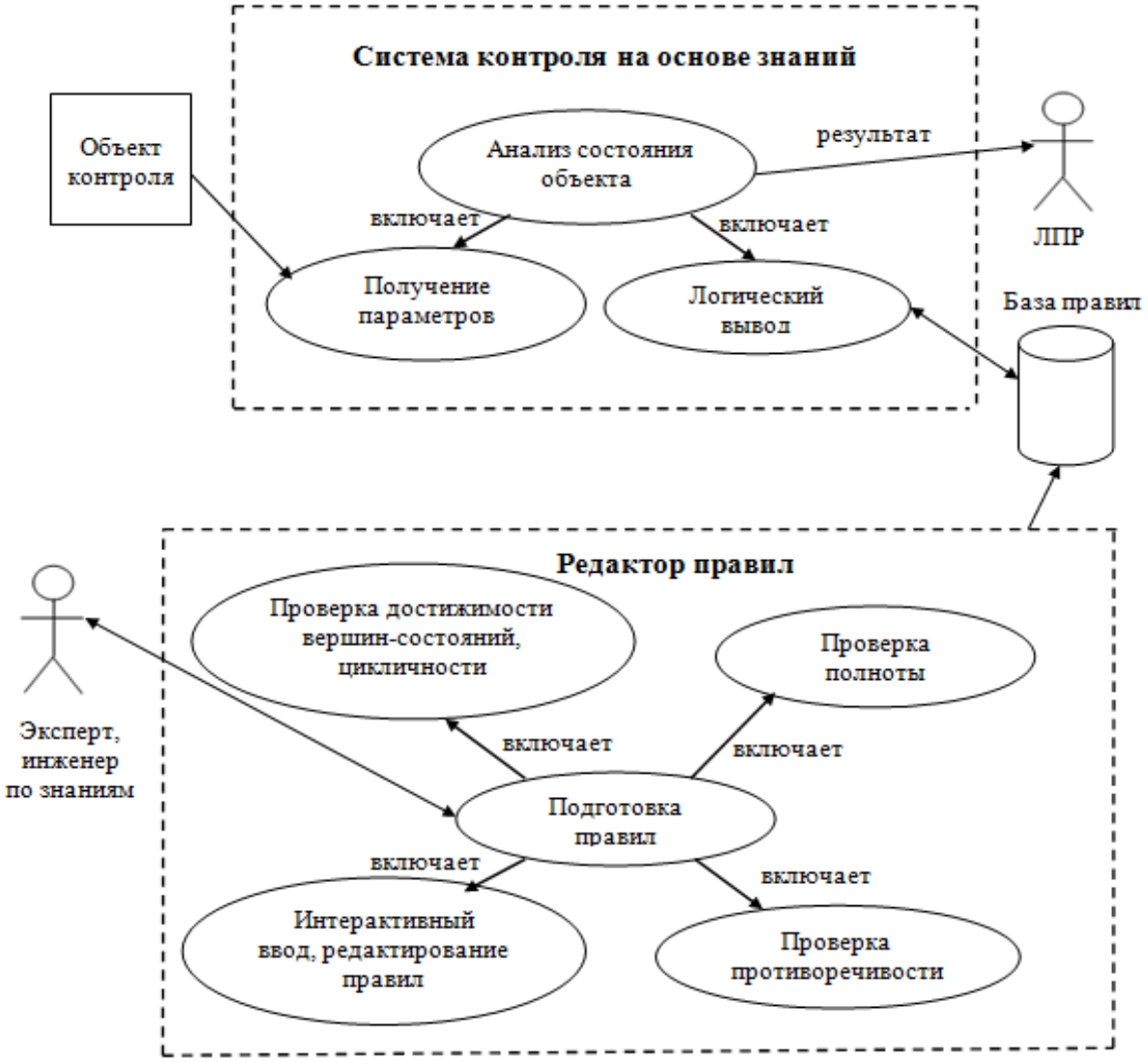

Рисунок 2.1 – Функциональная схема знаниеориентированной системы контроля

Инженер по знаниям и эксперт готовят правила для добавления в БЗ с помощью специального *редактора правил* контроля, выполняют эту работу интерактивно, при этом проводятся проверки правил на полноту, противоречивость, избыточность, достижимость вершин состояния, цикличность. В на-

чале разработки правила вводит инженер по знаниям вместе с экспертом, но одно из главных требований к редактору правил состоит в облегчении работы с правилами таким образом, чтобы изменение условий контроля на этапе сопровождения системы могло выполняться преимущественно экспертом.

Полученные параметры объекта анализируются с использованием машины вывода и базы правил, и результат передается ЛПР для принятия решений. Таким образом, *система контроля на основе знаний* состоит из следующих частей: подсистемы получения параметров от объекта контроля; подсистемы анализа параметров с использованием машины вывода и БЗ; подсистемы передачи результатов анализа ЛПР.

2.2 Особенности структурирования знаний для знаниеориентированных систем контроля

Обобщенно синтаксическую структуру знаний представляют как Pz = (I, O, M), где *I* – структура исходных данных, подлежащих обработке и интерпретации, *O* – структура выходных данных, то есть результата работы системы, *M* – операциональная модель предметной области, на основании которой происходит модификация *I* в *O*. Операциональная модель $M=(S_k,S_f)$ представляется как совокупность концептуальной структуры S_k,, отражающей понятийную структуру предметной области (статическая, неизменная составляющая знаний), и функциональной структуры S_f, моделирующей схему рассуждений эксперта (динамическая, изменяемая составляющая) [35].

Формирование S_k основано на выявлении понятийной структуры предметной области, обычно в виде иерархии понятий. Концептуальная структура включает понятия предметной области и моделирует основные функциональные связи или отношения между понятиями. S_f образует стратегическую составляющую *M,* часто она имеет форму таблицы решений [35]. Фрагменты S_k и S_f для предметной области по безопасной работе с электроустановками представлен на рис. 2.2 и таблице 2.1.

Отметим особенности знаний для предметных областей контроля.

Исходные данные I систем контроля – это параметры объекта, для которой производится контроль, и возможные значения параметров. Рассматриваемые параметры могут быть различными, то есть качественными, когда числовые параметры объектов поступают с датчиков и конвертируются в нечеткие параметры, или логическими. На рис. 2.2 параметры – это «ранг работ», «напряжение» и другие, а значения параметров, например, «по наряду», «по распоряжению».

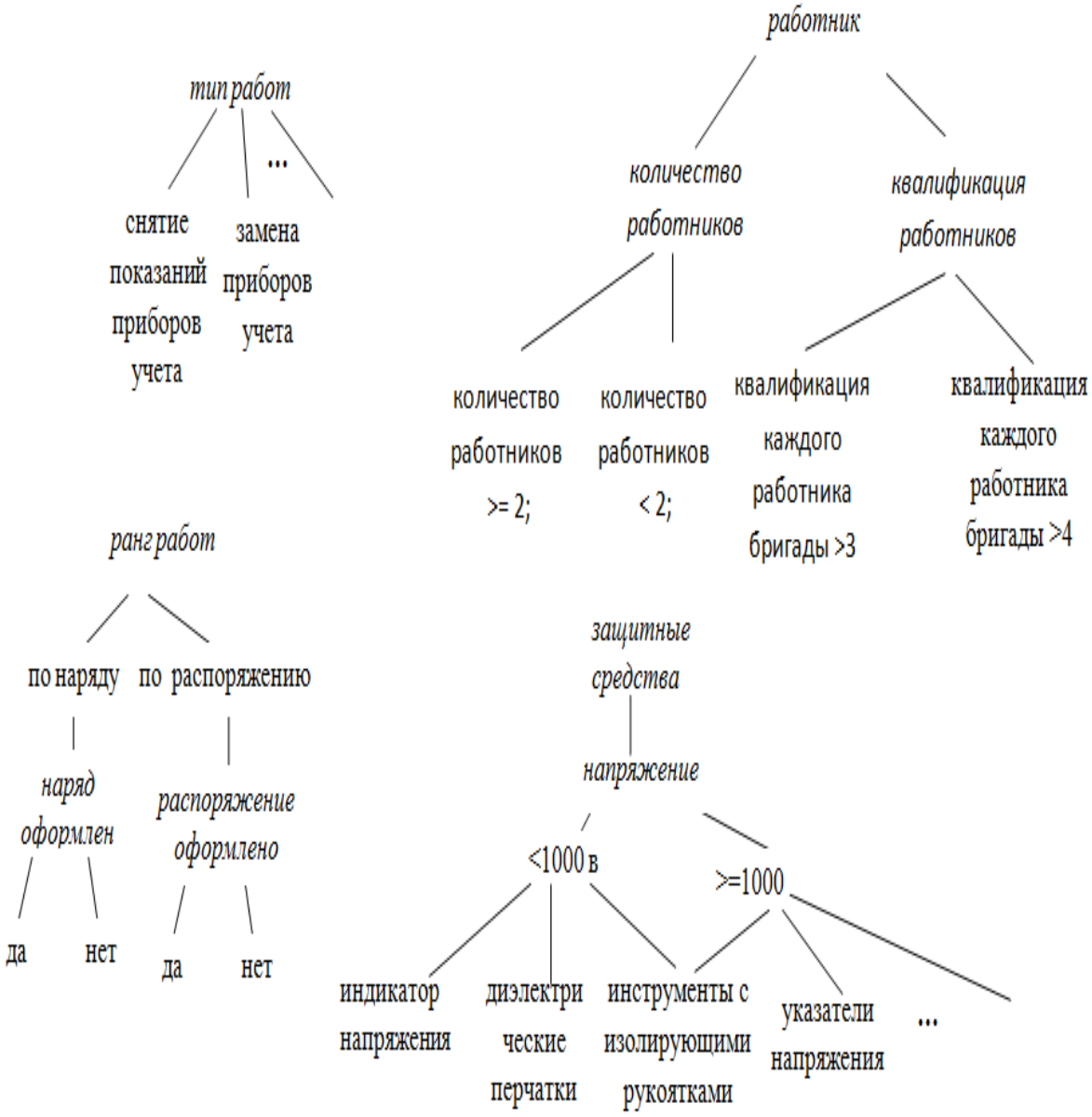

Рисунок 2.2 – Фрагмент концептуальной структуры знаний

Таблица 2.1 – Функциональная составляющая знаний

Параметр 1 ранг работ	Параметр 2 распоряжение оформлено	Параметр 3 наличие руководителя	Параметр 4 напряже-ние	Параметр 5 наряд оформлен	Заключение
по наряду				нет	работы выполнять нельзя
по наряду		нет	>=1000в		работы выполнять нельзя
в порядке тех-нической экс-плуатации	нет		>=1000в		работы выполнять нельзя
по распоря-жению	нет				работы выполнять нельзя
…	…	…	…	…	…

В качестве выходных данных О выступают заключения о выходе состояний объекта, контроль которых производится, за допустимые пределы, которые представляют собой аварии. Если отслеживаются несколько состояний объекта, то целесообразно разбить знания на группы, каждая из которых описывает одно состояние объекта. Если увелчивается объем групп, то целесообразно их разбить на подгруппы семантически близких для удобства работы. В таблице 2.1 рассмотрены параметры и их значения для одной из 5 разработанных подгрупп правил для знаниеориентированной системы контроля безопасной работы с электроустановками.

Иерархические связи понятийной структуры Sk позволяют разбить знания соответственно, выделив так называемые конечные (терминальные), из которых напрямую делается вывод о выходе значений отслеживаемых состояний объекта за допустимые пределы, и промежуточные, которые в качестве заключения содержат знания, с помощью которых в результате сцепления цепочек можно прийти к окончательному выводу. В таблице 2.1 показаны только терминальные правила контроля.

Таблица решений для контроля в каждой строке содержит значения параметров по всем или некоторым параметрам подгруппы, а в последнем столбце – заключение, говорящее о том, что произошел выход за пределы допустимого отслеживаемого состояния объекта.

Изучив отличительные особенности систем знаний для контроля, можно сделать вывод, о том, что построение знаний для этой задачи можно упростить за счет разбиения правил на группы, выбора наиболее приемлемой формы для визуализации знаний и интерактивной их обработки, создания методов проверки качества набора правил, ориентированных на контроль.

При увеличении объема групп для удобства работы и для возможности распределения процесса создания знаний между разными специалистами, правила предлагается разделить на семантически близкие подгруппы.

2.3 Разработка модели на основе И/ИЛИ-графа для построения, визуализации и интерактивной работы с правилами знаниеориентированных систем контроля

В качестве формы представления знаний для использования в системах контроля были выбраны правила – продукции. Такая форма представления знаний выгодно отличается от других близостью к естественному языку, однако отметим, что не всегда это означает легкость формулирования правил для использования в знаниеориентированных системах. Например, в нормативных документах правило может выглядеть следующим образом:

Если должны производиться погрузочно-разгрузочные работы в охранной зоне электрических сетей, то работы выполнять нельзя без согласования выполнения этих работ с организацией, эксплуатирующей электрические сети, и получения разрешения на их выполнение.

Для знаниеориентированной системы его нужно переформулировать так:

Если вид работ – строительные, И место проведения – в охранной зоне электрических сетей, И нет разрешения для выполнения работ, ТО работы выполнять нельзя.

При «запутанных» текстах такая переформулировка может быть произведена неверно.

Кроме того, проверка правил в текстовом виде затруднена при большом количестве правил. В частности, крайне трудно отслеживать взаимосвязи между правилами.

Для знаниеориентированных систем *контроля* удобно для каждого отслеживаемого состояния объекта строить граф, у которого одна «главная» вершина соответствует состоянию объекта, а остальные – это значения параметров объекта, и каждый путь представляет собой один набор условий, при которых отслеживаемое состояние объекта выходит за допустимые пределы (рис. 2.3).

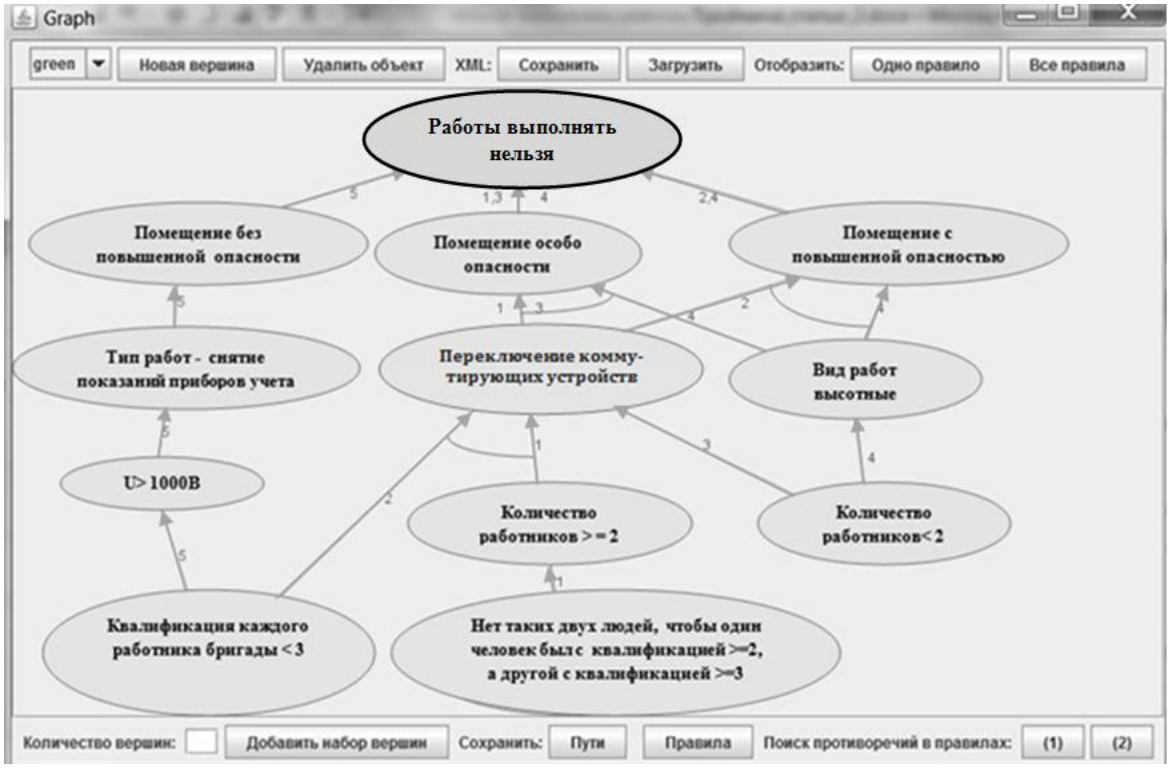

Рисунок 2.3 – Пример И/ИЛИ-графа для правил электробезопасности

Так как для систем контроля целесообразно использовать представление знаний в виде правил, а посылки правил связаны между собой И или ИЛИ связями, то для отображения правил контроля в графическом виде предлагается использовать такую известную структуру как И/ИЛИ-граф.

И/ИЛИ-граф – это ориентированный граф без циклов, все вершины графа разделены на три непересекающихся класса: «И»-вершина, «ИЛИ»-вершина, конечные (или целевые) вершины [83].

Решающий граф – это способ решение корневой задачи путем сведения ее к подзадаче (подцелям). Конечные вершины соответствуют элементарным подзадачам. Пространство решений определяется следующим образом: отдельное решение – это подграф И/ИЛИ-графа, который строится по правилам: в решение входит корень; если в решение входит «И»-вершина, то в нее включаются все его приемники и соответствующие дуги; если в решение входит «ИЛИ»-вершина, то в него включается лишь один из его приемников и соответствующая дуга.

Известно применение И/ИЛИ-графа как основы для логического вывода в системе MYCIN [40]. По завершении процесса диагностики выбирается рекомендуемый курс лечения.

Таким образом, решением выше перечисленных проблем является альтернативная текстовому модель представления правил в виде И/ИЛИ графа, представленная на рис. 2.4, 2.5 и 2.6. Вершинами этого графа являются посылки и выводы правил, а дуги задают отношения между ними, объединяя их в правила. Веса дуг графа соответствуют номерам правил.

Для улучшения дальнейшей обработки правил, а также в связи с использованием для контроля предложено усовершенствовать модель правил в виде ориентированного И/ИЛИ-графа добавлением специальной разметки графа, элементы которой перечислены ниже.

1) Вершинам, соответствующим посылкам правил, то есть возможным значениям входных параметров объекта, при необходимости указывают их свойства, а именно отмечаются два или более взаимно противоположных значений параметра (см. рис. 4.6, где обозначены группы вершин), а для нечетких параметров задаются нечеткие переменные с помощью нечетких множеств (см. рис. 2.10).

2) Помечаются вершины, соответствующие окончательным выводам правил, потому что в выводах всех правил одной группы находится одно и тот же аварийное состояние, поэтому на графе отображается одна вершина этого типа, и каждая группа правил отображается в виде отдельного графа.

3) Дуги имеют метки, которые являются номерами правил, это позволяет превратить пути в графе, которые соответствуют правилам, в булевы выражения для их применения в дальнейшем при проверке правил.

Предложено два способа визуализации И/ИЛИ-графа: для первого способа в соответствии с правилами соединяем все посылки каждого правила между собой и с выводом (рис. 2.4а, 4.9), и таким образом, предлагается использовать, если в группе правил присутствуют только терминальные правила, и все посылки связаны отношением И; второй способ: соединяем вершины-посылки для каждого правила с вершиной-выводом (рис. 2.4б, 4.6).

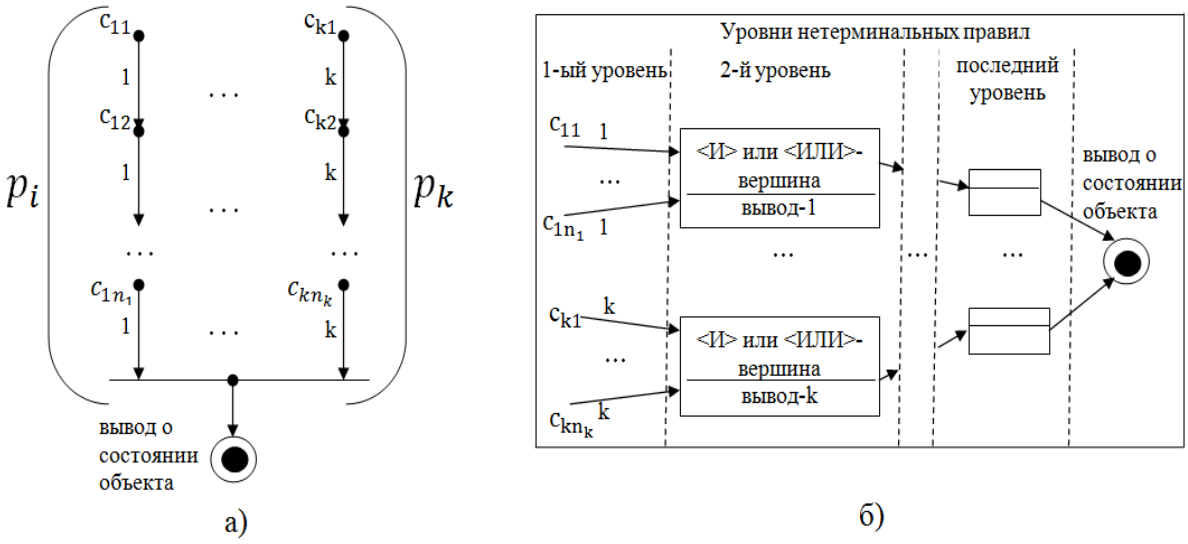

Рисунок 2.4 – Два способа визуализации И/ИЛИ-графа:
а) 1-ый способ визуализации; б) 2-й способ визуализации

При этом решается проблема недостатка наглядности, так как с помощью графа удобно прослеживать условия и вывод каждого правила, а также взаимосвязи в иерархии правил.

На рис. 2.5 показаны два правила, связанные между собой одним утверждением, которое является одновременно выводом правила 2 и условием правила 1.

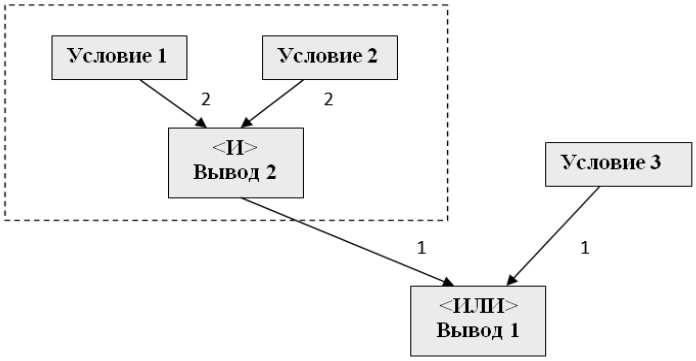

Рисунок 2.5 – Схема представления правил с помощью И/ИЛИ-графа

На рис. 2.6 представлен И/ИЛИ-граф для правил подгруппы «Сеть» (см. сами правила в прил. А). В данной подгруппе собраны правила, касающиеся сети, в том числе значения параметров каналов связи между узлами и нагрузки на них. Отметим, что в подгруппе правил «Сеть» присутствуют не только терминальные правила (это правило 1 на рисунке 2.6), но и промежуточные (это правила 2-5 на рисунке 2.6). И/ИЛИ-граф в этом случае отображает как цепочку для каждого правила (посылка-…-посылка-заключение), так и иерархию правил: на верхнем уровне – правило 1, на втором – правила 2 и 4, на третьем – правила 3 и 5).

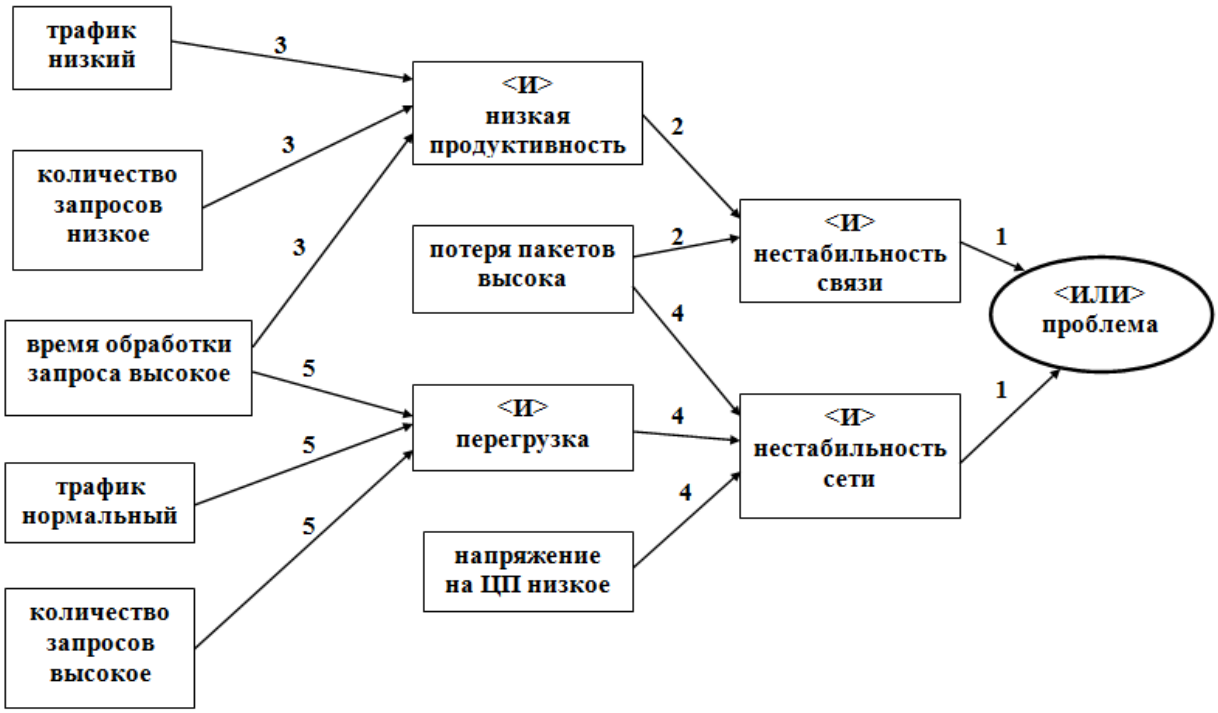

Рисунок 2.6 – И/ИЛИ граф для подгруппы правил «Сеть»

В случае же, когда в подгруппе правил присутствуют лишь терминальные правила, содержащие все лишь одно и то же заключение, а правила содержат много посылок более наглядным будет иное представление правил в виде И/ИЛИ-графа, а именно, все посылки каждого правила и заключение являются путем в графе.

Пример такого представления приведен на рис. 2.7 – И/ИЛИ-граф для правил подгруппы «Правила, зависящие от внешних условий» для экспертной системы контроля по безопасной работе с электроустановками. В рамках данной подгруппы присутствуют пять правил, при которых работы в электроустановках выполнять нельзя (см. сами правила в подразделе 4.3.2), и все правила – терминальные, то есть в выводах содержат вердикт «Работы выполнять нельзя».

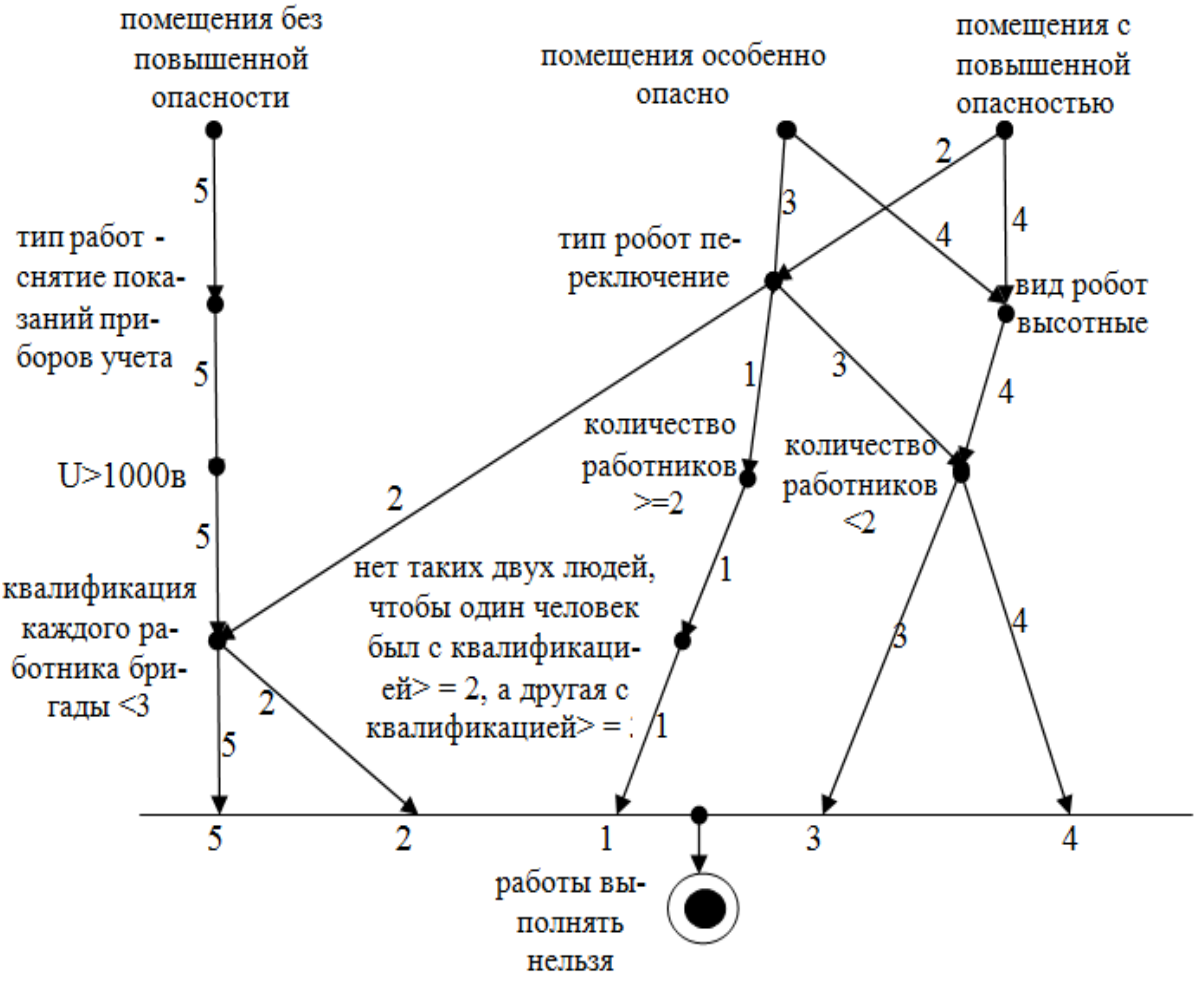

Рисунок 2.7 – И/ИЛИ-граф для правил, зависящих от внешних условий

При таком представлении меток на дуге, которые определяют номер правила, может быть несколько, так как одна и та же посылка может входить в несколько правил, значит, предлагаемый граф является мультиграфом, допускающим кратные дуги, то есть несколько дуг между двумя вершинами.

Возможно, *преобразование* из *одной формы* представления правил *в другую*. Для преобразования И/ИЛИ-графа в правила в текстовом виде предложен алгоритм, описанный в подразделе 3.4.

Полученные правила могут быть *автоматически преобразованы* в правила на каком-либо языке *продукционного программирования* для дальнейшего их использования (возможно, с некоторой доработкой) в правила знаниеориентированной системы.

И/ИЛИ-граф можно *анализировать и обрабатывать* с помощью методов *теории графов* и *математической логики*, учитывая специфику предметной области контроля. Предложены методы проверки противоречивости посылок правил и проверки на полноту, описанные в разделе 3.

В результате визуализации правил в форме представленного И/ИЛИ-графа, используя редактор правил для ввода и корректировки вершин и дуг, сохранения введенного в файле и считывания из файла, инженеру по знаниям и эксперту проще проанализировать и выявить несоответствия и ошибки, их исправить как в текстовом виде, так и непосредственно в графе.

Далее приведем пример нечетких правил для ПрО «Безопасная работа с электроустаовками»(см. табл. 2.2).

Данные правила содержат в условиях и нечеткие, и четкие посылки. Нечеткие посылки представляем в виде нечетких переменных. В правилах представлены две лингвистические переменные: “влажность” v и “температура” t. Переменная v может быть представлена 3 нечеткими переменными:

– <v1= “влажность низкая или нормальная”, X, A1>
– <v2= “влажность высокая”, X, A2>
– <v3= “влажность очень высокая”, X, A3>,

где X = [0,100] - универсум, A1, A2, A3 – нечеткие множества (см. рис.2.8).

Таблица 2.2 – Пример нечетких правил

Примеры правил	Четкость/нечеткость в посылках правил
Если влажность помещения высокая или	нечеткость
токопроводящие полы или	четкость
температура в помещении высокая или	нечеткость
… то помещение с повышенной опасностью	
Если влажность помещения очень высокая или	нечеткость
органическая среда или	четкость
… то помещение особо опасное	

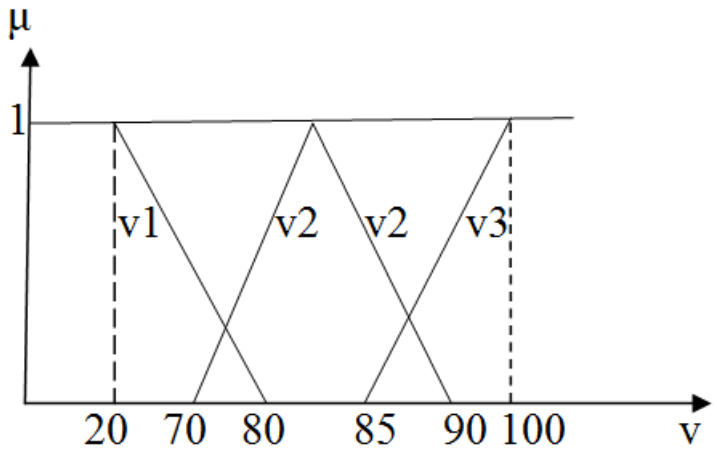

Рисунок 2.8 – Представление лингвистической переменной “влажность”

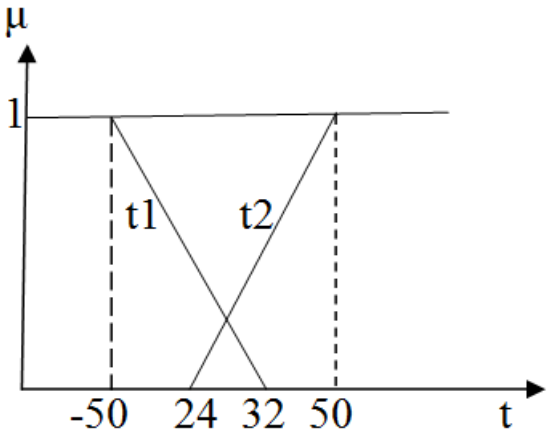

Рисунок 2.9 – Представление лингвистической переменной “температура”

Лингвистическая переменная t может быть представлена 2 нечеткими переменными:

– <t1= “температура низкая или нормальная”, X, B1>

– <t2= “температура высокая”, X, B2>,

где X={-50, 50}, см. рис. 2.9.

На рис. 2.8 и рис. 2.9 использовалась «треугольная» функция принадлежности. Эта функция используется для задания таких свойств множеств, которые характеризуют неопределенность типа: «расположен в интервале». Она служит для представления нечетких чисел и интервалов.

Выбор такой формы функции принадлежности обусловлен тем, что были найдены в литературе аналогичные формы и значения для лингвистических переменных "влажность" и "температура".

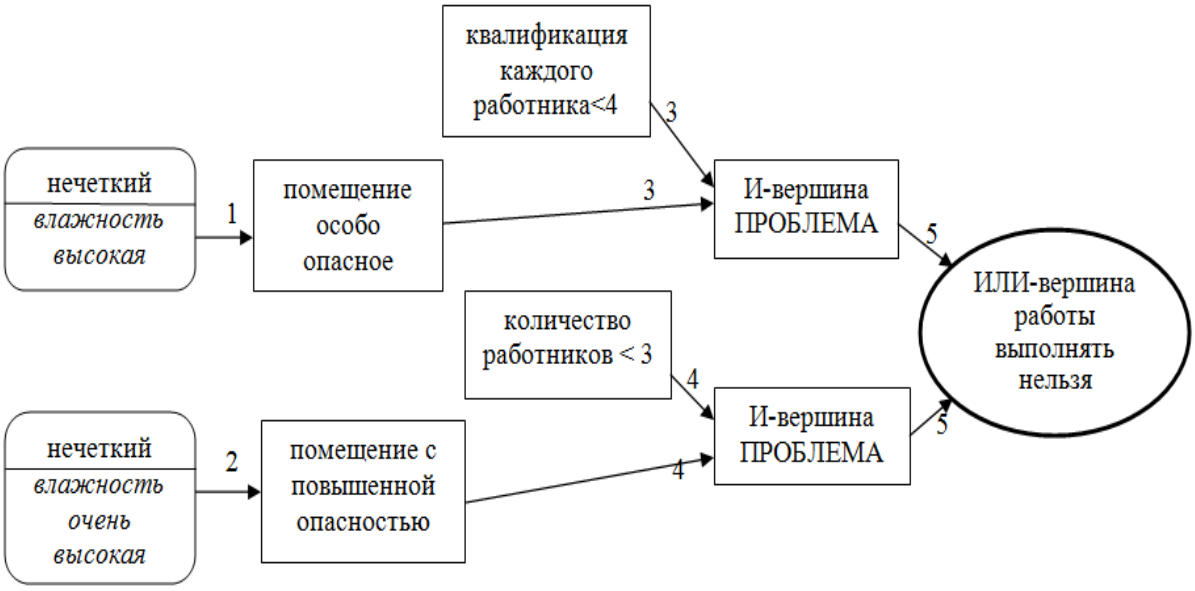

Рисунок 2.10 – Пример И/ИЛИ-графа с нечеткими вершинами

На рис 2.10 показано, что при построении графа помечаются не только вершина-вывод и номера правил, но и вершины, которые обозначают нечеткие переменные: «влажность высокая» и «очень высокая», также для этих вершин эксперт может вводить, просматривать и редактировать их диапазоны, как это представлено на рис. 2.8 и рис. 2.9. Далее И/ИЛИ-граф автомати-

чески преобразуется в БЗ на языке продукционного программирования. Начинает работать знаниеориентированная система, при этом машина вывода работает с полученной БЗ.

Нечеткие переменные, такие как “влажность воздуха”, “температура”, попадают в знаниеориентированную систему с датчиков в виде числовых значений, а затем машина вывода производит фазификацию, преобразуя с использованием нечетких множеств, числа в нечеткие переменные. Например, если с датчика влажности пришло значение 87%, то с уверенностью эксперта, равной 0.5, будет сделан вывод, что влажность высокая, а с уверенностью эксперта 0.2, что влажность очень высокая, а с уверенностью эксперта 0, что низкая или нормальная (рис.2.8, 2.9). В результате, если переменная «тип помещения» – нечеткая, то будет сделан вывод, что помещение с повышенной опасностью с уверенностью эксперта 0.5, а с повышенной опасностью – с уверенностью эксперта 0.2. А если «тип помещения» – четкая переменная (как видно на графе), то мы заранее на графе указываем, что это противоположные переменные, и будет сделан вывод по мах уверенности эксперта (0.5), что помещение с повышенной опасностью. Дальше уже вывод будет происходить с четкими переменными.

В более сложных случаях, если эксперт будет проводить проверки правил, то сначала будет произведена минимизация правил, мы получим прямые правила:

Если *влажность высокая* и квалификация каждого работника <4, то работы выполнять нельзя.

Если *влажность очень высокая* и, количество работников <3, то работы выполнять нельзя.

Существуют различные математические определения основных операций над нечеткими множествами, однако чаще всего используются следующие: чтобы получить уверенность эксперта для нечеткого множества при отрицании, необходимо из 1 вычесть уверенность эксперта исходного нечеткого множества, уверенность эксперта конъюнкция равна min уверенности экс-

перта, уверенность эксперта дизъюнкции – мах уверенностей эксперта [31, 47, 51]. Таким образом, если высказывание «температура нормальная» имеет достоверность 0.8, то противоположное ему высказывание «температура высокая» имеет уверенность эксперта 0.2. Если в лингвистической переменной три нечеткие переменные, то $\bar{c}$ = a v b, где v – нечеткая дизъюнкция.

Для получения «прямых» правил производится преобразование в нечеткую МДНФ [48], а для «инверсных» правил – нечеткое «НЕ».

Таким образом, из вышеприведенных рассуждений можно сделать следующий вывод. Для нечетких правил целесообразно конструировать И/ИЛИ-граф, строить «прямые» и «инверсные» правила и на их основе проводить проверки на противоречивость, полноту, достижимость состояний и цикличность, а затем из него получать БЗ на языке продукционного программирования с нечеткими правилами. Проверка на противоречивость требует на время проверки замены нечетких переменных на четкие.

В существующих моделях на основе И/ИЛИ-графа все вершины разделены на три непересекающихся класса: «И»-вершина, «ИЛИ»-вершина, конечные (или целевые) вершины. Новизна предложенной модели правил в виде И/ИЛИ-графа состоит в том, что к классическому И/ИЛИ-графу впервые добавлены дополнительные свойства, описывающие правила, а именно, описания нечетких и взаимопротивоположных вершин для посылок правил, а также номера правил в качестве меток для дуг. При этом правила разделены на группы, и каждой группе соответствует свой И/ИЛИ-граф с одной конечной вершиной.

2.4 Создание математической модели правил контроля в виде булевых выражений

В 2.3 предложена модель для правил контроля в виде И/ИЛИ-графа, на основе которой предлагается представлять наборы правил в виде логических булевых формул, а специфика предметной области контроля позволяет это сделать эффективно с точки зрения анализа и проверки качества правил.

Рассмотрим далее преобразование И/ИЛИ-графа для набора правил контроля в булевы выражения и получим булевы формулы для так называемых «прямого» и «инверсного» наборов правил.

Для простоты изложения примем два ограничения.

I. Все посылки правил связаны операцией И. А если есть одна или несколько ИЛИ-вершин в цепочке правила, то такое правило можно разбить на несколько правил, содержащих только И-вершины. Такое преобразование можно сделать всегда, так как всегда из любой булевой формулы можно получить ДНФ. Например, следующее правило:

Если помещение с повышенной опасностью или помещение особо опасное и вид работ – высотные, и количество работников <2, то работу выполнять нельзя

Такое правило можно разбить на следующие два правила:

Если помещение с повышенной опасностью, и вид работ – высотные, и количество работников <2, то работу выполнять нельзя

Если помещение особо опасное, и вид работ – высотные, и количество работников <2, то работу выполнять нельзя

II. Если в подгруппе есть промежуточные (нетерминальные правила), то их можно удалить, включив в терминальные правила (для этого используется минимизация булевых формул и получение МДНФ). Например, в группе два правила:

1. Если потеря пакетов высока и производительность узла сети низкая, то связь с узлом сети нестабильна.

2. Если связь с узлом сети нестабильна, или наблюдается перегрузки сети, то налицо ПРОБЛЕМА, требующая внимания системного администратора.

Такие правила могут быть преобразованы в одно правило:

Если потеря пакетов высока и производительность узла сети низкая, или наблюдается перегрузки сети, то налицо ПРОБЛЕМА, требующая внимания системного администратора.

Минимизацию булевых формул предлагается производить с помощью метода Куайна – Мак – Класки [63].

Таким образом, после преобразования правил в соответствии с вышеизложенными ограничениями все правила подгруппы, для которой строится граф, имеют один и тот же вывод, в частности, для системы по безопасной работе с электроустановками – это «нельзя выполнять работы», а также посылки всех правил связаны отношением И.

Формально такого рода правила выглядят следующим образом:

$$\begin{gathered} p_1 \rightarrow \bar{w} \\ p_2 \rightarrow \bar{w} \\ \dots \\ p_k \rightarrow \bar{w} \end{gathered} \qquad (2.1)$$

где p_i – конъюнкция посылок правил i, (i=1,…,k), k – количество правил, $p_i = c_{i,1} \wedge c_{i,2} \wedge \dots \wedge c_{i,j} \wedge \dots \wedge c_{i,n_i}$ где $c_{i,J}$ – j-ая посылка i-го правила (j=1,…,n_i), $\bar{w}$ – выход состояние объекта системы, для которого проводится контроль, за пределы допустимого (для системы с электроустановками – нельзя выполнять работы), w – состояние объекта системы в норме (работы выполнять можно). Эти правила означают, что при выполнении посылок правил происходит выход состояния объекта за пределы допустимого (работы выполнять нельзя), а иначе состояние объекта в норме (работы выполнять можно). Такой набор правил назовем «прямым».

Итак, получено k булевых формул (2.1). Далее объединим эти булевы формулы в одну:

$$p_1 \vee p_2 \vee \dots \vee p_k \rightarrow \bar{w} \qquad (2.2)$$

Приведем формальное доказательство правомочности такого объединения булевых формул. В приведенных ниже описаниях p_i обозначают посылки правил, где i – номер правила, B – вывод правил. Длинная горизонтальная

черта при формулировке утверждений разделяет то, от чего отталкиваются в утверждении (то, что дано), и то, что нужно доказать.

Утверждение 1

$p_1 \to B$

$p_2 \to B$

$p_1 \vee p_2 \to B$

Доказательство:

$(p_1 \to B) \wedge (p_2 \to B) \leftrightarrow (p_1 \vee p_2) \to B$

Что и требовалось доказать. А именно, что если есть несколько правил в БЗ, и вывод у них один тот же, то из дизъюнкции посылок следует этот же вывод. Таким образом, показана правомочность преобразования набора правил в булево выражение $p_1 \vee p_2 \vee ... \vee p_k \to \bar{w}$, где p_i – логическое выражение, соответствующее посылкам правила i.

Далее, для правил контроля возможно из булевой формулы (2.2) получить формулу, содержащую эквивалентность:

$$p_1 \vee p_2 \vee ... \vee p_k \leftrightarrow \bar{w} \qquad (2.3)$$

Приведем формальное доказательство правомочности такого преобразования.

Утверждение 2

$p_1 \to B$

$p_2 \to B$

$\overline{p_1} \wedge \overline{p_2} \to \bar{B}$

$(p_1 \vee p_2 \leftrightarrow B) \vee (\overline{p_1 \vee p_2} \leftrightarrow \bar{B})$

Доказательство:

$\overline{p_1} \wedge \overline{p_2} = \overline{p_1 \vee p_2}$ по закону де Моргана, так как по условию $\overline{p_1} \wedge \overline{p_2} \to \bar{B}$, то и $\overline{p_1 \vee p_2} \to \bar{B}$, отсюда $B \to p_1 \vee p_2$. Из Утверждения 1 следует $p_1 \vee p_2 \to B$. Значит, $p_1 \vee p_2 \leftrightarrow B$. Из эквивалентности следует и $\overline{p_1 \vee p_2} \leftrightarrow \bar{B}$. Что и требовалось доказать.

Утверждение 2 более «сильное», чем Утверждение 1: если есть несколько правил в базе знаний, и вывод у них один тот же, а также, если из того, что

все правила неверны, следует, что и общий вывод неверен, то дизъюнкция посылок эквивалентна выводу. То есть из дизъюнкции посылок следует этот же вывод, а также из вывода следует дизъюнкция посылок.

Таким образом, совокупность правил подгруппы может быть представлена в виде логической формулы, представляющей собой эквиваленцию, левая часть формулы – дизъюнктивная нормальная форма (ДНФ):

$$p_1 \vee p_2 \vee \ldots \vee p_k \leftrightarrow \bar{w}$$

Далее из формулы (2.3) на основе Утверждения 2 можно получить и так называемую «инверсную» формулу:

$$\overline{p_1 \vee p_2 \vee \ldots \vee p_k} \leftrightarrow w \qquad (2.4)$$

«Инверсная» булева формула (2.4) может быть преобразована в несколько булевых формул, соответствующих «инверсному» набору правил:

$$\begin{gathered} p_1' \rightarrow w \\ \mathrm{p}_2' \rightarrow \mathrm{w} \\ \ldots \\ \mathrm{p}_l' \rightarrow \mathrm{w} \end{gathered} \qquad (2.5)$$

где l – количество «инверсных» правил.

Приведем формальное доказательство правомочности такого преобразования.

Утверждение 3

$\overline{p_1 \vee p_2 \vee p_3} \leftrightarrow \bar{B}$ Приведем левую часть эквивалентности к ДНФ: $p_1' \vee p_2' \vee p_3' \leftrightarrow \bar{B}$

$p_1' \rightarrow \bar{B}$ $p_2' \rightarrow \bar{B}$ $p_3' \rightarrow \bar{B}$

Доказательство:

Если p_1' – истина, то $p_1' \vee p_2' \vee p_3'$– тоже истина. А так как $p_1' \vee p_2' \vee p_3' \leftrightarrow \bar{B}$, то $\bar{B}$ – истина. То есть, из p_1' – истина следует $\bar{B}$ – истина.

Аналогично доказывается и для остальных p_i'. Что и требовалось доказать.

Этим мы доказали, что из «прямых» правил БЗ (2.1), у которых один и тот же вывод, правомерно получать так называемые «инверсные» правила (2.5), у которых вывод противоположный относительно исходных правил.

(2.5) представляет так называемую «инверсную» подгруппу правил, которые означают, что при выполнении посылок правил отслеживаемое состояние объекта в норме (работы выполнять можно), а иначе происходит выход состояние объекта за пределы допустимого (работы выполнять нельзя).

В существующих моделях правил в виде булевых выражений присутствуют лишь формулы, аналогичные 2.1 (в общем случае с разными выводами правил), и полученные из них преобразованные с помощью логических законов. Частный случай модели правил, когда выводы у всех правил одинаковы, не рассматривался. Здесь же в связи со спецификой задачи контроля выводы у всех правил формулы (2.1) одинаковы, и на этой основе впервые получены формулы(2.2)-(2.5).

Формулы (2.1)-(2.5) представляют собой математическую модель правил в виде булевых выражений и наряду с моделью в виде И/ИЛИ-графа будут использоваться далее, в частности, для проверки качества наборов правил.

2.5 Выводы по второму разделу

Во втором разделе проведена постановка задачи контроля, представлены усовершенствованные модели правил в виде И/ИЛИ-графа и в виде булевых выражений с учетом специфики задачи контроля.

1. Предложена функциональная схема знаниеориентированных систем контроля, которая состоит из двух частей: 1) Редактор правил предусматривает ввод, корректировку и проверку правил и добавления их в БЗ. В начале разработки эту работу выполняет инженер по знаниям вместе с экспертом, но одно из главных требований к редактору правил состоит в облегчении работы с правилами таким образом, чтобы изменение условий контроля на этапе сопровождения системы могло выполняться преимущественно экспертом. 2)

Система контроля обеспечивает получение параметров объекта, анализ с использованием машины вывода и базы правил, результат передается ЛПР для принятия решений.

2. На основе синтаксической структуры знаний выделены особенности знаний для контроля и сделан вывод, что построение структуры знаний для этой задачи можно облегчить за счет разбиения правил на группы, в каждой из которых анализируется одно состояние объекта, выбора наиболее приемлемой формы для визуализации и интерактивной работы с правилами, а также методов их проверки на ранних этапах разработки.

3. Предложена модель для построения, визуализации и интерактивной работы с правилами знаниеориентированных систем контроля на основе усовершенствованного И/ИЛИ-графа с дополнительной разметкой, помечающей взаимно-противоположные и нечеткие вершины, соответствующие посылкам правил, вершину для состояния объекта контроля, а также дуги, на которых отмечаются номера правил. Предложены два способа визуализации правил. По результатам структуризации знаний правила конвертируются в язык продукционного программирования для дальнейшего использования в знаниеориентованых системах.

4. Предложена математическая модель правил контроля в виде булевых выражений, а специфика предметной области контроля позволяет это сделать эффективно с точки зрения проверки правил. Показано, что совокупность правил контроля для группы может быть представлена в виде так называемых «прямой» и «инверсных »логических формул. Доказано три утверждения, подтверждающие правомерность предложенных преобразований. Модель правил контроля в виде булевых выражений используется при построении методов проверки правил знаниеориентированных систем контроля.

Результаты исследований раздела 2 опубликованы в работах автора [22, 70, 74, 76].

РАЗДЕЛ 3
РАЗРАБОТКА МЕТОДОВ ПРОВЕРКИ ПРАВИЛ

Знания могут быть субъективными, неполными, противоречивыми либо даже ошибочными, и для устранения этих недостатков приходится проделывать большой объем механической работы, которую можно и нужно автоматизировать – это, в частности, проверки правил.

Дедуктивная теория считается заданной, если:

– Задан алфавит и правила образования выражений (слов) в этом алфавите.

– Заданы правила образования формул (правильно построенных, корректных выражений).

– Из множества формул некоторым способом выделено подмножество T теорем (доказуемых формул).

Далее рассматриваются понятия противоречивости и полноты, как с точки зрения дедуктивных теорий, так и для знаниеориентированных систем, приведено описание разработанных методов проверки правил на противоречивость, полноту и достижимость состояний объекта контроля, основанных на предложенной модели правил в виде булевых выражений.

3.1 Метод проверки противоречивости условий правил знаниеориентированных систем контроля на основе задачи выполнимости булевых формул (SAT)

Теория противоречива, если существует такое утверждение, что из теории вытекает и оно, и его отрицание:

$$T \to \varphi \text{ и } T \to \overline{\varphi},$$

где T – теория, φ – утверждение.

Непротиворечивой называется система, в которой каждая теорема является логически действительной правильно построенной формулой. Иными

словами, непротиворечивая система не позволяет вывести заключение, не являющееся логическим следствием из его посылок [39].

Поспеловым И. Г. [57] предложено классифицировать противоречия в знаниеориентированной системе на внешние (противоречия между системой продукций и моделью мира, имеющих природные аналоги в научных теориях) и внутренние противоречия в системе продукций. Последнее означает, что существуют, во-первых, правила, которые внутри себя содержат противоречия (то есть, например, некоторые посылки в них противоречат друг другу), а во-вторых, в системе правил есть два или более правил, которые сами по себе непротиворечивы, но одно правило противоречит другому либо нескольким другим.

Если система продукций содержит правила, делающие противоречивые выводы, возможно появление конкурирующих гипотез, и заключительные рекомендации системы будут зависеть от стратегии выбора правил (от способа разрешения коллизии) [66]. В результате система может сделать не те выводы, которые ожидал бы от нее эксперт, или вообще не прийти ни к какому решению.

Для проверки противоречивости правил контроля в связи с их спецификой, описанной в 2.1, предлагается использовать булевы формулы, соответствующие набору правил (см. 2.3), и задачу выполнимости булевых формул (SATisfiability problem, SAT) [79].

Входом SAT является булева формула, состоящая из имен переменных, логических констант, скобок и логических операций И, ИЛИ и НЕ. Задача заключается в нахождении ответа на вопрос: можно ли назначить переменным, встречающимся в формуле, значения ЛОЖЬ и/или ИСТИНА так, чтобы формула стала истинной? То есть необходимо определить, бывает ли формула истинна хотя бы при одном определенном наборе значений переменных. Если при некоторых значениях переменных формула принимает значение ИСТИНА, то булева формула выполнима, иначе – невыполнима (рис. 3.1).

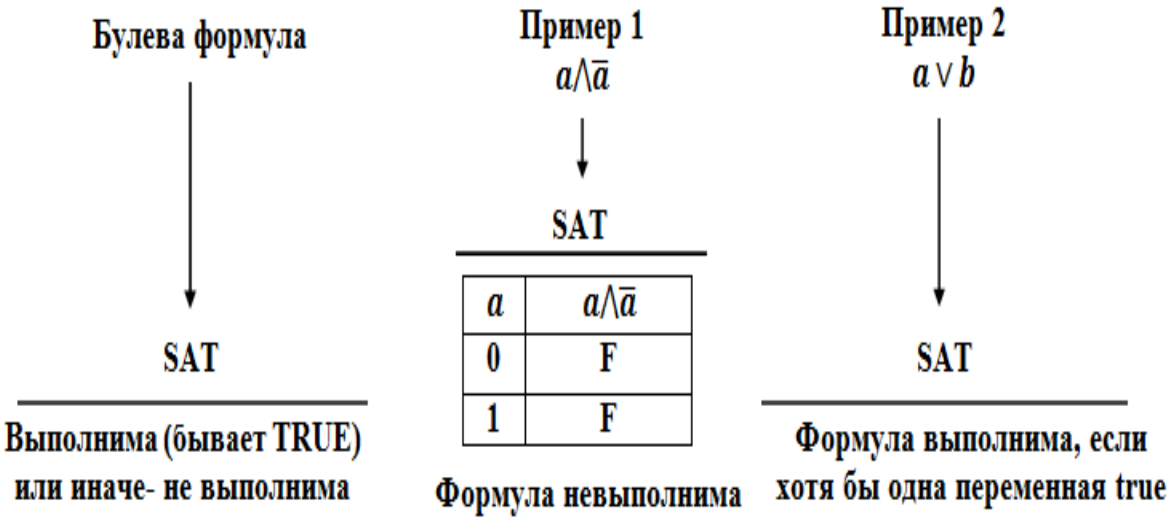

Рисунок 3.1 – Задача выполнимости булевых функций SAT

По одной из *классификаций* задача выполнимости булевых формул делится на два типа: k-SAT и неограниченная SAT. Задача является k-SAT при условии, что формула, которая подается на вход, записана в k-конъюнктивной нормальной форме. *k*-конъюнктивной нормальной формой называют конъюнктивную нормальную форму, в которой каждая дизъюнкция содержит РОВНО *k* литералов, иначе задача SAT неограничена.

Теперь рассмотрим, каким образом задача SAT может быть применена при анализе противоречивости посылок правил контроля. Для эксперта и инженера по знаниям при написании правил важно исключить противоречия посылок внутри каждого правила, а также противоречия между посылками разных правил.

При проверке противоречивости будем подавать на вход SAT левые части формул (2.3) и (2.4), количество литералов у которых в общем случае неодинаково, а зависит от количества посылок в правилах. Поэтому используется так называемая неограниченная SAT.

Предлагается *метод проверки посылок правил контроля на противоречивость* (рис. 3.2), включающий три этапа проверки.

Этап 1. Для проверки так называемой «внутренней» противоречивости («внутренняя» означает, что производится проверка в пределах каждого пра-

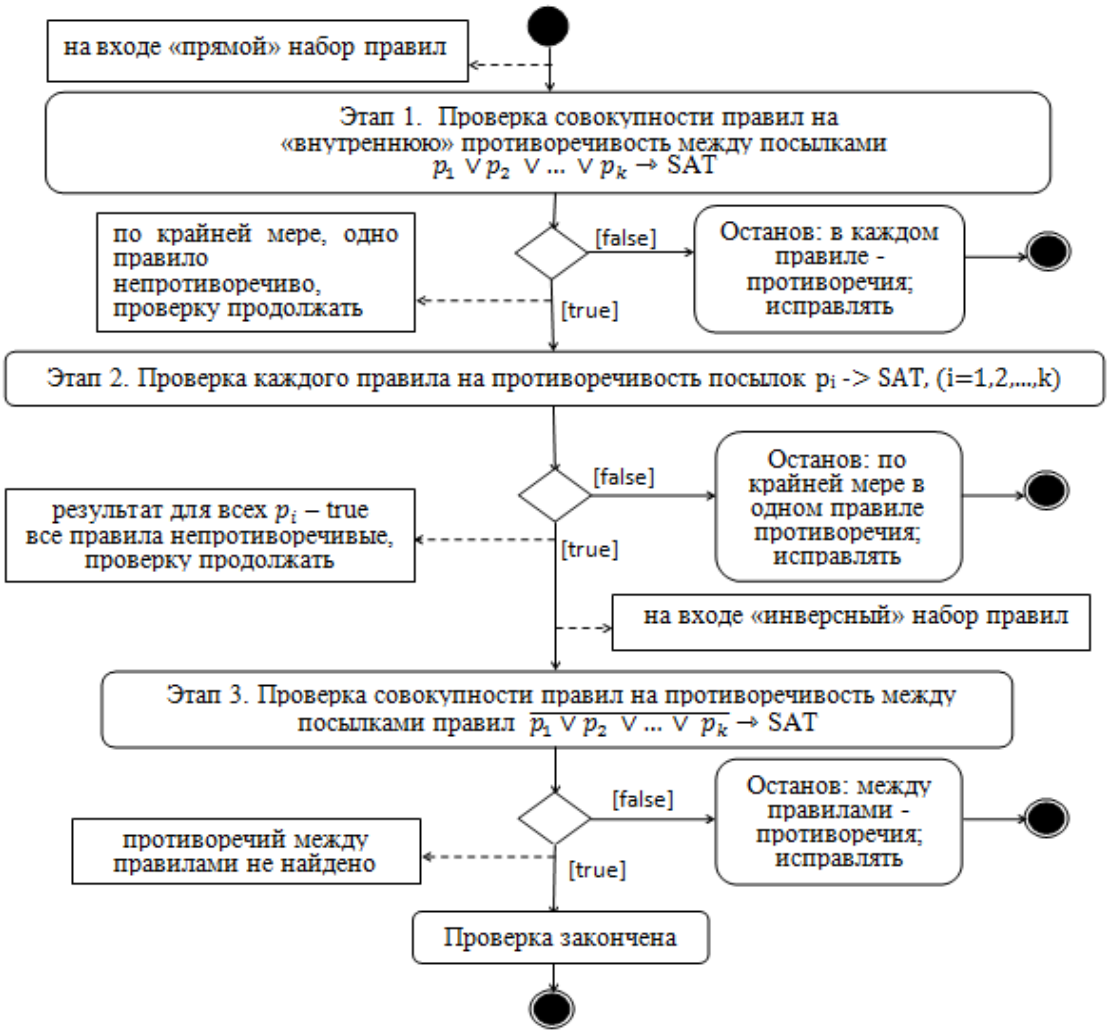

Рисунок 3.2 – Этапы проверки правил контроля на противоречивость

вила, а не между правилами) всех правил подгруппы подаем левую часть формулы (2.3) на вход задачи SAT, а именно, $p_1 \vee p_2 \vee \ldots \vee p_k$. Если результат решения задачи ЛОЖЬ, это значит, что все части в ДНФ для любых значений входящих в них переменных ложны, то есть каждое правило подгруппы содержит внутри себя противоречия, а именно, противоречащие друг другу посылки. Пример такого набора из одного правила для ПрО по безопасной работе с электроустановками приведен в таблице 3.1 (пример 1). Получив такой результат, делаем вывод, что ни одно правило в подгруппе никогда не сработает, то есть система, никогда не выдаст вердикт о том, что отслеживаемое состояние объекта выходит за пределы допустимого, нужно исправлять каждое правило, и общая проверка на противоречивость окончена.

Если результат задачи SAT – ИСТИНА, это значит, что, по крайней мере, одно правило в подгруппе не содержит противоречий, и не более того, таким образом, требуется дальнейшая проверка.

Таблица 3.1 – Примеры проверки правил на противоречивость

	Пример 1	**Пример 2**	**Пример 3**
Правило(а)	Если U>1000В и U≤1000В, то работы выполнять нельзя	Если U>1000В, то работы выполнять нельзя Если U≤1000В, то работы выполнять нельзя	Если U>1000В и бригада меньше трёх человек, то работы выполнять нельзя Если U≤1000В, то работы выполнять нельзя Если бригада больше или равна 3-м работникам, то работы выполнять нельзя
Обозначения посылок	**a** - U>1000В	**a** - U>1000В	**a** - U>1000В, **b** – бригада меньше трёх человек
Булева формула «прямая»/ «инверсная»	$a \wedge \bar{a} \rightarrow SAT$	$\overline{a \vee \bar{a}} \rightarrow SAT$	$\overline{ab \vee \bar{a} \vee \bar{b}} = \overline{ab} \vee \overline{\bar{a}\bar{b}} \rightarrow SAT$
Результат задачи SAT	**всегда = FALSE**	**всегда = FALSE**	**всегда = FALSE**
Вывод	***Правило никогда не выполняется удаляем из системы***	***Система правил не имеет смысла, правила срабатывают всегда***	***Нужно исправлять правила, срабатывают всегда***

Этап 2. Для проверки посылок каждого правила на «внутреннюю» противоречивость подаем на вход задачи SAT поочередно конъюнкции посылок правил p_i, i=1,2,…,k (формула (2.2)).

Если результат решения задачи для некоторого правила ЛОЖЬ, это означает, что посылки этого правила содержат противоречие, а именно, конъюнкция содержит одновременно x и NOT x для некоторой переменной x (таблица 3.1, пример 1). Такое правило никогда не выполнится, нужно его либо исправлять, либо удалять из системы. Если хотя бы одно правило оказалось противоречивым, общая проверка на противоречивость завершается.

Если результат решения для всех правил ИСТИНА, это значит, что все правила в подгруппе не содержат внутри своих посылок противоречий, и требуются дальнейшие этапы проверки.

Этап 3. Для проверки противоречивости между посылками разных правилами на вход SAT подаем левую часть формулы (2.4), а именно, $\overline{p_1 \vee p_2 \vee \ldots \vee p_k}$. Напомним, что эта логическая форма эквивалентна w – состояние объекта системы в норме (работы выполнять можно).

Если результат решения задачи ЛОЖЬ, это значит, что формула $p_1 \vee p_2 \vee \ldots \vee p_k$ всегда истинна, то есть имеет место тавтология, а такое возможно, когда, по крайней мере, одна пара членов дизъюнкции представляют собой некоторую переменную x и ее отрицание, например, формула $p_1 \vee p_2 \vee \ldots p_i \vee \overline{p_i} \vee \ldots \vee p_k$ всегда истинна, потому что всегда истинна $p_i \vee \overline{p_i}$. А это, в свою очередь, говорит о том, что, по крайней мере, посылки двух правил в подгруппе противоречивы.

В терминах знаниеориентированной системы результат ЛОЖЬ говорит о том, что всегда происходит выход состояния объекта за пределы нормы, и ни при каких посылках правил знаниеориентированная система не выдаст сообщения о том, что отслеживаемое состояние объекта в норме (работы выполнять можно). Таким образом, выход за пределы нормы предопределен, потому что существуют противоречащие друг другу посылки в разных правилах (см. пример 2 в таблице 3.1).

Аналогичен случай, если противоречат между собой не два правила, а больше, чем два правила (пример 3 в таблице 3.1). Например, $ab \vee \bar{a} \vee \bar{b} = ab \vee \overline{ab}$. Тогда, если мы подадим формулу $\overline{ab \vee \overline{ab}}$ на вход SAT, то никогда не получим ИСТИНА, что свидетельствует о противоречивости посылок этих 3-х правил.

На рисунке 3.3 в текстовом и графическом виде представлены правила для ПрО контроля компьютерной сети, содержащие противоречия в посылках.

Рис. 3.3 состоит из 2-х частей – левой и правой. Каждое правило подаем на вход задачи SAT. Формула для каждого правила означает, что система охлаждения неэффективна. Таким образом, мы определяем, есть ли такой

Рисунок 3.3 – Проверка правил на основе SAT

набор значений переменных, при котором эта формула ИСТИНА (т.е. система охлаждения неэффективна). В первом случае (левая часть рис. 3.3), если нет, т.е. задача SAT выдала ЛОЖЬ, то наша ЭС никогда не скажет, что система неэффективна. Это значит, что такое правило никогда не сработает, и его можно убрать из системы правил.

Справа на рис. 3.3 изображено отрицание посылок совокупности правил, которое подаем на вход задачи SAT. Инверсная формула означает, что система охлаждения эффективна. Т.о. мы определяем, есть ли такой набор значений переменных, при котором эта формула ИСТИНА (т.е. система охлаждения эффективна.). Если задача SAT выдала ЛОЖЬ, то нет таких значений переменных, при которых система охлаждения эффективна, т.е. она всегда неэффективна. Это значит, что хоть одно из изначальных правил будет сра-

батывать всегда, и всегда будет происходить вывод, что система охлаждения неэффективна. Такие правила несовместимы, т.е. противоречивы.

Понятно, что такие противоречия представляются очевидными, однако они могут «затеряться» в правилах, где имеется несколько посылок; а такая проверка осуществляется автоматически.

Если задача SAT выдала ИСТИНА, то противоречий между посылками правил нет, и при определенных значениях параметров состояние объекта будет в норме (работу можно будет выполнять).

Основные достоинства этого метода состоят в том, что проверка противоречивости посылок правил производится автоматически и с учетом специфики задач контроля.

Метод проверки посылок правил на противоречивость позволяет находить как противоречия, которые были привнесены при конструировании правил, так и присутствующие в оригинальных текстах, в частности, в нормативных документах.

3.2 Метод проверки правил знаниеориентированных систем контроля на полноту на основе визуализации «инверсных» правил и конструирования недостающих

Теория полна, если для любого утверждения можно вывести из теории, что оно или верно, или неверно:

$$T \rightarrow \varphi \text{ или } T \rightarrow \overline{\varphi}$$

(для любого φ), где Т – теория, φ – утверждение.

Множество аксиом считается полным, если с его помощью можно доказать или опровергнуть любую правильно построенную формулу. Термин «опровергнуть» означает доказать, что некоторое утверждение ложно. В полной системе каждая логически действительная правильно построенная формула является теоремой.

Под полнотой (неполнотой) для знаниеориентированной системы понимается достаточность (недостаточность) знаний теории Т для решения задач с использованием ЭС, то есть получения результатов логического вывода при определении верны или нет утверждения φ, для проверки которых создавалась ЭС.

В работе Поспелова Л. Я. [57] неполнота определяется как вид дефекта, который носит скорее содержательный, чем формальный характер и выражается в неспособности системы делать выводы для ряда определенных исходных ситуаций, то есть критерий полноты определяет, насколько набор правил позволяет охватить все возможные комбинации исходных данных. Формально неполнота может проявляться в следующем: пропущенные факты, например, из-за “очевидности” некоторых знаний эксперт может не указать их в явном виде; пропущенные правила – этот дефект выражается как тупики в цепочках логического вывода, а каждый атрибут заключения правила должен быть либо целевым, либо содержаться в условиях других правил; каждый атрибут условия правила должен быть либо заключением другого правила, либо терминальным (значение атрибута вводится пользователем, запрашивается в базе данных, снимается измерительным устройством, вычисляется процедурой); недостижимые цели, то есть ни одна из возможных цепочек логического вывода не приводит к данному значению целевого атрибута; невостребованные факты, то есть терминальный факт не входит в условия правил; несвязанные сегменты знаний.

В [42] в случае индуктивного (на основе примеров) построения правил знаниеориентированной системы неполнота подразумевает отсутствие у обучающего множества свойства репрезентативности, то есть наличия достаточного количества примеров для представления всей генеральной совокупности, а не только отдельных её классов.

3.2.1 Проверка полноты правил знаниеориентированных систем контроля

Для проверки полноты правил контроля в связи с их спецификой, описанной в 2.2, предлагается использовать булевы формулы, соответствующие «прямому» набору правил (см. 2.3), и автоматическое дополнение набора правил противоположным по смыслу («инверсным»), а также оценка экспертом правил противоположного набора, которые определяют, в каких случаях выполняются заключения, противоположные начальным. При выявлении экспертом на основе «инверсных» правил недостающих знаний предлагается помощь эксперту в конструировании новых правил.

«Инверсный» набор правил для проверки полноты нужен, поскольку на этапе разработки знаниеориентированной системы набор правил может оказаться неполным, нужно задавать вопрос эксперту: «Что вы знаете такого, что еще не знает программа".

Эксперт имеет возможность работы с двумя визуальными вариантами правил в виде соответствующих И/ИЛИ-графов:

1) условия, при которых состояние объекта контроля аварийное;

2) условия, при которых состояние объекта в норме.

Примечательно, что для правил контроля набор правил группы, в котором описывается одно отслеживаемое состояние объекта, из одного визуального варианта правил можно автоматически получить другой визуальный вариант правил – это формулы (2.1)-(2.5).На основе такого рассмотрения эксперт в состоянии оценить, каких правилах не хватает, или в каких правилах не полностью заданы посылки.

В полученном «инверсном» И/ИЛИ-графе вершины соответствуют посылкам, при которых состояние объекта в норме. Если одно или несколько правил, когда можно выполнять работы, покажутся эксперту неверными, он сможет добавить в исходные правила еще дополнительные правила, либо дополнительные посылки к правилам, либо убрать неверные. Для помощи эксперту предлагается следующее: эксперт нажимает на проблемное «инверс-

ное» правило, система показывает ему все возможные способы дополнения полноты и так делается для всех проблемных, на взгляд эксперта, правил, причем, таким образом, эксперт формирует и «прямые», и «инверсные» правила, добавляя с его точки зрения правильные предъявляемые посылки и отмечая, что при таких посылках будет происходить, т.е. будет ли выходить состояние объекта за пределы допустимого; на следующем шаге автоматически из исправленных «инверсных» правил формируются «прямые» правила; затем автоматически проверяется противоречивость посылок полученных «прямых» правил; полученные «прямые» правила предъявляются эксперту, в случае противоречий выдается сообщение. Такая работа по конструированию правил производится итерационно, пока эксперт не будет удовлетворен построенным набором правил.

Таким образом, метод проверки правил контроля на полноту, основанный на построении «инверсного» набора правил из «прямого» набора и наоборот, состоит из следующих этапов (рис. 3.4):

Этап 1 Получение исходного И/ИЛИ-графа «прямого» набора правил и автоматическое преобразование И/ИЛИ-графа в булево выражение (формулу (2.3)).

Этап 2 Проверка «прямого» набора правил на противоречивость.

Этап 3 Преобразование «прямого» набора правил в «инверсный», строится «инверсная» формула (2.4) и автоматически приводится в ДНФ, а затем в «инверсный» набор правил (формула (2.5)) и предъявления его эксперту. Автоматически выполняется построение И/ИЛИ-графа «инверсного» набора правил.

Этап 4 Полученный И/ИЛИ-граф предъявляется для рассмотрения эксперту, который выбирает «проблемные» правил «инверсного» набора

Этап 5 Предъявление эксперту возможных способов дополнения «проблемных» правил, как «инверсных», так и «прямых» правил, при этом система автоматически «выясняет» возможные способы дополнения правил и

предъявляет их эксперту; эксперт дополняет набор новыми знаниями (новыми правилами или новыми посылками существующих правил).

Этап 6 Сконструированные «инверсные» правила автоматически преобразовываются в «прямые» и наоборот, и новый набор правил автоматически проверяется на противоречивость посылок и предъявляется эксперту для ознакомления.

Этап 7 Формирование нового набора «прямых» правил преобразованием с «инверсных» и добавление новых «прямых».

Этапы 1-7 повторяются итерационно, пока эксперт не удостоверится, что правила, с его точки зрения, удовлетворительны.

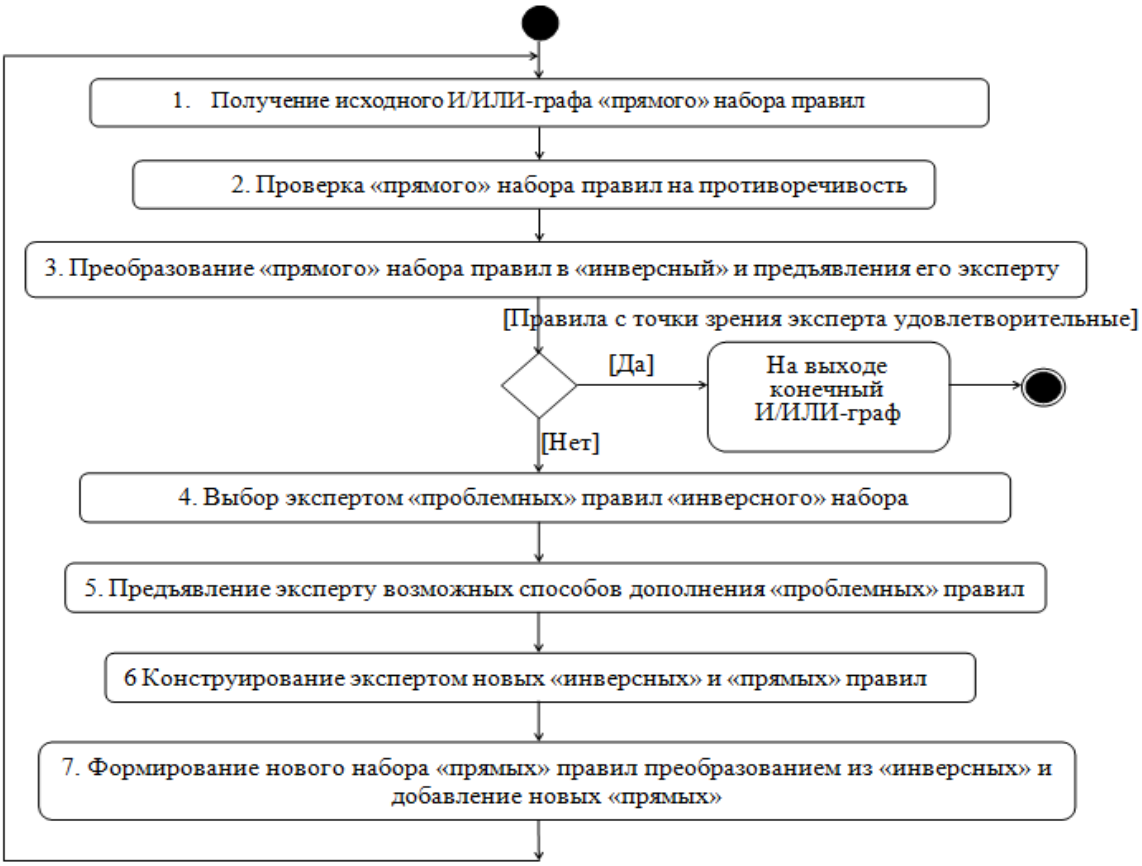

Рисунок 3.4 – Этапы проверки правил на полноту

Предложенный метод работает аналогично методу доказательства «от противного», которое проводится следующим образом: для доказательства утверждения A предполагают, что оно неверно, то есть верно A, затем доказывают, что из $\bar{A}$ следует некоторое заведомо неверное утверждение B. В предлагаемом методе в качестве A выступает «прямой» набор правил, $\bar{A}$ –

«инверсный» набор, B – «инверсное» правило, которое с точки зрения эксперта неверно.

Далее рассмотрим проверку правил контроля на полноту на основе предложенного метода для двух предметных областей. Отметим, что в 3.2.2 описано, как метод применялся при автоматическом (программном) построении «инверсных» правил, в 3.2.3 расчет производился вручную.

3.2.2 Пример проверки правил контроля и мониторинга компьютерной сети на полноту

Рассмотрим проверку на полноту на примере группы правил знаниеориентированной системы контроля мониторинга компьютерной сети.

В приложении А представлены правила подгруппы «Сеть», отвечающие за общие проблемы в сети, а на рис. 3.5 эти правила показаны в виде И/ИЛИ-графа.

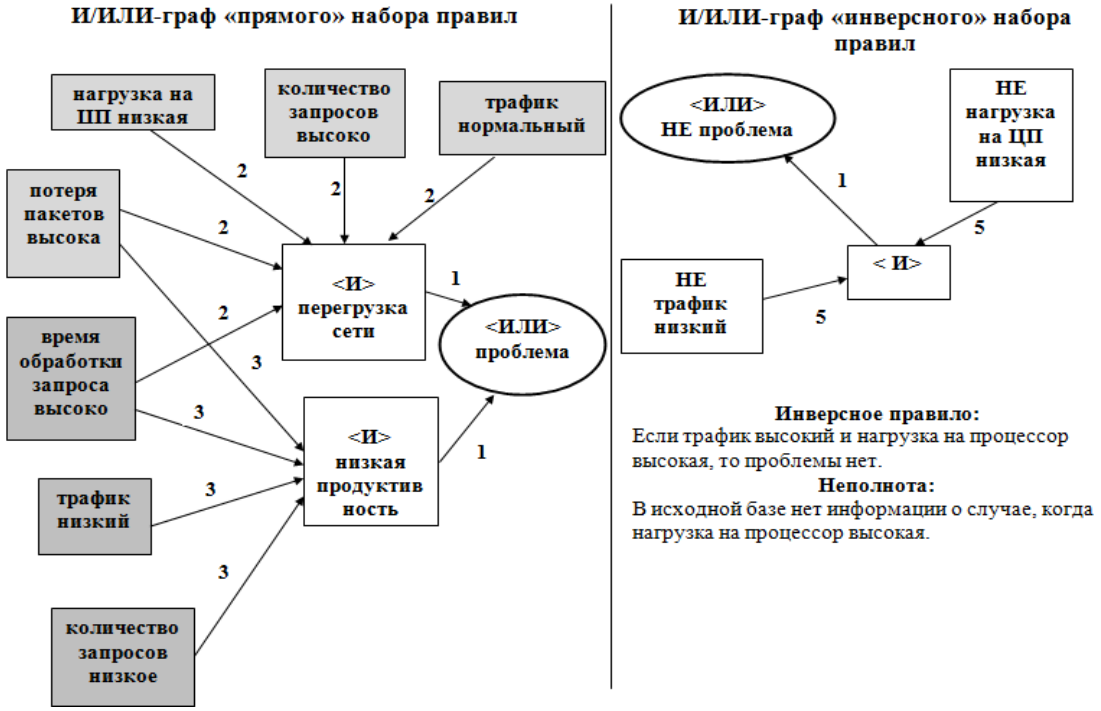

Рисунок 3.5 – Пример неполного набора правил

В левой части рис. 3.5 изображен граф исходного набора правил, в правой части – фрагмент графа для «инверсного» набора. Правило № 5 «инверсного» набора гласит, что если трафик не низкий (т.е. высокий) и нагрузка на

процессор не низкая (т.е. высокая), то проблемы нет. Очевидно, что это правило некорректно, так как высокие значения трафика и загрузки процессора означают перегрузку узла сети. Наличие такого правила означает, что исходная система правил, возможно, неполная. В данном случае неполнота заключается в отсутствии правил в исходном наборе для случая, когда нагрузка на процессор высокая.

3.2.3 Примеры проверки правил контроля электробезопасности на полноту

ПРИМЕР 1

Покажем применимость предложенного метода проверки правил на полноту на примере подгруппы правил по безопасной работе с электроустановками под названием «Правила, зависящие от внешних условий» (см. подраздел 4.3.2).

Обозначим посылки правил данной группы переменными, тогда правила можно будет представить в виде логических выражений:

1) a – помещение особо опасное;
2) b – помещение без повышенной опасности;
3) c – помещение с повышенной опасностью;
4) d - тип работ – переключение;
5) e – количество работников <2;
6) f – есть два человека: один человек с квалификацией >=2, а другой с квалификацией >=3;
7) g – квалификация каждого работника бригады < 3;
8) h – тип работ - снятие показаний приборов учета ;
9) i – U>1000в;
10) j - вид работ высотные.

В таблице 3.2 приведены этапы проверки и, для краткости, показаны лишь некоторые выводы. Выявлено, что при составлении правил по безопасности, зависящих от внешних условий, рассмотрены не все виды работ в сочетании с другими факторами (к примеру, монтаж/демонтаж оборудования и другие). Такая оплошность должна быть исправлена.

Всего же из 6 правил группы «Правила, зависящие от внешних условий» было получен «инверсный» набор из 17 правил.

Интерпретация 11 полученных правил, на основании которых выявлена неполнота, представлена ниже.

1) $b\bar{h} \vee b\bar{\imath} \vee b\bar{g} = b(\bar{h} \vee \bar{\imath} \vee \bar{g})$

Если помещение без повышенной опасности и тип работ – не снятие показаний приборов учета или U<1000в или квалификация одного из работников >= 3, то выполнять работы можно

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме снятие показаний приборов учета для безопасных помещений)

2) $\bar{e}\bar{g}c$

Если количество работников >=2 и квалификация каждого работника бригады >= 3, то неважно, что помещение с повышенной опасностью, работы выполнять можно.

Нужно уточнить в Нормативах, действительно ли при таких условиях можно выполнять работы, если нет, то добавить правило в исходный набор.

3) $\bar{e}\bar{g}\bar{d}$

Если количество работников >=2 и квалификация каждого работника бригады >= 3 и тип работ – не переключение, то работы выполнять можно

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ, кроме переключения

4) $\bar{d}\bar{e}a \vee \bar{d}\bar{e}c = \bar{d}\bar{e}(a \vee c)$

Если тип работ – не переключение и количество работников >=2, то даже при помещении с повышенной опасностью и с особой опасностью работы выполнять можно.

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме переключения при опасных помещениях)

5) $\bar{d}\bar{e}\bar{\imath}$

Если тип работ – не переключение и количество работников >=2 и U<1000в, то работы выполнять можно

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме переключения при низком напряжении <1000в)

6) $\bar{d}\bar{e}\bar{h}$

Если тип работ – не переключение и не снятие показаний приборов учета и количество работников >=2, то работы выполнять можно

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме переключения и снятия показаний приборов учета даже при количестве работников >=2)

7) $\bar{d}\bar{\jmath}a \vee \bar{d}\bar{\jmath}c = \bar{d}\bar{\jmath}(a \vee c)$

Если тип работ не переключение и не высотные, то даже при небезопасных помещениях работы выполнять можно.

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме переключения для небезопасных помещений).

8) $\bar{d}\bar{\jmath}\,\bar{h}$

Если тип работ не переключение и не высотные и не снятие показаний приборов учета, то работы выполнять можно.

Это правило совсем абсурдное, потому что значит, для остальных видов работ выполнять их всегда можно. Но, действительно, в основном наборе правил про них ничего не сказано.

9) $\bar{d}\bar{\jmath}\,\bar{\imath}\;\bar{d}\bar{\jmath}\bar{g} = \bar{d}\bar{\jmath}(\bar{\imath}\;v\;\bar{g})$

Если тип работ не переключение и не высотные, то если квалификация каждого работника бригады >= 3 или U<1000в, то работы выполнять можно.

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме переключения и высотных).

10) $\bar{\jmath}\,\bar{g}c$

Если вид работ не высотные и квалификация каждого работника бригады >= 3, то даже для помещений с повышенной опасностью работы выполнять можно.

Здесь неполнота состоит в том, что мы не знаем, что с остальными видами работ (кроме высотных).

11) $\bar{e}fa$

Если количество работников >=2 и есть два человека, чтобы один человек с квалификацией >=2, а другой с квалификацией >=3, то даже для особо опасных помещений работы выполнять можно.

Здесь неполнота состоит в том, что мы, возможно, не учли еще какие-то условия безопасной работы в особо опасных помещениях.

Таким образом, на основе интерпретации «инверсных» правил видно, каким образом эксперт может заметить неполноту знаний, заложенных в набор правил.

Таблица 3.2 – Проверка правил на полноту

Результаты поэтапной проверки	**Пример**
Булева формула для правил, которые зависят от внешних условий (работы выполнять нельзя)	$ade \vee ad\bar{e}f \vee cdg \vee bhig \vee cje \vee aje$
Булева формула для «инверсных» правил (работы выполнять нельзя)	$\overline{ade \vee ad\bar{e}\bar{f} \vee cdg \vee \bar{b}hig \vee cje \vee aje}$
Одно из «инверсных правил» после преобразования в ДНФ	$bh \vee b\bar{\imath} \vee b\bar{g} = b(h \vee \bar{\imath} \vee \bar{g})$
Обозначения посылок	**b** – помещение без повышенной опасности; **h**– тип работ - снятие показаний приборов учета; **i**– U>1000в; **g** – квалификация каждого работника бригады < 3;
«Инверсное» правило на естественном языке	Если помещение без повышенной опасности и тип работ - не снятие показаний приборов учета или U<1000в или квалификация одного из рабочих >= 3, то выполнять работы можно
Неполнота	В исходной базе нет информации о правилах для других видов работ и для напряжения U <1000в. Проверить нормативные документы на наличие таких правил.

ПРИМЕР 2

Рассмотрим пример, иллюстрирующий конструирование экспертом новых дополненных правил после просмотра набора «инверсных» правил.

Имеется набор из одного правила:

Если U≥1000в, и нет наблюдателя, и помещение опасное, то нельзя выполнять работы.

Представим правило в виде булевой формулы. Обозначим посылки правил:

a - U>1000В; b - нет наблюдателя; c - помещение опасное, а так же вывод w – можно выполнять работы.

Булевы формулы:

$$a \wedge b \wedge c \leftrightarrow \bar{w},\ \overline{a \wedge b \wedge c} \leftrightarrow w,\ \bar{a} \vee \bar{b} \vee \bar{c} \leftrightarrow w,\ \bar{a} \leftrightarrow w,\ \bar{b} \leftrightarrow w,\ \bar{c} \leftrightarrow w$$

Интерпретация полученных правил:

Если U<1000в, то можно выполнять работы. Правило не полное.

Если есть наблюдатель, то можно выполнять работы. Правило не полное.

Если помещение неопасное или особо опасное, то можно выполнять работы. Правило не полное.

Далее для каждого «проблемного» инверсного правил автоматически формируем все возможные способы дополнения полноты (см. рис 3.6):

для $\bar{a}$ = U<1000в	нет наблюдателя	можно выполнять работы
	есть наблюдатель	нельзя выполнять работы
	помещение опасное	
	помещение неопасное	
	помещение особо опасное	

Рисунок 3.6 – Возможные способы дополнения полноты

Далее пользователь выбирает те посылки, которые он считает необходимыми для полноты правил и формирует уже и «прямые» правила, и «инверсные». Например, он выбирает помещение особо опасное и нельзя выполнять работы, тогда автоматически формируется «прямое» правило:

Если U<1000в, и помещение особо опасное, то нельзя выполнять работы.

Затем подаем данное правило на вход SAT для проверки на противоречивость посылок, а также осуществляем проверку противоречивости посылок всех новых «прямых» правил.

3.3 Метод проверки достижимости в правилах состояний объектов контроля

Правила, то есть подграф, из которого недостижимы вершины состояний, целесообразно выявлять и удалять, так как для контроля состояний объектов они не используются. Такие правила визуализируются на И/ИЛИ-графе, а также в теории графов существуют методы нахождения компонент связности для ориентированных графов, и если в одной из компонент нет вершины для состояния объекта, то это говорит о том, что такая компонента бесполезна для контроля.

Однако для каждой группы правил контроля проверка достижимости из правила вершины состояния объекта упрощается, и метод состоит из 2х этапов: 1) получение из И/ИЛИ-графа группы булевых выражений для каждого правила, то есть формулы (2.1); 2) если в каком-либо булевом выражении правая часть импликации не будет содержать переменную, соответствующую вершине, отвечающей за состояние объекта контроля, то это будет означать, что существует компонента связности, в которой вершина для состояния объекта недостижима. На рис. 3.7 представлен пример компоненты связности графа, содержащего правила 5 и 4, из которого недостижима терминальная вершина "проблема".

Рисунок 3.7 – Пример недостижимости вершины состояния

3.4 Преобразование правил из И/ИЛИ-графа в текстовый вид

Начальной формой представления системы правил, в которой она находится в результате ввода пользователем, является И/ИЛИ граф. Этот граф выражает зависимость между отдельными утверждениями, каждое из которых может быть либо исходным условием, или выводом одного из правил. Таким образом, логическими единицами формы представления в виде И/ИЛИ графа есть отдельные утверждения и отношения между ними.

Алгоритмы контроля знаний, которые разрабатываются, оперируют правилами, анализируя или отдельное правило, или несколько правил. Именно поэтому для применения этих алгоритмов необходимо от утверждений и их отношений, заданных И/ИЛИ графом, перейти к правилам.

Рассмотрим пример И/ИЛИ графа, изображенного на рисунке 2.5. В И/ИЛИ-графе дуги обозначают отношение «условие - вывод». Эти дуги является направленными в сторону выводу, то есть исходят из условия правила и заходят в его заключение. Таким образом, каждая вершина графа с ненулевой степенью захода является выводом некоторого правила, а каждая вершина с ненулевым степенью исхода является условием некоторого правила. Если полустепень захода некоторой вершины равна нулю, это означает, что она не является выводом никаких правил, то есть, является входной для всей системы правил. Аналогично, если полустепень исхода вершины равна нулю, она не является условием никаких правил, то есть выводом системы правил. Рассмотрим пример И/ИЛИ-графа, изображенный на рисунке 3.7. В И/ИЛИ-графе дуги обозначают отношение «условие - вывод». Эти дуги является направленными в сторону вывода, то есть исходят из условия правила и заходят в его заключение. Таким образом, каждая вершина графа с ненулевой степенью захода является выводом некоторого правила, а каждая вершина с ненулевой степенью исхода является условием некоторого правила. Если полустепень захода некоторой вершины равна нулю, это означает, что она не является исходом никаких правил, то есть, является входной для всей систе-

мы правил. Аналогично, если полустепень исхода вершины равна нулю, она не является условием никаких правил, то есть выводом системы правил.

На основе приведенных выше соображений можно составить следующий алгоритм нахождения правил в И/ИЛИ-графе:

1. Среди всех вершин графа находим те, которые являются выводами системы правил (то есть такие, полустепени исхода, для которых равны нулю).

2. Каждую найденную вершину берем в качестве начальной для поиска правил.

3. Если начальная вершина имеет нулевую полустепень захода, поиск завершается. Если полустепень захода ненулевая, создаем правило, заключением которого является данная вершина, а каждая инцидентная ей вершина, связанная с ней входной дугой, является условием этого правила.

4. Добавляем созданное правило в список правил.

5. Для каждой инцидентной вершины, связанной с данной входной дугой, запускаем процедуру поиска, начиная с шага 2.

Данный алгоритм является рекурсивным. При ацикличности входного графа он завершается за время O (N), где N - количество вершин графа.

Рассмотрим работу алгоритма на примере графа, изображенного на рисунке 3.8.

1. Перебирая вершины графа, устанавливаем, что вершина «Вывод 1» является выводом системы правил1. Перебирая вершины графа, устанавливаем, что вершина «Вывод 1» является выводом системы правил

2. Берем вершину «Вывод 1» в качестве начальной для поиска.

3. Полустепеь захода вершины «Вывод 1» равна 2. Инцидентными являются вершины - «Условие 1» и «Вывод 2». Таким образом, есть правило №1.

4. Запускаем поиск для вершины «Условие 1». Ее полустепень исхода равна 0, поэтому поиск завершается.

5. Запускаем поиск для вершины «Вывод 2». Ее полустепень захода рав-

на 2, инцидентные вершины - «Условие 2» и «Условие 3». Таким образом, есть правило №2.

6. Запускаем поиск для вершины «Условие 2». Ее полустепень захода равна 0, поэтому поиск завершается.

7. Запускаем поиск для вершины «Условие 3». Ее полустепень захода равна 0, поэтому поиск завершается.

Схематическое изображение последовательности обхода графа изображенно на рисунке 3.8. В результате работы данного алгоритма получаем список из двух правил, который соответствует их действительному количеству.

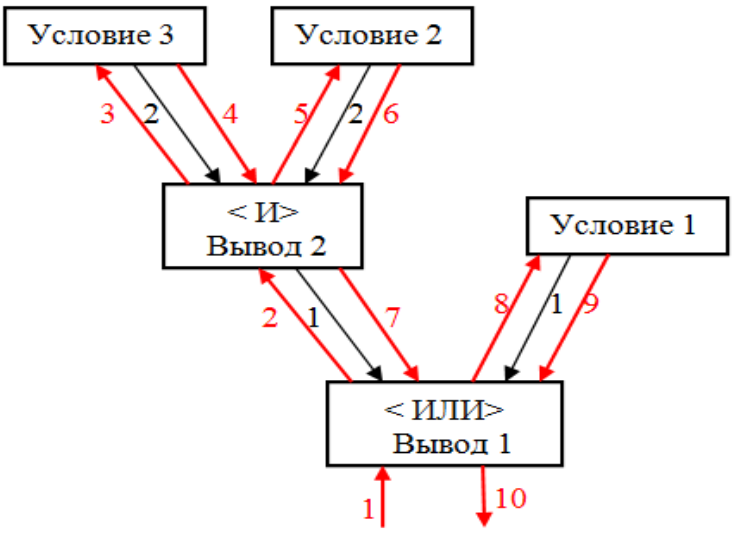

Рисунок 3.8 – Последовательность обхода И/ИЛИ-графа

3.5 Анализ сложности алгоритмов

Рассмотрим далее сложность алгоритмов, применяемых в рассмотренных моделях и методах.

При получении формул математической модели на основе булевых формул решаются задачи минимизации и приведения булевых формул в ДНФ. Эти задачи являются NP-полными.

При использовании метода проверки правил на полноту и метода проверки достижимости вершины-состояния используются формулы математической модели.

При использовании метода проверки на противоречивость посылок правил лишь на этапе 3 задаче SAT передается произвольная «инверсная» формула, а задача SAT для формул в общем случае является NP-полной. Алгоритм 1-ого и 2-го этапов не являются NP-полной задачей, так как задаче SAT передается ДНФ и для выполнимости формулы требуется только наличие хотя бы одной конъюнкции, не содержащей одновременно *x* и NOT *x* для некоторой переменной *x*.

Хотя и некоторые из рассмотренных алгоритмов и являются NP-полными, однако это не является ограничением для применения ее при анализе противоречивости посылок правил контроля по нескольким причинам:

– существуют эффективные алгоритмы решения задачи SAT, позволяющие достичь приемлемой эффективности для реальных задач (рассматривались такие задачи как планирование, составление расписаний, синтез решений);

– естественно решать такую сложную задачу на распределенных системах с большой вычислительной мощностью;

– существует приближенные алгоритмы, которые за полиномиальное время решают задачи SAT.

Но главная причина состоит в том, что такого рода проверки производятся не этапе проведения контроля объектов, когда с системой работает конечный пользователь, и где нужна высокая эффективность, а такая задача не требует чрезмерно высокой производительности в части скорости вычислений, так как решается только на этапах создания и корректировки правил.

3.6 Выводы по третьему разделу

В третьем разделе представлены разработанные методы проверки правил, основанные на моделях правил в виде И/ИЛИ-графа и булевых выражений для правил контроля.

1. Для проверки противоречивости посылок правил контроля предлагается использовать булевы формулы, соответствующие набору правил, и зада-

чу выполнимости булевых формул SAT. Предложен метод проверки противоречивости посылок правил контроля на основе неограниченной SAT, включающий 3 этапа проверки. Основные достоинства этого метода состоят в том, что проверка противоречивости правил производится автоматически и с учетом специфики предметной области контроля.

2. Предложен метод проверки правил контроля на полноту на основе визуализации «инверсных» правил и конструирования недостающих. Предложенный метод работает аналогично методу доказательства «от противного», при этом доказываемым утверждением является «прямой» набор правил, противоположное утверждение – это «инверсный» набор, а заведомо неверное утверждение, к которому приходят при доказательстве – это неполное с точки зрения эксперта «инверсное» правило. Рассмотрены примеры проверки правил контроля на полноту на основе предложенного метода.

3. Предложен метод для проверки *достижимости* в правилах *состояний* объектов контроля. Для группы правил контроля проверка достижимости из компоненты связности вершины состояния объекта упрощается, и метод состоит из 2х этапов: 1) получение из И/ИЛИ-графа группы булевых выражений для каждого правила, то есть формулы (2.1); 2) если в каком-либо булевом выражении правая часть импликации не будет содержать переменную, соответствующую вершине второго типа, отвечающую за состояние объекта контроля, то это будет означать, что существует компонента связности, в которой вершина для состояния объекта недостижима.

4. Предложен алгоритм автоматического преобразования правил из И/ИЛИ-графа в тестовый вид и наоборот.

5. Проведен анализ сложности алгоритмов, используемых в предложенных методах, и сделан вывод о том, что часть алгоритмов являются NP-полными, что не является ограничением для их применения, приводятся аргументы, а главный состоит в том, что такого рода проверки происходит не на этапе консультации, когда с правилами работает конечный пользователь, и где нужна высокая эффективность, а на этапе создания и корректировки пра-

вил, а такая задача не требует чрезмерно высокой производительности вычислений, так как решается лишь при создании и корректировке правил.

Результаты исследований раздела 3 опубликованы в работах автора [22, 72].

РАЗДЕЛ 4
РАЗРАБОТКА И АПРОБАЦИЯ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ СОЗДАНИЯ И СОПРОВОЖДЕНИЯ ЗНАНИЕОРИЕНТИРОВАННЫХ СИСТЕМ КОНТРОЛЯ

4.1 Технология создания и сопровождения знаниеориентированных систем контроля

Базовая информационная технология для создания *знаниеориентированных систем* содержит следующие этапы [35]:

– этап получения знаний, включающий анализ предметной области, понятий и их взаимосвязей, определения целесообразности применения знаниеориентированной системы в выбранной области;

– этап структурирования знаний, разрабатывается неформальное наглядное описание знаний о предметной области в виде графа, таблицы, диаграммы или текста, которое отражает основные концепции и взаимосвязи между понятиями предметной области;

– этап формализации – это организация и представление знаний в виде, удобном для работы знаниеориентированной системы;

– этап реализации, при котором создается один или несколько прототипов знаниеориентированной системы, решающих поставленные задачи;

– этап тестирования, при котором оценивается выбранный способ представления знаний знаниеориентированной системы в целом.

– этап сопровождения – процесс улучшения, оптимизации и устранения дефектов знаниеориентированной системы после передачи в эксплуатацию.

Предлагается прикладная информационная технология для разработки систем контроля, основанная на вышеприведенной базовой, с использованием инструментального средства в виде редактора правил для создания, проверки и поддержки правил контроля. Технология основана на синтаксической структуре знаний для контроля, на модели представления знаний в виде правил – продукций, на предложенной модели, методах и способах проверки

качества правил контроля (разделы 2 и 3), а также на возможности и необходимости частичного переноса этапа тестирования знаний на более ранние этапы разработки (см. таблица 4.1).

Таблица 4.1 – ИТ для создания и сопровождения знаниеориентированных систем контроля

Базовая ИТ	Прикладная ИТ для систем контроля
Этап 1. Получение знаний – анализ ПрО – выделение понятий – целесообразность разработки знаниеориентированной системы	– составление словаря ПрО; – выявление параметров объектов контроля; – выбор отслеживаемых состояний; – определение целесообразности разработки в соответствии с критериями из базовой ИТ
Этап 2. Структурирование –наглядное представление и декомпозиция понятий и взаимосвязей в виде графа, таблицы или текста	– построение иерархии понятий для ПрО, разбиение на терминальные и промежуточные – распределение знаний на группы по состояниям объекта, используется одна группа для каждого состояния; при большом объеме группы разделение её на подгруппы – построение правил на естественном языке и в виде И/ИЛИ-графа с возможностью преобразования из одной формы в другую (в редакторе правил) – автоматизированная проверка правил на непротиворечивость, полноту, избыточность, достижимость состояний (в редакторе правил)
Этап 3 Формализация знаний – представление знаний в форме, которую использует машина вывода	– автоматическое преобразование каждой группы (подгруппы) правил в язык продукционного программирования с учетом уменьшения времени логического вывода за счет оптимального назначения приоритетов правилам
Этап 4 Кодирование	– выбор готовой машины вывода либо разработка собственной – доработка каждой группы (подгруппы) в соответствии с полученными правилами на этапе формализации – объединение групп (подгрупп) правил в общую модульную структуру с учетом частоты ошибок и срабатывания в двух режимах: 1) до первого несоответствия, 2) полная проверка – интеграция БЗ и машины вывода с двумя компонентами системы контроля: – первым – для настройки режимов и получения параметров объекта контроля, – второй – для выдачи результатов работы и организации графического интерфейса с пользователем
Этап 5 Тестирование	– отладка каждой группы правил: всех условий для аварийных состояний, а также условий нормальной работы – отладка модульной структуры, полученной при объединении групп (подгрупп) правил – интеграционное тестирование системы контроля
Этап 6 Сопровождение	– изменение правил контроля (в редакторе правил)
Этапы 1-6 выполняются итерационно, пока система не будет удовлетворять требованиям пользователя	

Этап 1 Получение знаний о предметной области контроля

1.1 Автоматизированное составление словаря предметной области с использованием инструментальных средств [52]. Для создания качественного

словаря необходимо подобрать такой набор текстов, который бы содержал максимальное количество специфических для данной предметной области терминов. Однако в большинстве случаев тексты содержат большое количество служебных слов, терминов из других или смежных предметных областей [53], поэтому для составления словаря целесообразно привлечь эксперта в данной предметной области. Такой словарь поможет избежать большого количества ошибок на всех последующих этапах при построении структуры знаний, в особенности, если подключить его и возможности работы с ним к инструментальному средству для составления правил. Эксперт может просматривать термины словаря, выбирать необходимые и добавлять в правила. Такой словарь предметной области помогает избежать большого количества ошибок при составлении правил экспертной системы, а также сокращает время их разработки.

В [36] предлагается метод автоматизированного создания словаря терминов предметной области, который основывается на синтаксическом и семантическом анализе текстов, и на основе этого метода создана свободно распространяемая программа, реализующая построение словаря, позволяющая просматривать выделенные слова и словосочетания, число их вхождений, частотность, а также записывать результаты в .xml–файл.

1.2 Определение входных данных, то есть параметров системы и их возможных значений; причем, значения используются в дальнейшем в качестве посылок правил.

1.3 Выявление отслеживаемых значений параметров, для которых проводится контроль, а также условий их выхода за пределы допустимого.

1.4 Определение целесообразности использования знаниеориентированной системы для контроля в определенной ПрО.

Этап 2 Структуризация знаний для контроля

2.1 Построение понятийной структуры знаний в виде иерархии параметров для контроля и их значений. Разбиение понятий, то есть параметров и их значений каждой группы для выделения так называемых конечных (терми-

нальных), из анализа которых напрямую делается вывод о выходе отслеживаемого состояния за допустимые пределы, и промежуточных, с помощью которых за несколько шагов можно прийти к окончательному выводу.

2.2 Разбиение знаний на группы, каждая группа соответствует одному отслеживаемому состоянию объекта. Таким образом, при построении правил в каждой группе окажутся правила, описывающие одну ситуацию, для которой производится контроль. В свою очередь, если группа большая, то она может быть разбита на подгруппы независимых либо слабо зависимых друг от друга знаний. Такая модульность позволяет визуально проще создавать БЗ и поддерживать ее.

2.3 Построение функциональной структуры предметной области контроля, моделирующей рассуждения эксперта, в виде правил для каждой группы (подгруппы) на естественном языке в текстовом виде либо виде И/ИЛИ-графа с возможностью автоматического перехода из одного вида в другой, а также сохранения/загрузки правил. Для этого используется такое автоматизированное инструментальное средство как редактор правил.

2.4 Автоматизированная проверка правил на непротиворечивость, полноту, достижимость состояний, цикличность (в редакторе правил):

– проверка каждого правила, а также всего набора правил на непротиворечивость с использованием задачи выполнимости булевых формул (SAT); таким образом можно определить правила, которые никогда не смогут выполниться либо будут выполняться всегда: и первый, и второй случай недопустимы и говорят о противоречиях в системе правил;

– проверка правил на неполноту на основе автоматического построения и визуализации «инверсных» правил, описывающих в каких случаях отслеживаемое состояние объекта в норме;

Помощь в конструировании новых правил проводится таким образом: для каждого проблемного правила система показывает все возможные способы дополнения; эксперт формирует и «прямые» и «инверсные» правила, добавляя с его точки зрения правильные предлагаемые посылки и отмечает, что

при таких посылках будет происходить: авария или нормальное состояние объекта; автоматически из исправленных «инверсных» правил формируются «прямые»; автоматически проверяется противоречивость посылок полученных «прямых» правил; полученные «прямые» правила предъявляются эксперту, в случае противоречий выдается сообщение. Такая работа по конструированию правил проводится итерационно, пока эксперт не будет уверен в полноте построенного набора правил.

– проверка достижимости из правил вершины состояния объекта состоит из 2-х этапов: 1) получение с И/ИЛИ-графа группы булевых выражений для каждого правила, то есть формулы (2.1); 2) если в каком-нибудь булевому выражении правая часть импликации не содержит переменную, соответствующую состоянию объекта контроля, то это будет означать, что существует правило, в котором вершина для состояния объекта недостижима.

Для проверок используется автоматизированное инструментальное средство, как редактор правил, позволяющий проводить преобразования булевых выражений автоматически.

Этап 3. Формализация знаний для контроля

3.1 Автоматическое преобразование правил, представленных в виде И/ИЛИ-графа, в текстовый вид на естественном языке с помощью редактора правил, просмотр разработчиком БЗ, при необходимости, корректировка в текстовом виде и обратное автоматическое преобразование в И/ИЛИ-граф. Таким образом, можно «увидеть» правила еще с одной точки зрения.

3.2 Преобразование правил, представленных в виде И/ИЛИ-графа, в язык продукционного программирования для конкретной платформы.

Чтобы уменьшить число срабатываемых правил при контроле, присваиваем более высокие приоритеты правилам на основе следующих соображений:

– раньше должны выполняться те правила, которые содержат посылки, общие для многих правил, так как такие посылки наиболее значимые при кон-

троле, и, таким образом, если они выполняются, то при этом больше вероятность того, что сработает какое-то правило и контроль окончится;

– раньше должны выполняться те правила, которые имеют меньшее число посылок, так как в этом случае будет меньше проверок.

Этап 4 Реализация знаниеориентированных систем контроля

4.1 Выбор готовой машины вывода или разработка собственной; настройка машины вывода.

4.2 Программирование каждой группы (подгруппы) правил, в соответствии с полученными правилами на этапе формализации 3.2.

4.3 Объединение групп (подгрупп) правил в общую модульную структуру для возможности их срабатывания в заданной последовательности.

Очередность выполнения модулей назначается на основе частоты возникновения ошибок (раньше запускается та группа, для которой чаще всего при контроле фиксируются нарушения).

Для обеспечения оптимального объединения модулей знаниеориентированной системы выделены различные режимы срабатывания модулей и на их основе предложены два режима объединения. В *первом режиме* (быстром) решение о том, что отслеживаемое состояние объекта выходит за допустимые пределы выносится по первой же группе правил, которые нарушены, и дальше модули не запускают. Второй режим работы знаниеориентированной системы предусматривает срабатывание всех модулей, что позволяет выявить все проблемы.

4.4 Интеграция БЗ и машины вывода с другими компонентами системы контроля путем встраивания их в существующую систему контроля дополнением их двумя компонентами:

– первым – для настройки режимов и получения параметров с объекта контроля;

– вторым – для выдачи результатов работы и организации графического интерфейса с пользователем.

Этап 5. Отладка знаниеориентированных систем контроля.

5.1 Отладка для каждой группы правил. Выполняем столько тестов, сколько правил в группе, таким образом тестируются все условия выхода за пределы исследуемого состояния объекта. Далее выполняем тесты, проверяющие случаи, когда не происходит выход состояние объекта за допустимые пределы.

5.2 Отладка общей модульной структуры правил. Тестируем каждое управляющее правило и все их возможные сочетания.

5.3 Интеграционное тестирование системы контроля.

Этап 6. Сопровождение включает, в первую очередь, изменение правил контроля. Предполагается оперативное внесение изменений в правила, преимущественно экспертом (в редакторе правил).

Некоторые подпункты этапов 1-6 могут выполняться в разной последовательности, а зачастую и итерационно. Удобно, например, инженеру по знаниям вместе с экспертом сначала представить, какие могут быть группы правил, потом для каждой группы определить параметры и их значения, затем построить правила. Далее – произвести проверки правил и при исправлении неточностей и ошибок вернуться на предыдущие шаги.

В практике создания ЭС последовательность этапов может быть незначительно изменена, к примеру, сначала строят правила, потом выбирают из них параметры и их значения, одновременно, корректируя правила, а потом правила разделяют на группы и производят проверку.

В результате просмотра визуализированного графа инженер по знаниям вместе с экспертом могут просмотреть правила, выявить несоответствия и логические ошибки в построенных ранее правилах, параметрах и их значениях, и их исправить.

Аналогично, этапы 4 и 5 выполняются итерационно как и при разработке любых програмных систем.

Предлагаемая ИТ рассчитана в полном объеме на создание БЗ, анализирующих четкие параметры объекта контроля. Однако, если параметры нечеткие [48], то большая часть этапов ИТ выполняется полностью, один пункт на

этапе структурирования знаний, а именно, проверка нечетких посылок правил на противоречивость может быть выполнена заменой на время проверки нечетких посылок на четкие, так как задача SAT не работает с нечеткими булевыми формулами. Набор нечетких условий правил представляется в И/ИЛИ-графе дополнительным заданием нечетких переменных для вершин, включая соответствующие нечеткие множества, и для остальных проверок, использующих в булевых выражениях логические операции НЕ, И, ИЛИ, применяются существующие аналоги нечетких операций и метод минимизации нечетких булевых выражений. Существуют инструментальные средства для кодирования знаниеориентированных систем с четкими правилами, в частности, Clips, а также FuzzyClips для нечетких правил [51], и правила контроля, полученные на этапе 2, конвертируются в продукционные языки для этих систем.

4.2 Инструментальное средство для создания и поддержки правил знаниеориентированных систем контроля

На основе предложенной модели для визуализации и интерактивной работы с правилами, а также методов проверки качества базы был создан редактор правил и использован при разработке БЗ для двух предметных областей контроля: безопасной работе с электроустановками и работе с компьютерной сетью.

Редактор правил является инструментальным средством поддержки прикладной информационной технологии для создания знаниеориентированной системы контроля, приведенной в 4.1, поэтому функции системы обусловлены содержанием этапов, где редактор задействован: на этапе 2 – это построение правил на естественном языке и в виде И/ИЛИ-графа с возможностью конвертации, на этапе 3 – автоматизированная проверка правил на непротиворечивость, полноту, избыточность, цикличность, достижимость состояний, а также преобразование правил в язык

продукционного программирования (получение «заготовки») с учетом эффективности логического вывода.

Диаграмма вариантов использования [45] представлена на рис. 4.4.

Для разработки системы [37] был избран объектно-ориентированный подход. Анализируя представленное в разделах 1-3 описание предметной области контроля, можно выделить следующие сущности: база знаний экспертной системы, булево выражение, правило, тип правила, И/ИЛИ-граф, вершина графа, дуга графа.

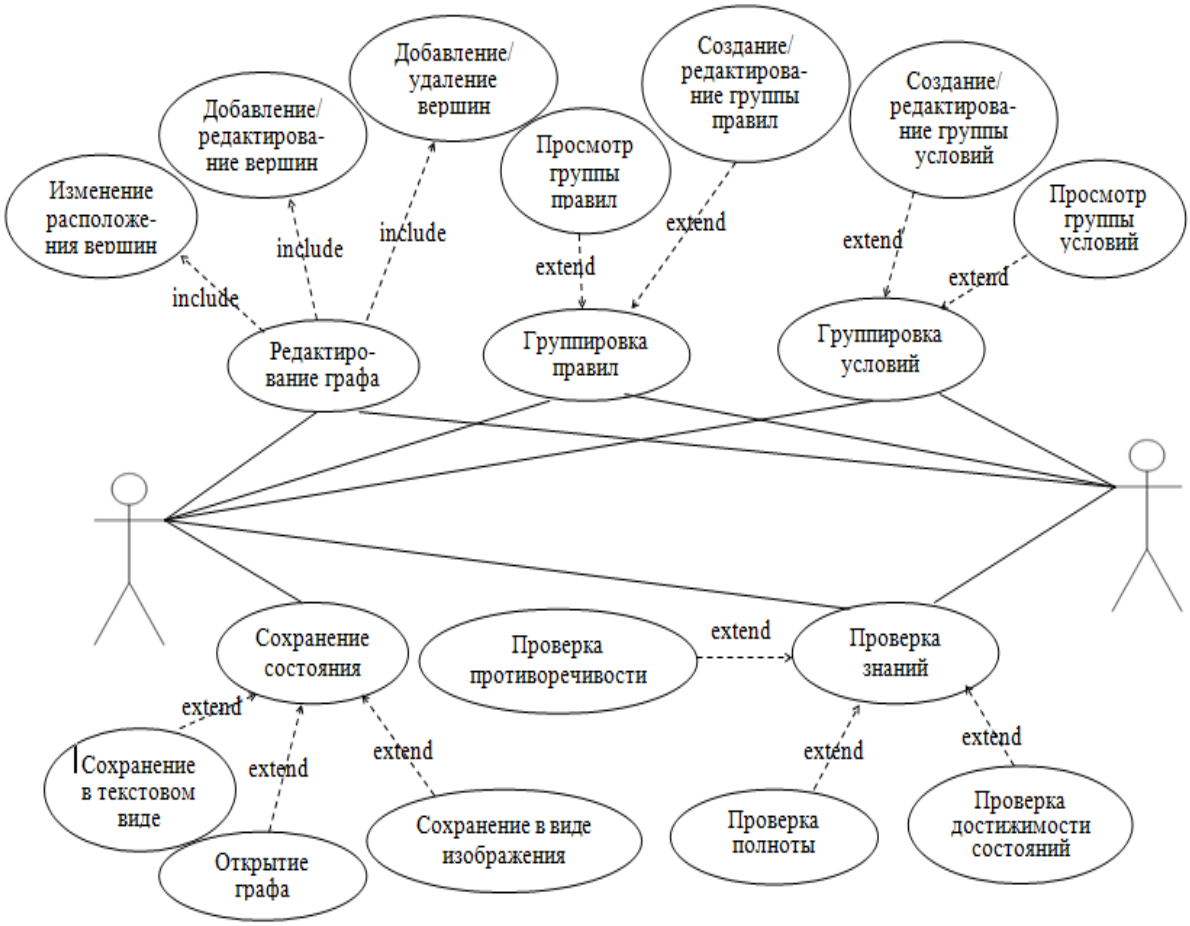

Рисунок 4.4 – Диаграмма вариантов использования для редактора правил

Для разработки системы был избран объектно-ориентированный подход. Анализируя представленное в разделах 1-3 описание предметной области контроля, можно выделить следующие сущности: база знаний экспертной системы, булево утверждение, правило, тип правила, И/ИЛИ-граф, вершина графа, дуга графа.

Получив сведения о сущностях предметной области, их атрибуты и определенные для них операции получаем модель предметной области в виде диаграммы классов (рис. 4.5).

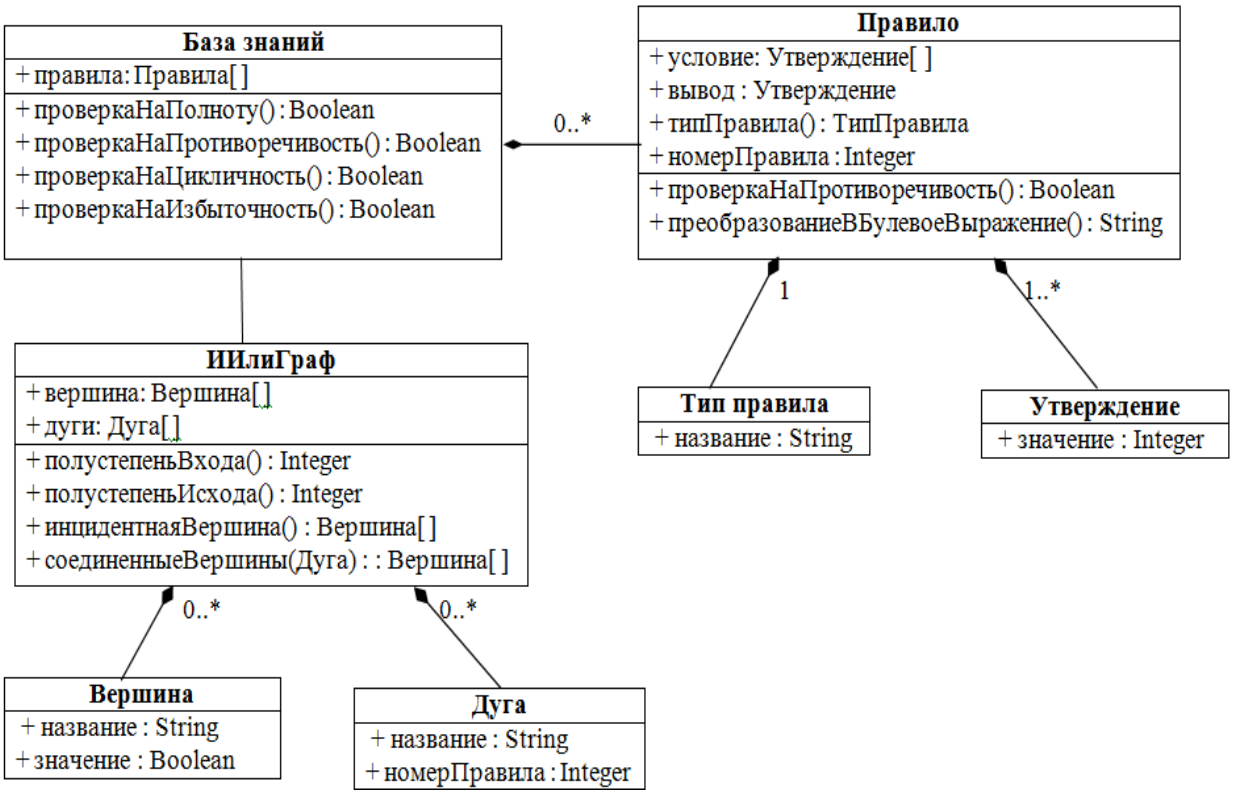

Рисунок 4.5 – Диаграмма классов для редактора правил

Далее рассмотрены шаблоны проектирования, использованные при разработке редактора правил.

Наблюдатель (Observer). Данный паттерн применяется в разрабатываемом редакторе правил при предоставлении графического интерфейса пользователю, для осуществления реакции программы на изменения состояния, обусловленные действиями пользователя, которые, как правило, являются асинхронными.

Посетитель (Visitor). Поскольку класс для графа определен во внешней библиотеке, его изменения невозможны, а использования шаблона Посетитель является единственной возможностью для добавления новых операция к этому классу.

Адаптер (Adapter). В разрабатываемом редакторе правил данный шаблон активно используется для преобразования объектов (например, вершин графа) в вид, который позволяет отображать их в графическом виде.

Шаблонный метод (Template Method). Применение данного шаблона позволяет значительно упростить структуру дерева наследования для этих классов и обеспечить повторное использование кода.

При разработке редактора правил был использован следующий набор инструментов: разработка ведется на языке Java [41] в среде IntelliJ IDEA [10] с использованием Git для контроля версий, JUNG для работы с графами и Swing в качестве библиотеки графического интерфейса пользователя.

Инструментом в языке Java, который обеспечивает поддержку модульности, есть пакеты. Классы разрабатываемой структуры [38] были сгруппированы в пакеты следующим образом: пакет *com.achernenko.editor* является основным пакетом программы, он, в свою очередь, состоит из следующих пакетов: *bool_operations* содержит классы, реализующие операции над булевыми выражениями; *model* содержит классы, являющиеся моделями сущностей предметной области, и содержит такие пакеты: *bool* – классы, моделирующие дерево булева выражения; *graph* – классы, моделирующие И/ИЛИ-граф и базу правил. *ui* содержит классы, предназначенные для графического интерфейса пользователя, и содержит такие пакеты: *graph* – классы, отвечающие за отображение И/ИЛИ-графа; *plugins* – классы, реализующие возможности редактирования И/ИЛИ- графа.

На рис. 4.6 изображен скриншот редактора правил, ставшего результатом разработки.

В левой части окна находится область редактирования графа, в правой – элементы управления, позволяющие изменять свойства вершины, группы условий, правил, а также поле, в которое выводятся сообщения о событиях (лог). Под областью для редактирования графа находятся кнопки, позволяющие запустить проверки качества базы правил. Основные операции продублированы пунктами в главном меню, а также клавиатурными сокращениями.

Для того, чтобы пользователь смог создать или редактировать группы исходных условий, т. е. вершин И/ИЛИ-графа, система отображает средства редактирования, находящиеся с правой стороны редактора правил. Пользователь вводит название группы, выбирает цвет вершин, а так же вершины графа, которые должны входить в группы и сохраняет внесенные изменения, так же в диалоговом окне можно просмотреть внесенные изменения.

Есть возможность создания группы правил, которая является подмножеством общей системы правил, для улучшения их восприятия и отладки. Для создания или редактирования группы исходных условий система отображает средства редактирования. Пользователь вводит название группы, выбирает вершины графа, которые должны входить в группы. Система выбирает дуги, которые относятся к выбранным вершинам. Пользователь может сохранить внесенные изменения, а также просмотреть внесенные изменения в редакторе правил.

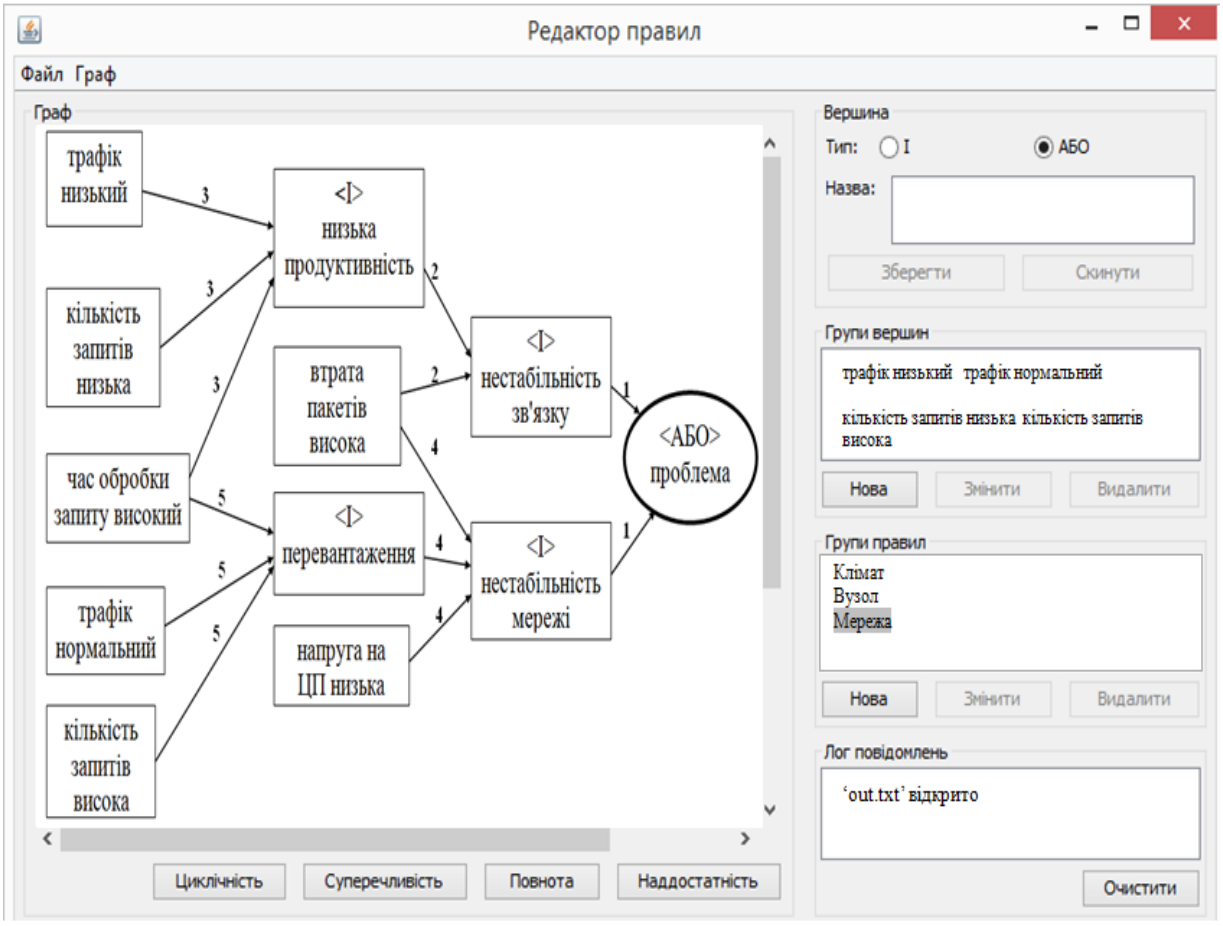

Рисунок 4.6 – Редактор правил

4.3 Разработка знаниеориентированных систем контроля

В данном разделе способы построения систем контроля рассматриваются на примере двух ЭС: для контроля выполнения требований по безопасной эксплуатации электроустановок и принятия решений о допустимости проведения работ; для контроля и мониторинга компьютерной сети. Целью первой знаниеориентированной системы является улучшение безопасности работы на электрооборудовании. Лицо, принимающее в ней решение,– это диспетчер, который следит за техникой безопасности. Для второй системы ЛПР – системный администратор, который следит за работоспособностью и эффективностью функционирования компьютерной сети. Первая разрабатываемая система принятия решений должна повысить безопасность на предприятии путем предоставления рекомендаций диспетчеру. В свою очередь, возможность автоматизированного выявления неполадок сети с использованием знаниеориентированной системы полезна, поскольку позволяет снизить нагрузку на системного администратора, также повысить вероятность раннего обнаружения проблем и их исправления. Возможность определения собственных правил, в отличие от существующих систем, в которых правила встроены, и их изменения требуют переработки всей системы, позволяет улучшить качество автоматизации.

Рассмотрим далее этапы разработки ЭС в соответствии с информационной технологией, предложенной в разделе 4.1.

4.3.1 Получение знаний

Создание обоих ЭС базируется как на знаниях экспертов, так и на текстах нормативных документов. Основные этапы создания обусловленные предложенной ИТ.

С помощью интервьюирования экспертов получена необходимая информация. В случае выявления недостатков знаний, полученных от эксперта, проблемы устранялись путем использования письменных источников.

Из текстов, предложенных экспертами, с помощью свободно распространяемого инструментального средства автоматически получены словари терминов.

Для извлечения знаний о контроле по безопасной эксплуатации электроустановок целесообразно использовать профдокументы [58], а также знания эксперта в области безопасной работы с электроустановками. Для предметной области «Безопасная работа с электроустановками» были подобраны тексты, взятые из нормативных документов. Тексты были разбиты на составляющие, соответствующие разделам документов: «Подготовка», «Ответственность», «Требования», «Порядок заказа», «Мероприятия», «Обслуживание», «Выполнение работ», «Надзор», «Оформление перерывов», «Состав бригады», «Перевод бригады», «Окончание работ». Каждая часть в дальнейшем стала соответствовать подгруппе правил ЭС, и для каждой автоматически был составлен словарь терминов (см. рис. 4.7), на котором представлен словарь для одной из частей.

Слово	Число вхождений	Частотность слова
работа	34	22,398
напряжение	30	19,763
часть	22	14,493
провод	15	9,881
средства	13	8,564
электроустановка	12	7,905
работник	12	7,905
трос	12	7,905
стремянка	11	7,246
выполнение	10	6,588
применение	9	5,929
снятие	8	5,270
земля	8	5,270
место	8	5,270

Слово	Число вхожде...	Частотность с...
[токопроводящий] часть	12	0,791
[изолировочный] часть	5	0,329
[диэлектрический] перчатки	4	0,264
[электрозащитный] средства	4	0,264
[технологический] карта	3	0,198
[случайный] прикосновение	2	0,132
[рабочий] место	2	0,132
[лаковый] покрытие	2	0,132
[изолировочный] средства	2	0,132
[опорный] часть	2	0,132
[металлический] площадка	2	0,132
[коммутационный] аппарат...	2	0,132
[защитный] очко	2	0,132
[изолировочный] клещ	2	0,132

Рисунок 4.7 – Словарь данных

Параметры объекта и их значения представлены на рис. 2.2, выявлены два состояния: «работы выполнять нельзя», «работы выполнять можно».

Для извлечения знаний **о контроле и мониторинге компьютерных сетей** целесообразно придерживаться следующей стратегии: в качестве основного источника знаний выбрать тексты, описывающие работу компьютерных сетей; обратить внимание на такие аспекты, как основные неполадки, возникающие в компьютерных сетях, их последствия, а также пути их обнаруже-

ния; при возникновении вопросов привлечь экспертов в области администрирования компьютерных сетей; формой взаимодействия с экспертом выбрать интервью.

Были выбраны тексты «Неполадки в узлах сети», «Неполадки сети от внешних факторов», «Общие неполадки в сети» и по ним были составлены словари терминов предметной области. Также была выделена информация о неполадках в сети, таких как перегрев**,** неполадки системы охлаждения, перегрузка узла сети, перегрузка сети, нестабильность канала связи. Далее представлено описание неполадок.

Перегрев. Перегревом является превышение любым из компонентов компьютерной системы оптимальных значений рабочей температуры из-за чрезмерной нагрузки или из-за недостаточности охлаждения. О первом случае свидетельствует значительная нагрузка на процессор, оперативную память и диск при том, что интенсивность работы системы охлаждения (опосредованно может быть определена по частоте вращения вентиляторов в оборотах в минуту) является высокой. Вторая причина может быть идентифицирована с высокой температурой окружающей среды.

Неполадки системы охлаждения. Отдельно можно выделить ситуацию, в которой режим работы системы охлаждения не соответствует температурному режиму. Первой разновидностью такой ситуации является высокая интенсивность работы системы охлаждения при нормальной температуре системы, вторая разновидность – наоборот. Причиной этого является неправильная настройка системы охлаждения.

Перегрузка узла сети – ситуация, при которой нагрузка превышает возможности узла. Факторами, влияющими на количество и сложность задач, выполняемых узлом, является производительность процессора, объем оперативной памяти, а также производительность дисковой подсистемы. Независимо от типа и количества выполняемых задач, может произойти ситуация, при которой один из ресурсов системы исчерпывается, в результате может произойти перегрев (если система охлаждения работает недостаточно эффек-

тивно), производительность системы может снизиться (вплоть до нулевой отметки, когда узел сети перестает выполнять свои функции – так называемая ситуация отказа в обслуживании). Часто перегрузки является следствием DOS-атаки (англ. Denial of service, отказ в обслуживании), когда злоумышленники делают большое количество запросов к узлу сети, намеренно исчерпывая ресурсы системы. Поскольку ситуация перегрузки характеризуется почти полным исчерпанием ресурсов системы, ее можно обнаружить с высокими значениями загрузки процессора, оперативной памяти, а также по высокой интенсивностью обращений к дисковой подсистеме; увеличивается время обработки запроса и объемы трафика к узлу сети.

Перегрузки сети (англ. Network congestion) – особая ситуация перегрузки, когда нагрузка на узлы сети является относительно небольшой, но количество трафика слишком большое, такое, что значительно превышает возможности сетевого интерфейса, появляются задержки обработки сетевых запросов, может стать невозможным создание новых сетевых соединений, что приводит к отказу в обслуживании. Признаками перегрузки сети есть большие объемы трафика к узлам, потеря части пакетов. Если процент потерянных пакетов достигает критических значений, возникает ситуация коллапса, когда при интенсивном трафике количество полезных пакетов очень низко (трафик при этом продолжает генерироваться через попытки повторного направления потерянных пакетов). Перегрузка сети может быть обнаружена с высокими объемами трафика и значительной потерей сетевых пакетов.

Нестабильность канала связи. Для сохранения устойчивого функционирования сети нужен устойчивый канал связи. В случае применения беспроводной связи устойчивость связи снижается с расстоянием, а также из-за наличия помех. Если канал связи проводной, то провод может быть поврежден, в случае чего связь прерывается вообще. В любом случае, снижение качества связи приводит к снижению производительности. Нестабильная связь может вызвать потерю сетевых пакетов. Потерянные пакеты должны быть отправлены повторно, из-за чего увеличивается трафик, а также время на обработку

запроса. Признаками нестабильности канала связи является увеличение времени обработки узлом сетевого запроса и потеря сетевых пакетов.

Целесообразность использования знаниеориентированных систем для контроля и мониторинга двух ПрО обоснована в подразделе 1.3.3.

4.3.2 Структурирование знаний

Так как для ПрО по безопасной работы с электроустановками аварийное состояние одно – «нельзя выполнять работы», то все правила принадлежат одной группе; аналогично и для ПрО, связанной с работой компьютерной сети, также аварийное состояние одно – «проблема в сети», и все правила принадлежат одной группе. Решено группу для каждой ПрО разбить на подгруппы (5 и 3 подгруппы соответственно) в связи с большим объемом правил, а также естественной их разбивкой согласно рекомендациям эксперта, причин аварийного состояния и разделов нормативов. Построена понятийная структура ПрО, содержащая параметры объектов, значения параметров и связи между ними. Для каждой подгруппы созданы правила на естественном языке и параллельно И/ИЛИ-графы. Проведено построение, а также проверка правил итеративно с использованием разработанного редактора правил.

Фрагменты понятийной структуры предметной области «Безопасная работа с электроустановками» по подгруппам «Наряды, распоряжения, инструктаж» и «Правила, зависящие от внешних условий» представлены на рис. 4.8. Все параметры являются входными для системы, однако они разделяются на параметры верхнего уровня (промежуточные), например, как «ранг работ», а также на терминальные, например, «наряд оформлен», распоряжение оформлено». Такое разделение в дальнейшем полезно при составлении правил: если ранг работ – по наряду, то дальше нужно проверять, оформлен ли наряд. Параметры «наличие руководителя», «наличие наблюдающего» являются одновременно и промежуточными, и терминальными.

Построены следующие подгруппы правил: **«Работы по наряду и распоряжению, инструктаж», «Правила, зависящие от внешних условий»,**

«Работа с приборами учета», «Защитные средства», «Работы в охранной зоне электрических сетей» [73].

а) **наряды, распоряжения, инструктажи**

б) **правила, зависящие от внешних условий**

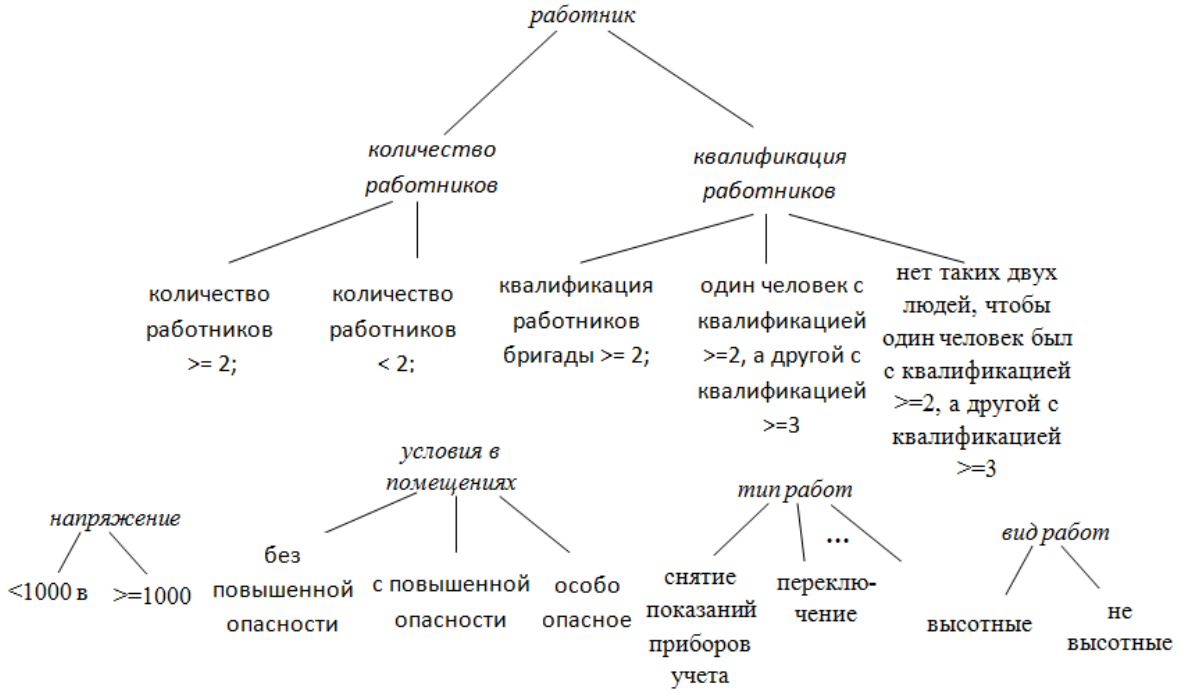

Рисунок 4.8 - Иерархия понятий ПрО по безопасной работе с электроустановками

Ниже показаны правила и графы (рис. 2.7 и 4.9) для двух подгрупп, остальные приведены в приложении А.

Подгруппа правил «Работы по наряду и распоряжению, инструктаж»

Если U>1000в, работа должна выполняться в порядке технической эксплуатации и распоряжение на работу не оформлено и наряд на работу не оформлен, то работу выполнять нельзя.

Если U>1000в и работа должна выполняться по наряду, и нет наблюдающего, то работу выполнять нельзя.

Если U>1000в и работа должна выполняться по наряду, и нет руководителя, то работу выполнять нельзя.

Если работа должна выполняться по наряду, и наряд на работу не оформлен, то работу выполнять нельзя.

Если работа должна выполняться по распоряжению, и нет наблюдающего, то работу выполнять нельзя.

Если работа должна выполняться по распоряжению, и распоряжение на работу не оформлено, то работу выполнять нельзя.

Если не для каждого работника есть отметка о целевом инструктаже, то работу выполнять нельзя.

Если не для каждого работника дата следующей проверки по охране труда<WorkEnd_time,то работу выполнять нельзя.

Подгруппа правил «Правила, зависящие от внешних условий»

Если помещение особо опасное и если тип работ – переключение, и количество работников >= 2, и нет таких двух людей, чтобы один человек был с квалификацией >=2, а другой с квалификацией >=3, то работу выполнять нельзя.

Если помещение с повышенной опасностью и если тип работ - переключение, и квалификация каждого работника бригады < 3, то работу выполнять нельзя.

Если помещение особо опасное, и если тип работ – переключение, количество работников < 2, то работу выполнять нельзя.

Если помещение с повышенной опасностью или помещение особо опасное и вид работ – высотные, и количество работников <2, то работу выполнять нельзя.

Если помещение без повышенной опасности и тип работ - снятие показаний приборов учета, и U>1000в и квалификация каждого работника бригады < 3, то работу выполнять нельзя.

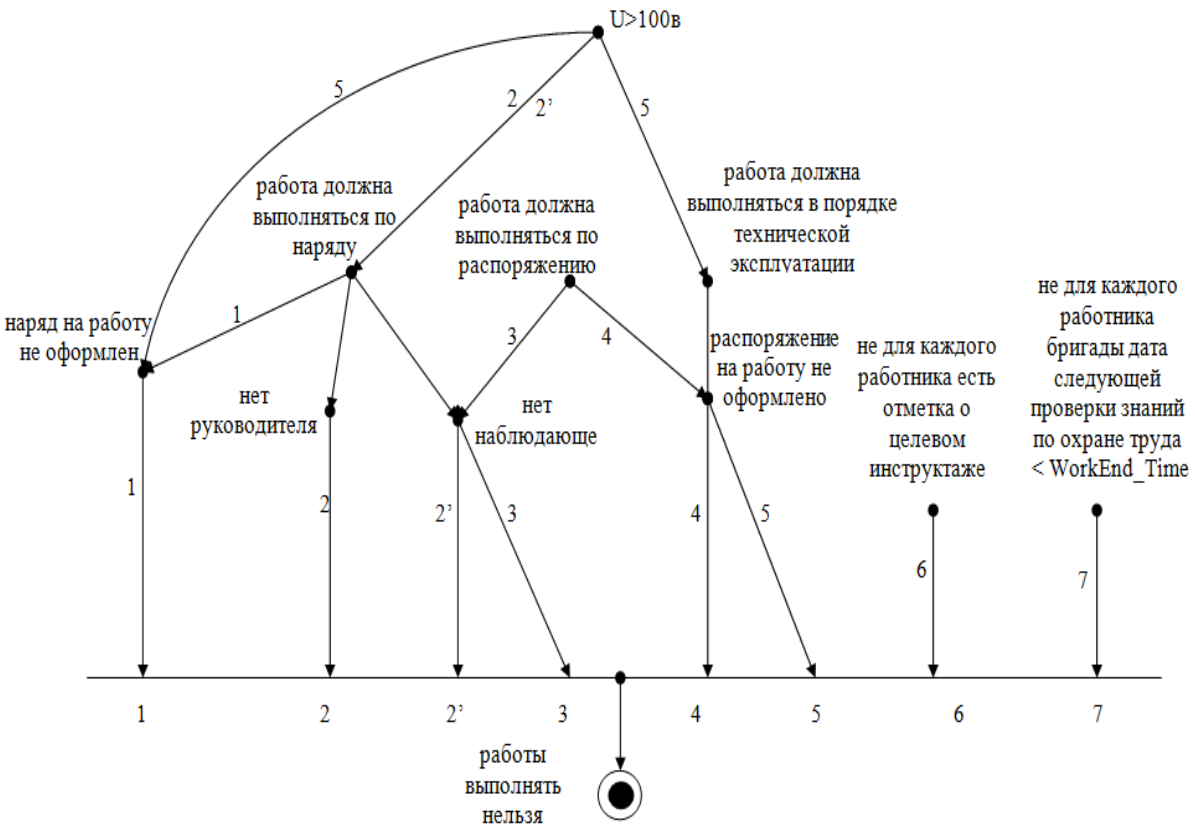

Рисунок 4.9 – И/ИЛИ-граф: работы по наряду и распоряжению, инструктаж

Вершинами графов являются условия, и выводы правил, а дуги задают отношения между ними, объединяя их в правила. Веса дуг соответствуют номерам правил.

На основе И/ИЛИ-графов были произведены проверки посылок подгрупп правил на противоречивость с помощью предложенного в разделе 3.1 метода, выявлены противоречия (некоторые из них показаны в табл. 3.1). Затем были автоматически получены «инверсные» подгруппы для дальнейшей проверки полноты на основе метода, предложенного в разделе 3.2. В подразделе 3.2.3 показаны примеры проверки для подгруппы правил, зависящих от внешних условий.

Для ЭС **контроля и мониторинга компьютерной сети** построены 3 подгруппы правил. В подгруппе «Климат» собраны правила, касающиеся климатических условий, в которых функционирует сеть и ее узлы, а также средств, обеспечивающих соблюдение необходимых условий. В подгруппе правил «Узел» собраны правила, касающиеся отдельного узла сети, в частно-

сти его состояния и нагрузки на него. В подгруппе правил «Сеть» собраны правила, которые касаются собственно сети, в частности состояния каналов связи между узлами и нагрузки на них. Ниже показаны правила для подгрупп «Климат», «Узел» и граф (рис. 4.10) для подгруппы «Климат», остальные приведены в приложении А.

Подгруппа правил «Климат»

1. Если произошло возгорание, или система охлаждения вышла за строя, или система охлаждения неэффективна, или система кондиционирования неэффективна, или необходимо налаживание системы охлаждения, то произошла проблема, требующая вмешательства системного администратора.

2. Если произошел перегрев, и температура окружающей среды высокая, то система кондиционирования помещения работает неэффективно.

3. Если температура системы высокая, то произошел перегрев.

4. Если произошел перегрев, и температура окружающей среды нормальная, то система охлаждения неэффективна.

5. Если необходимо снизить обороты вентилятора системы охлаждения, или необходимо повысить обороты вентилятора системы охлаждения, то система охлаждения требует отладки.

6. Если произошел перегрев системы, и обороты вентилятора системы охлаждения нормальные, то необходимо повысить обороты вентилятора системы охлаждения.

7. Если температура системы нормальная, и обороты вентилятора системы охлаждения высокие, то необходимо снизить обороты вентилятора системы охлаждения.

8. Если температура системы очень высока и температура окружающей среды очень высокая, то произошел пожар.

9. Если обороты вентилятора системы охлаждения отсутствуют, то система охлаждения вышла из строя.

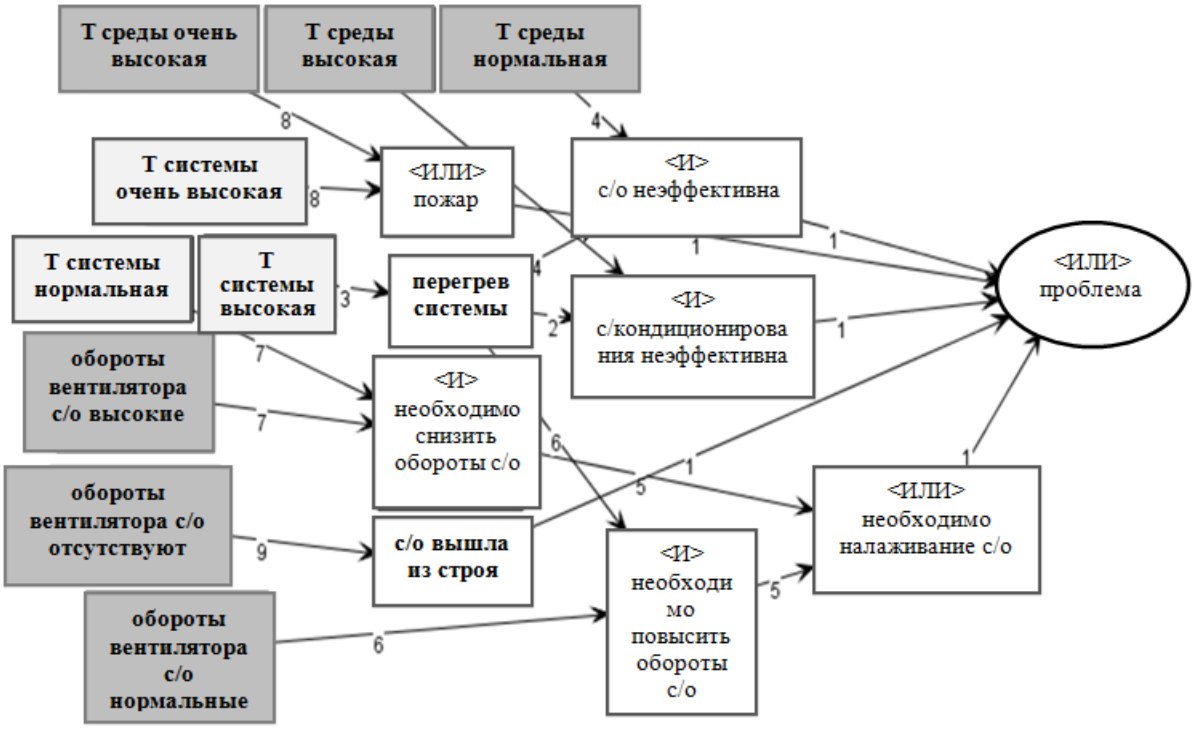

Рисунок 4.10 – И/ИЛИ-граф для подгруппы правил «Климат»

Подгруппа правил «Узел»

1. Если система находится в состоянии активной подкачки, или произошло зависание системы, или система находится в состоянии тротлинга, или произошла утечка памяти, или произошла перегрузки системы, то имеется проблема, требующая вмешательства системного администратора.

2. Если наблюдается перегрев системы, и загруженность процессора большая, и загрузка оперативной памяти низкая, и загрузка дисковой подсистемы низкая, и производительность системы низкая, то возник тротлинг процессора.

3. Если время обработки запроса большое, то производительность системы низкая.

4. Если температура системы высокая, то произошел перегрев системы.

5. Если загрузка процессора низкая, и загрузка дисковой подсистемы низкая, и загрузка оперативной памяти высокая, и время обработки запроса низкое, то произошла утечка памяти.

6. Если система перегружена, и нагрузка на процессор высокая, и загрузка диска высокая, и загрузка оперативной памяти высокая, то произошла перегрузка узла сети.

7. Если время обработки запроса высокое, и количество запросов высокое, и трафик нормальный, то произошла перегрузка системы.

8. Если производительность системы низкая, и загрузка оперативной памяти высокая, и загрузка диска высокая, то происходит интенсивный свопинг.

9. Если производительность системы низкая, и загрузка оперативной памяти низкая, и загрузка диска низкая, и нагрузка на процессор высокая, то произошло зависание системы.

На основе И/ИЛИ-графов была произведена проверка подгрупп посылок правил на противоречивость с помощью предложенного в разделе 3.1 метода, выявлены противоречия.

Затем были автоматически получены «инверсные» подгруппы для проверки полноты на основе метода, предложенного в разделе 3.2. В подразделе 3.2.2 показан пример проверки и выявления неполноты для подгруппы правил «Сеть».

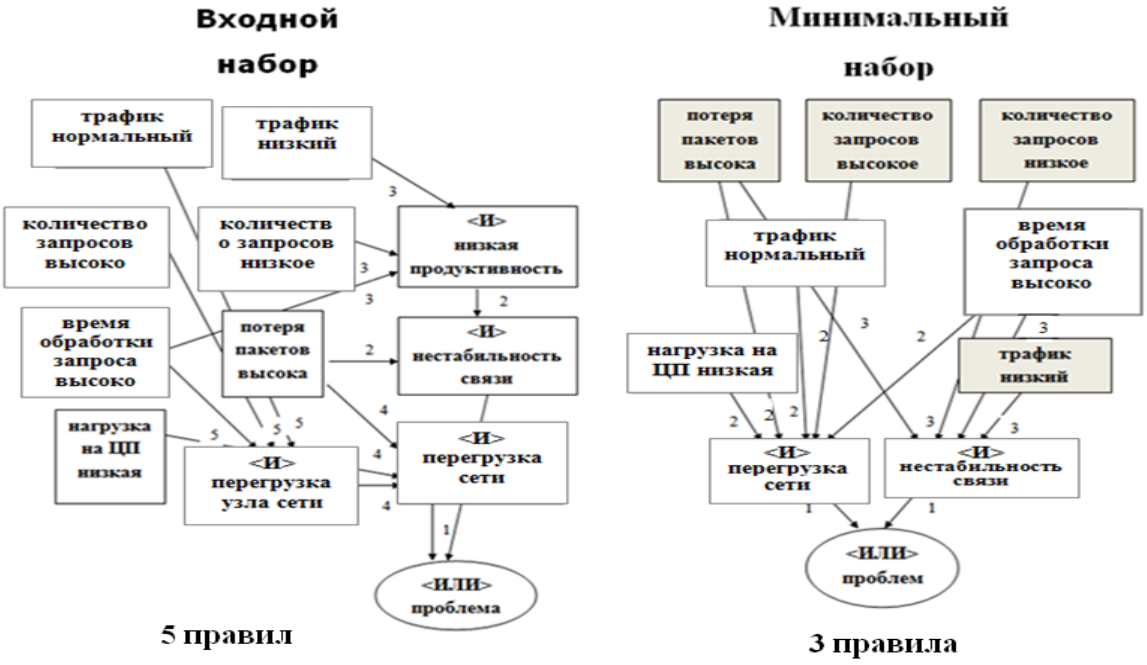

Рисунок 4.11 – Пример минимизации набора правил

На рис. 4.11 показана минимизация правил для подгруппы «Сеть». В левой части рис. 4.11 изображен И/ИЛИ-граф исходного набора, содержащего пять правил, в правой – минимального, в котором три правила. Здесь видно, что связанные правила объединяются вместе. Таким образом, исходный набор из пяти правил может быть заменен тремя.

Затем был произведен поиск циклических зависимостей в правилах [59]. Обращая внимание на вершины «перегрев системы», «необходима настройка системы охлаждения» и «необходимо увеличить обороты системы охлаждения» подгруппы «Климат» на рис. 4.12, замечаем, что они находятся в циклической зависимости. Циклические зависимости могут привести к невозможности получения вывода, поэтому они были найдены и устранены.

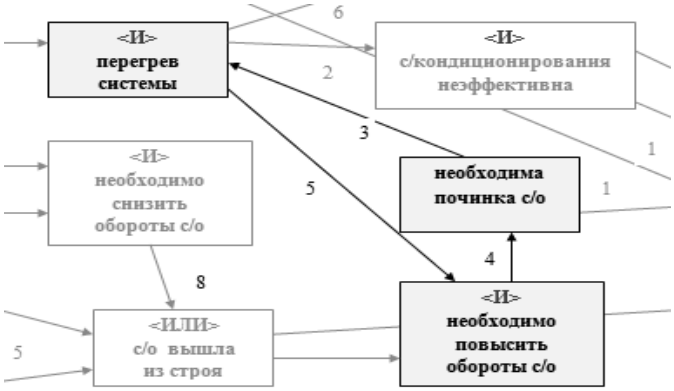

Рисунок 4.12 – Пример цикличности в правилах

4.3.3 Формализация знаний

Для формализации знаний была выбрана готовая машины вывода для свободно распространяемая среды разработки CLIPS. Аббревиатура CLIPS означает C Language Integrated Production System, удобного для разработки баз знаний и макетов экспертных систем [39].

Факты в CLIPS представляет собой фрагмент данных, помещенных в текущий список фактов системы (рабочую память). Правила состоят из посылок и следствия. Посылки правил удовлетворяются в зависимости от наличия или отсутствия некоторых заданных фактов в списке фактов. Условные элементы в виде образцов состоят из списка ограничений полей, групповых

символов (wildcards) и переменных, которые используются для поиска множества фактов, которые соответствуют заданному образцу. Следствие правила представляется набором действий, которые нужно выполнить в случае, если правило применимо к текущей ситуации. Правилам могут назначаться приоритеты (salience).

Согласно предложенной в 4.1 ИТ на данном этапе происходит автоматическое преобразование каждой группы (подгруппы) правил, сформированных в редакторе правил на этапе структуризации, в язык продукционного программирования CLIPS с учетом уменьшения времени логического вывода за счет оптимального назначения приоритетов правилам.

Описание И/ИЛИ-графа для редактора правил хранится в формате XML.

Первым шагом является выбор из файла названий параметров и их возможных значений. Полученная информация автоматически формирует промежуточные правила на языке CLIPS, в частности, правила нулевого уровня определяют значения входных данных для контроля. Пример правила, которое определяет температуру системы, представлено ниже.

```
(defrule  dop-1
  (not (av (a T системы) (v ?)
  (not (result ?))
=>
  (if (yes-or-no-p "T системы - Высокая? (yes/no) ")
  then
  (assert (av (a T системы) (v Высокая)))
  else
  (assert (av (a T системы) (v Нормальная)))
))
```

Уровней промежуточных правил может быть несколько. Далее формируем терминальные правила, в выводах которых находится состояние объекта контроля, к примеру, правило следующего вида:

```
(defrule  base-1
  (av (a T системы) (v Высокая))
  (not (result ?))
=>
  (assert (av (a Состояние) (v Перегрев_системы)))
```

```
(assert (result “Проблема: Перегрев системы”
))
```

На рис. 4.13 показаны уровни правил подгруппы «Сеть». Каждый предыдущий уровень вложенности зависит только от следующего. Нулевой уровень – это правила, которые определяют значения входных параметров, уровень 4 – терминальный, уровни 2 и 3 – промежуточные.

Чтобы минимизировать число срабатываемых правил при контроле, назначались приоритеты правилам в соответствии с соображениями, указанными в 4.1.

Итак, правила на языке CLIPS, соответствующие сформированным в редакторе правил и описывающие ПрО, готовы. Однако программа с такими правилами не будет работать, потому, во-первых, каким-то образом в систему должны попадать начальные факты, а во-вторых, результат должен выводиться пользователю.

Поэтому предварительно сформирован *шаблон модуля*, предназначенного для правил контроля, одинаковый для любых правил контроля, который состоит из управляющих правил, предназначенных для вывода результата, и функций, необходимых для получения входных данных.

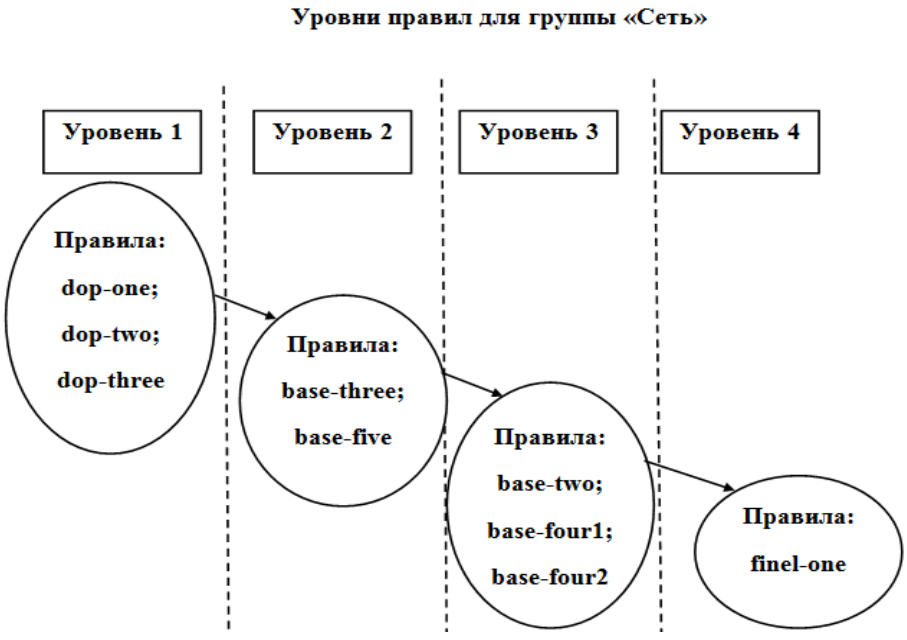

Рис.4.13 – Уровни правил для подгруппы «Сеть»

4.3.4 Кодирование ЭС

Модули Clips, соответствующие подгруппам правил, получены автоматически на этапе формализации. Далее они объединяются в общую модульную структуру с помощью такого средства, как модули Clips, с учетом предложенных режимов (см. подраздел 4.1). При соединении модулей ЭС системы используется дополнительные связующие управляющие правила, в которых указывается последовательность запуска модулей, а также, в каких случаях они могут запускаться либо не запускаться. Для первого режима, когда система прекращает работу по первой же подгруппе правил, которые нарушены, указывается, что следующий модуль должен быть запущен только в случае, если на основе предыдущего модуля не выявлены нарушения. Для второго режима все подгруппы запускаются подряд. Программные модули и управляющие правила представлены в приложении Б.

При работе ЭС используется машина вывода CLIPS. В CLIPS используется *прямой логический вывод*. На каждом шаге активированные правила (т.е. правила, условия которых удовлетворяются в данный момент) помещаются в план решения задачи. Размещение в плане определяется приоритетом правила (salience) и текущей стратегией разрешения конфликтов. В случае, если в данный момент применимо более одного правила, механизм логического вывода использует текущую *стратегию разрешения конфликтов* (conflict resolution strategy), которая определяет, какое именно правило будет выполнено. После этого CLIPS выполняет действия, описанные в следствии выбранного правила (которые могут оказать влияние на список применимых правил), и приступает к выбору следующего правила. Этот процесс продолжается до тех пор, пока список применимых правил не опустеет. Таким образом, настройка машины вывода состоит в выборе необходимой стратегии разрешения конфликтов. Для правил контроля предпочтителен поиск в глубину в связи с тем, что каждая подгруппа правил содержит один окончательный вывод, соответствующий состоянию объекта. Поиск в ширину используется в случае, когда выводов может быть много, например, в задачах диагностики, ко-

гда целесообразно сначала просмотреть все симптомы, соответствующие одному уровню.

4.3.5 Тестирование правил ЭС

Правила систем, как для каждой подгруппы, так и для общей модульной структуры, были протестированы в соответствии с рекомендациями информационной технологии, приведенной в подразделе 4.1. Протоколы тестирования приведены в приложении В.

На примере контроля и мониторинга сети на рис. 4.14 показаны все ветви, по которым проводилось тестирование подгруппы «Сеть».

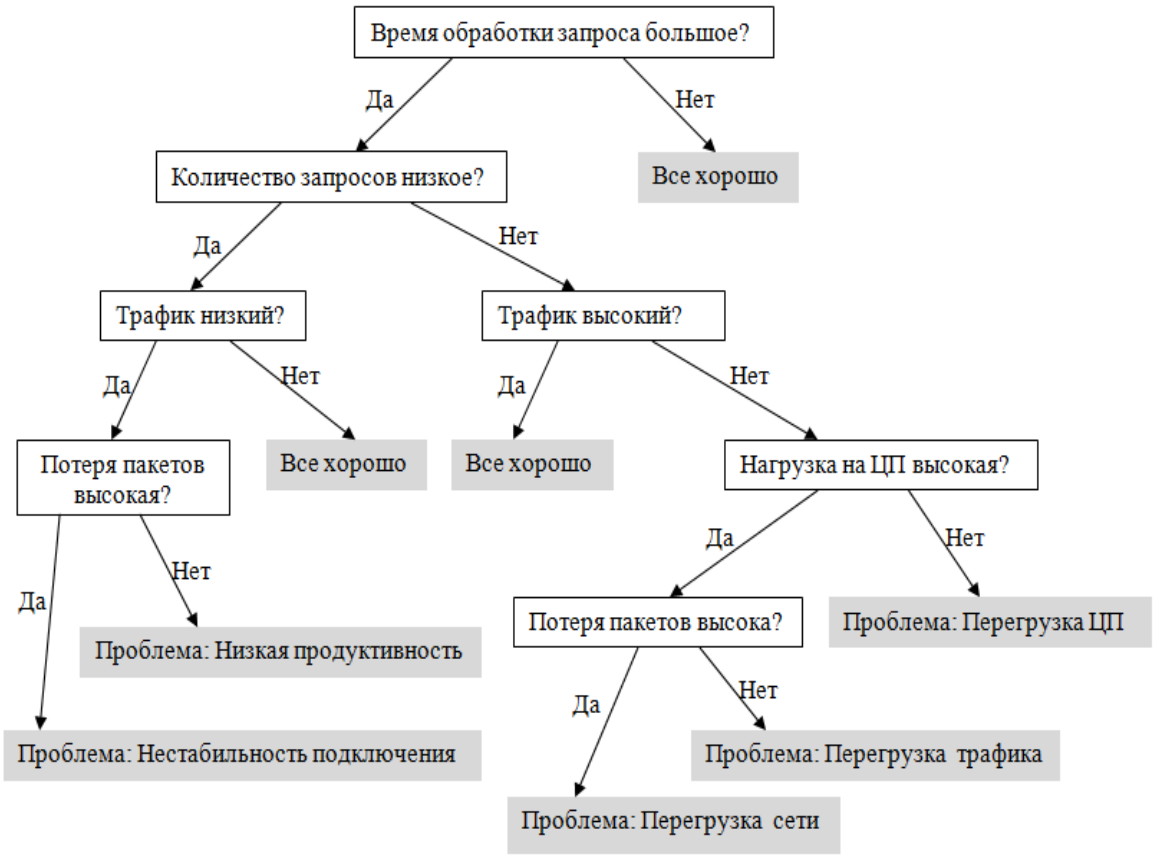

Рисунок 4.14 – Дерево решений при тестировании ЭС

4.3.6 Интеграция БЗ и машины вывода с компонентами для получения параметров и вывода результатов

В результате проведенного анализа сделан вывод о том, что ЭС по безопасной работе с электроустановками наиболее эффективно использовать в

составе автоматизированного рабочего места диспетчера предприятия. В автономном режиме ЭС задает вопросы пользователю о степени опасности помещения, составе бригад и тому подобное, а затем делает выводы о допустимости проведения работ, однако наиболее эффективно ее использование в составе автоматизированного рабочего места (АРМ) диспетчера организации, эксплуатирующей электрические сети, совместно с системой визуализации на основе сенсорных датчиков. При этом диспетчер сможет видеть на экране компьютера помещение, где проходят работы на электрооборудовании, работников бригады и другое.

В [79] предложена ИТ на основе ЭС для принятия решений в составе автоматизированного рабочего места диспетчера организации по безопасной эксплуатации электрических установок наряду с системой локализации и визуализации объектов в помещениях, где производятся работы. Данные из системы WSL о работающих бригадах (их местоположение, состав), степень опасности помещений и другие передаются в подсистему принятия решений (рис. 4.15). Кроме того, диспетчер сообщает о заданиях для бригад, а ЭС система выдает вердикт о возможности выполнения работ. Такого рода системы визуализации может быть использована в разных областях для принятия решений при анализе расположения пространственных объектов, в том числе, и движущихся, это контроль, мониторинг, управление, навигация и другие. Идентификация движущегося объекта (человека или техники) производится по сигналу беспроводного устройства, которое может определяться MAC-адресом или логическим именем.

Рис. 4.15 – Визуализация положений датчиков и принятие решений

Кроме беспроводных сенсорных сетей, еще более широкие возможности для контроля объектов дает машинное зрение. Оно может быть использовано как для контроля безопасности выполнения работ, так и для охраны объектов. Рассмотрим несколько объектов (блоков), важных для контроля при выполнении работ на электрооборудовании, состояние которых необходимо отслеживать. Такие объекты тем или иным образом обозначаются, и для них важно понятие состояния:

1) Выключатель линии, состояние: включен или выключен.

Правило: Если выключатель линии включен, то нельзя проводить определенные работы и нельзя устанавливать перемычку.

Типы объектов этого вида: различные типы выключателей, разъединителей, ячеек.

2) Ограждения, состояния: расположение относительно других объектов (блоков), наличие плакатов, пребывание людей, наличие механизмов.

Правила:

Выполнять определенные работы в границах ограждения: там должны быть или люди и инструменты, или люди и механизмы.

Если ограждения предназначены для определенного вида работ, то на ограждениях должны быть определенные плакаты.

3) Токоведущие части, состояние: расстояние до людей и механизмов.

Правила: расстояние от токоведущих частей до людей и механизмов должно быть не менее заданной величины.

4) Соединения (перемычки), состояния: расположение относительно других объектов (блоков).

Правила определяют, где (между какими объектами) их можно устанавливать и где нельзя.

Машинное зрение может "увидеть" определенные объекты, ограждения, токоведущие части и перемычки, а также их состояния, что автоматически передается в правила, которые предупреждают диспетчера о возможных не-

соответствиях. Чтобы машинное зрение могло «увидеть», его заранее обучают распознавать, существуют картинки с различными видами объектов.

Для **контроля и мониторинга сети** разработаны компоненты для получения параметров объекта контроля и графический интерфейс с пользователем [61] на языке C++, а также проведена их интеграция с машиной вывода и базами правил.

В случае специальной проверки по требованию пользователя или системного администратора ЭС система получает данные о состоянии компьютерной сети, анализирует, сообщает о неполадках и проблемах в сети и дает рекомендации по их устранению.

Для получения параметров сети, которые анализируются ЭС, необходимо воспользоваться существующими системами мониторинга сети, а машину вывода с правилами интегрировать в эту систему.

Был проведен анализ существующих систем контроля и мониторинга компьютерной сети (см. подраздел 1.2.2) и выбрана NetXMS [15] – система контроля и мониторинга сети с открытым исходным кодом, разработанная на языках C/C++ (сервер, агент) и Java [41] (центр контроля за системой мониторинга сети). Отличительной особенностью системы является то, что данная система работает на всех основных операционных системах.

Для этой системы был разработан дополнительный модуль [30], отвечающий за анализ состояния сети, на вход от NetXMS поступают такие параметры как пинг, количество пакетов в секунду, трафик в секунду, количество потерянных пакетов, нагрузка на центральный процессор и другие. Программа на основе ЭС анализирует состояние сети и выдает вердикт, есть ли проблемы в сети, такие как низкая производительность, нестабильность связи, перегрузки сети и тому подобное. В случае проблем выдаются рекомендации по их исправлению.

Данные хранятся в двух файлах: INI-файл настроек эталонов и CSV-файл с оперативными данными о состоянии сети (рис. 4.16).

Оба файла содержат одни и те же состояния объекта, только в первом файле – эталоны, а во втором – текущие оперативные значения. Так как каждая сеть отличается друг от друга аппаратным и программным обеспечением, они обладают уникальными значениями состояний объекта, которые для этой системы считаются нормой. Для правильного функционирования системы нужно перед началом использования однократно ввести диапазоны стандартных состояний объекта для конкретной сети. Также надо указать формат и единицы измерения, например, МБ/с или Мбит/с.

INI-файл настроек эталонов

```
[NetworkSettings]

ping=100
cpuLoad=75
packageLoss=10
traffic=500
queriesNumber=1000
```

CSV-файл с оперативными данными про состояние сети

Код	Название	Значение	Время	Статус
25	ping	1	1.16.46	OK
28	cpuLoad	3	1.17.17	OK
38	packLoss	0	1.17.18	OK
39	traffic	354	1.17.09	OK
40	qNumber	647	1.16.48	OK

Рисунок 4.16 – Структуры данных для хранения информации о сети

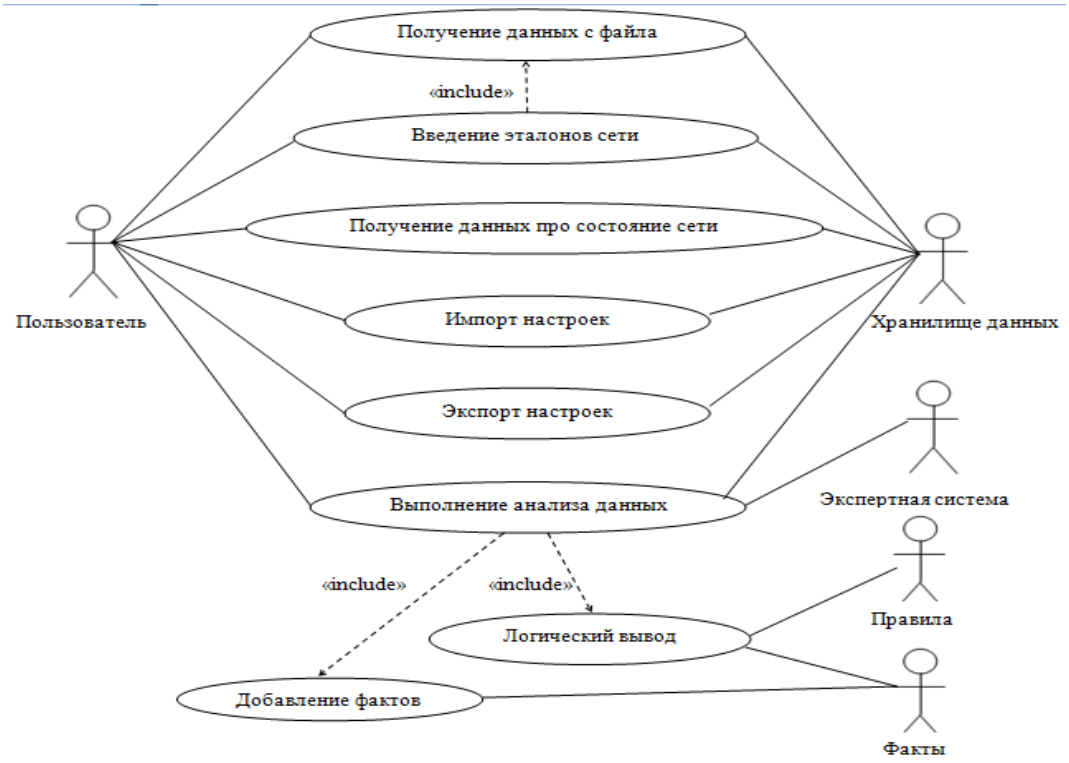

Рисунок 4.17 – Диаграмма прецедентов

На рис. 4.17 представлена диаграмма прецедентов, включающая такие функции как работа с эталонами сети, получение данных из сети и их анализ с выводом результатов.

Архитектура модуля анализа в виде диаграммы компонентов представлена на рис. 4.18. Система состоит из двух частей: взаимодействие с пользователем, получение данных из источника, настройка эталонов; анализ данных на базе ЭС.

Инструменты разработки: язык C++, Clips – средство разработки ЭС, GIT – система контроля версий продукта. Для интеграции Microsoft Visual Studio и Clips использована библиотека clipscpp.h (clipsstatic64.dll), которая скомпилирована специально под C ++ и под 64x-битные системы, с помощью этой библиотеки clips файл запускается автономно и работает в таком же виде, как и в среде Clips (ранее для такой интеграции использовались каналы (pipes)).

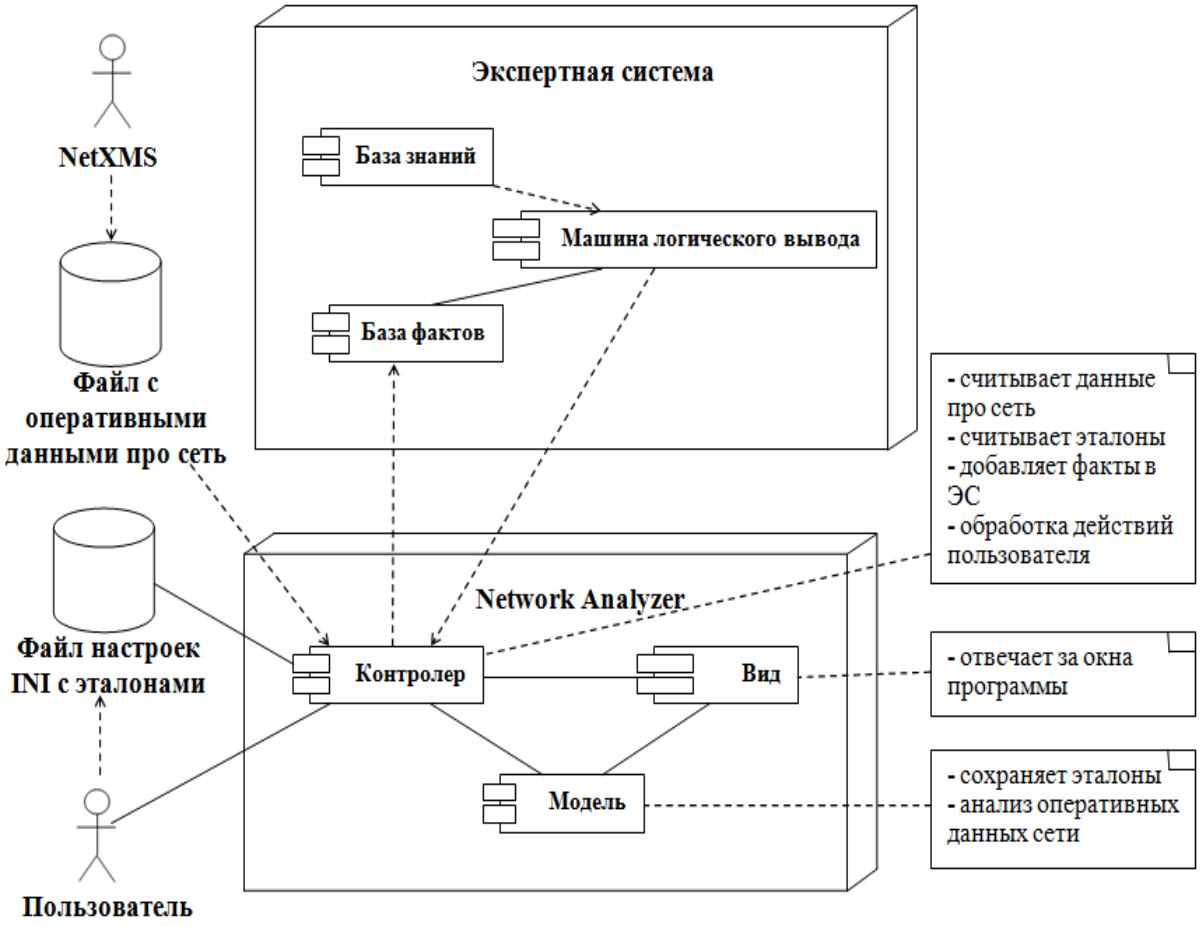

Рисунок 4.18 – Диаграмма компонентов

На рис. 4.19 представлено главное окно разработанного модуля. В левой части находится поле для создания эталонов, в правой части поле с результатами анализа как в автоматическом режиме, когда данные вводятся от сети, так и в автономном режиме для ЭС, когда модуль задает вопросы пользователю о параметрах сети и выдает вердикт о наличии либо отсутствии проблемы. В последнем случае выдаются рекомендации о способах устранения проблемы.

Данная разработка может быть интересна тем, кто пользуется системами мониторинга сети и желает анализировать получаемые от сети данные комплексно, в существующих системах мониторинга проводится анализ для каждого параметра отдельно. Кроме того, в предлагаемой технологии конечный пользователь может сам создавать правила анализа, в существующих системах правила встроены и недоступны для корректировок.

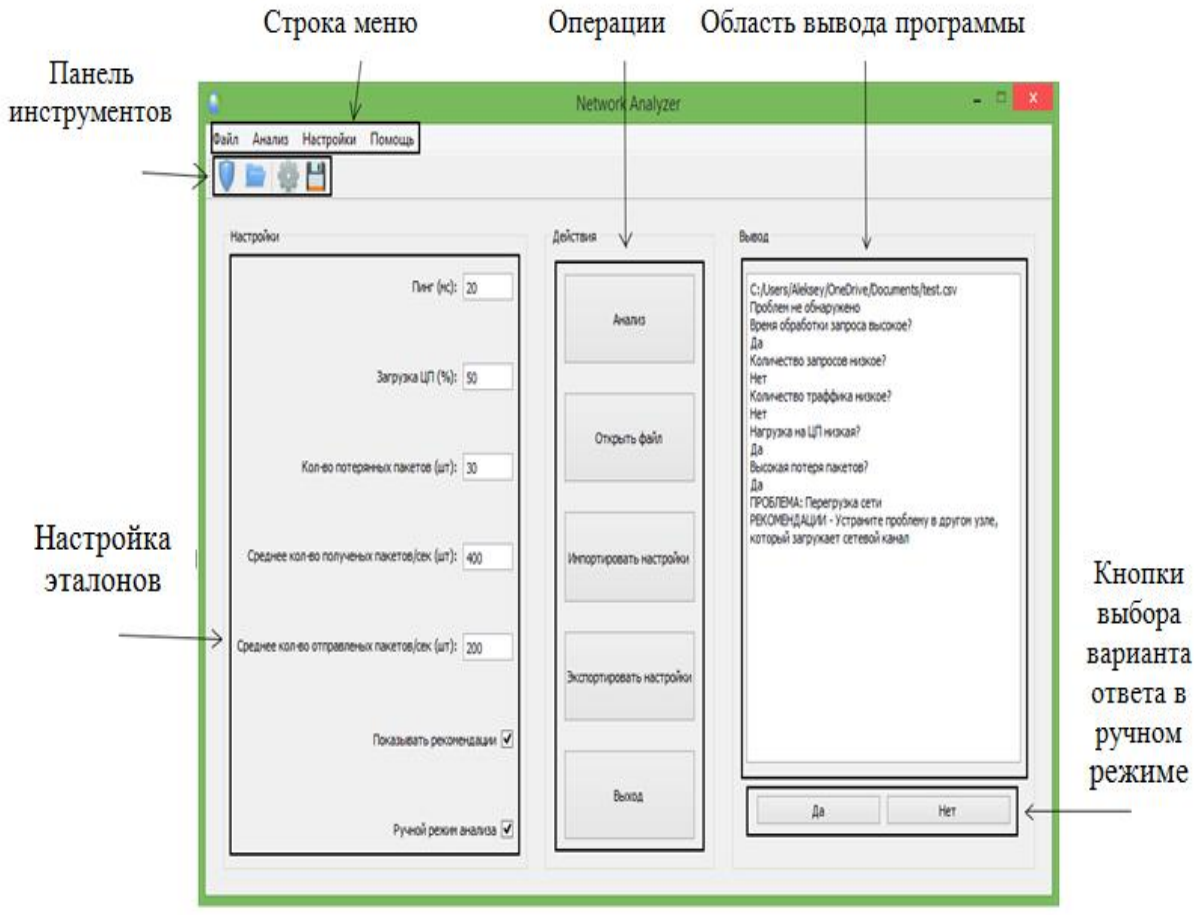

Рисунок 4.19 – Подсистема консультаций модуля анализа сети

При интеграционном тестировании в программу через INI и CSV файлы передаются тестовые наборы данных, получается результат, и сравнивается с результатом, которого мы ожидаем, если результат совпадает, тест можно считать успешным.

Эффективность предлагаемой ЭС для анализа в составе системы контроля и мониторинга сети обусловлена уменьшением времени реакции администратора сети на возникновение опасной и/или аварийной ситуации, что, в конечном счете, уменьшает количество аварийных ситуаций в сети.

Данная разработка может быть интересна тем, кто пользуется системами мониторинга и контроля сети и желает анализировать получаемые от сети данные комплексно, в существующих системах мониторинга проводится анализ для каждого параметра отдельно.

При сопровождении созданных ЭС время редактирования правил контроля значительно сократилось и занимает 3 – 4 часа, так как этой работой занимается, преимущественно, эксперт с помощью редактора правил. При традиционных технологиях такая работа обычно проводится в течение нескольких суток.

4.4 Апробация редактора правил и систем контроля, исследование их результативности

4.4.1 Использование редактора правил для обучения

Разработанный редактор правил контроля использовался при обучении студентов при выполнении ими курсовой работы по дисциплине «Интеллектуальный анализ данных» Института Компьютерных Систем ОНПУ. Задания состояли в разработке демонстрационных прототипов ЭС контроля для различных ПрО, в частности, подгрупп для контроля работы компьютерной сети и безопасной работы с электроустановками с помощью среды разработки Clips. Так как студенты не являются экспертами в предлагаемых ПрО, а учатся по специальностям в сфере информационных технологий, им были пред-

ложены описания основных сведений из этих областей, необходимых для работы. Целью курсовой работы является получение навыков в построении знаниеориентированных систем.

Для апробации редактора правил контроля и исследования его результативности было проведено сравнение полученных прототипов ЭС с использованием редактора правил и без него.

Общее количество студентов, принимавших участие в испытаниях, составило 42 человека, которые были разбиты на две группы (контрольную и экспериментальную) по 21 человеку в каждой. Проводили разбиение согласно баллу успеваемости по базовой дисциплине. Баллы оценивались по 100-бальной системе. Баллы студентов по базовой дисциплине для каждой группы приведены в приложении Г, таблицы Г.1, Г.2.

Покажем далее статистическую неразличимость двух групп по успеваемости. Для этого будем использовать такие характеристики как мода, медиана и гистограмма распределения. Для построения описательных характеристик упорядочим средние баллы успеваемости по возрастанию (см. табл. Г.1 приложение Г).

В табл. Г.3 приложения Г вариационный ряд *x* соответствует группе, где не был использован редактор правил, вариационный ряд *y* соответствует группе, в которой редактор был использован.

Выборочная медиана определяется как результат наблюдения, занимающий центральное место в вариационном ряду, построенном по выборке с нечётным числом элементов, или полусумма двух результатов наблюдений, занимающих два центральных места в вариационном ряду, построенном по выборке с чётным числом элементов [54]. В нашем случае количество студентов нечетно, поэтому центральным в вариационном ряду является место с номером 11. Из таблицы Г.3 видно, что выборочная медиана и для ряда *x,* и для ряда *y* равна 4.

Далее оценим статистическую надежность полученного результата. Для расчёта доверительного интервала медианы применим следующую формулу [29]:

$$N_d = \left[\frac{n}{2} - \frac{u_\alpha \sqrt{n}}{2} \right] \qquad (4.1)$$

где n – число элементов выборки; u_α – α-квантиль нормального распределения; [29] – операция взятия целой части числа.

Доверительный интервал медианы равен $X(N_d)....X(n+1-N_d)$ (4.2)

где $X(N_d)$ – элемент, стоящий на позиции N_d в вариационном ряду; $X(n+1-N_d)$–элемент, стоящий на позиции $n+1-N_d$ в вариационном ряду.

Рассчитаем доверительный интервал медианы для двух полученных выборок. Значения переменных: n=21, $\alpha = 95\%$, $u_\alpha = 1{,}96$. Пользуясь формулой (4.1), получаем:

$$\mathrm{N_d} = \left[\frac{21}{2} - \frac{1{,}96\sqrt{21}}{2} \right] = 6$$

Тогда, согласно формуле (4.2), доверительный интервал медианы будет равен:

$$Md_z = 3...4,\ \alpha = 95\%$$
$$Md_y = 3...4,\ \alpha = 95\%$$

Далее построим ряды распределения для обеих выборок (табл. Г.4 приложения Г). Мода определяется как наиболее вероятное значение случайной величины. Из таблицы Г.4 видно, что для обеих выборок мода равняется 3.

Итак, и медиана, и мода одинаковы для обеих выборок. Таким образом, можно сделать вывод о статистической неразличимости показателей успеваемости двух групп – контрольной и экспериментальной.

Далее каждой из групп были поставлены одинаковые задачи по разработке ЭС. Студенты экспериментальной группы использовали для разработки редактор правил, затем правила преобразовывались в язык продукционного программирования Clips. В контрольной группе прототипы ЭС разрабаты-

вались непосредственно в Clips. Студенты экспериментальной группы создавали правила на естественном языке как в текстовом, там и с помощью И/ИЛИ-графа, проверяли их на противоречивость, полноту и достижимость состояний в редакторе правил, результаты их работы в таблице Г.5 приложени Г.

Как видно из таблицы Г.5, при проверке на противоречивость правил ошибки появлялись и исправлялись в среднем в 10% правил, при проверке на достижимость состояний удалось удалить в среднем 5% неверных правил, при проверке полноты база правил была увеличена в среднем на 11%.

Количество студентов, вовремя сдавших курсовые работы, увеличилось при использовании редактора правил в среднем на 8%, как показано на рисунке 4.20.

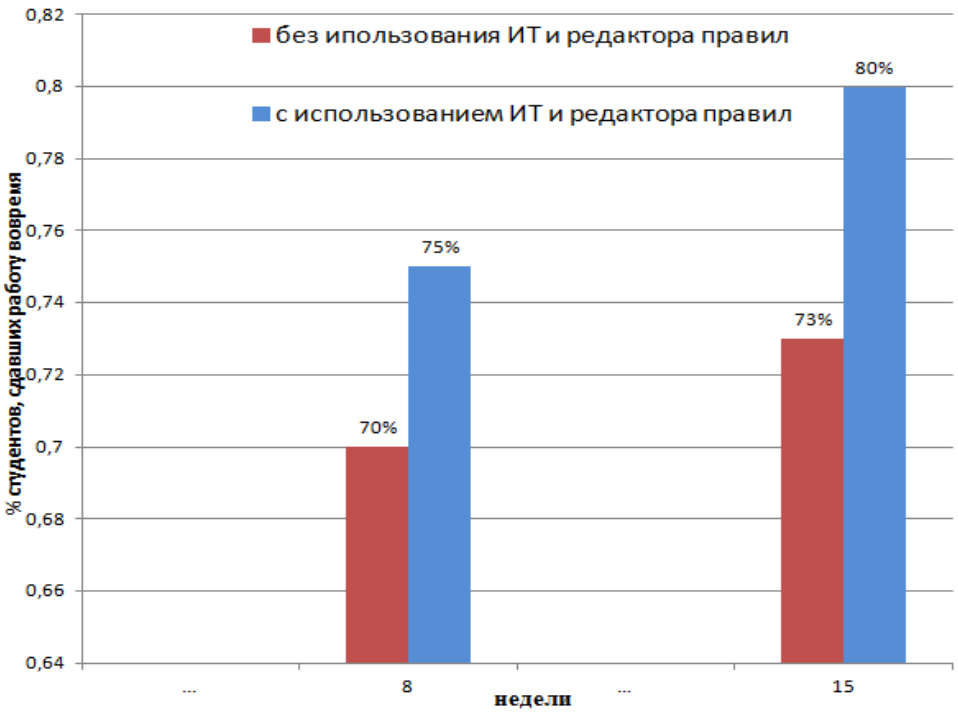

Рисунок 4.20– График сдачи курсовых работ

4.4.2 Использование ЭС контроля для обучения ЛПР

ЭС применялись в качестве *тренажеров* при обучении ЛПР, как диспетчеров по безопасной работе с электроустановками, так и системных администраторов компьютерных сетей. При этом обучаемый наблюдает и анализирует, как должен действовать ЛПР, на примере ЭС, имитирующей работу

опытного эксперта, пробует разные ситуации и на этой основе самостоятельно делает умозаключения,

Методика использования разработанных ЭС в учебном процессе в качестве тренажеров для обучения ЛПР состоит из следующих шагов:

1) знакомство с теорией и нормативными документами;

2) запуск ЭС в автономном режиме при различных вариантах входных параметров, в результате автоматически определяются состояния объекта контроля;

3) обучающийся самостоятельно делает выводы о том, при каких параметрах объект переходит в аварийный режим, а при которых работает нормально, и самостоятельно строит правила для одной или несколько групп с помощью И/ИЛИ-графа и в текстовом виде, используя редактор правил;

4) построеные правила проверяются редактором правил на противоречивость посылок, полноту, достижимость состояний и цикличность;

5) созданные графы сравниваются с графами-эталонами, на этой основе оцениваются результаты обучения; затем граф-эталон предъявляется обучаемому для сравнения.

На рис. 4.21 представлено окно приложения, где ЭС работает в автономном режиме и позволяет обучаемому запускать консультации для возможных входных параметров, и показывает результаты о состояниях объекта, соответствующих введенным параметрам. Затем обучаемый самостоятельно строит правила в редакторе правил. Затем граф-эталон предъявляется обучаемому для сравнения.

На базе обучения системных администраторов сети и диспетчеров по электробезопасности были проведены эксперименты, позволяющие оценить работу тренажера, включающего как саму ЭС, так и редактор правил. В экспериментах участвовало соответственно 20 и 17 человек, им было предложено создать от 6 до 15 правил в рамках обучения и проверить данные правила на противоречивость, полноту, достижимость состояний с помощью редактора правил.

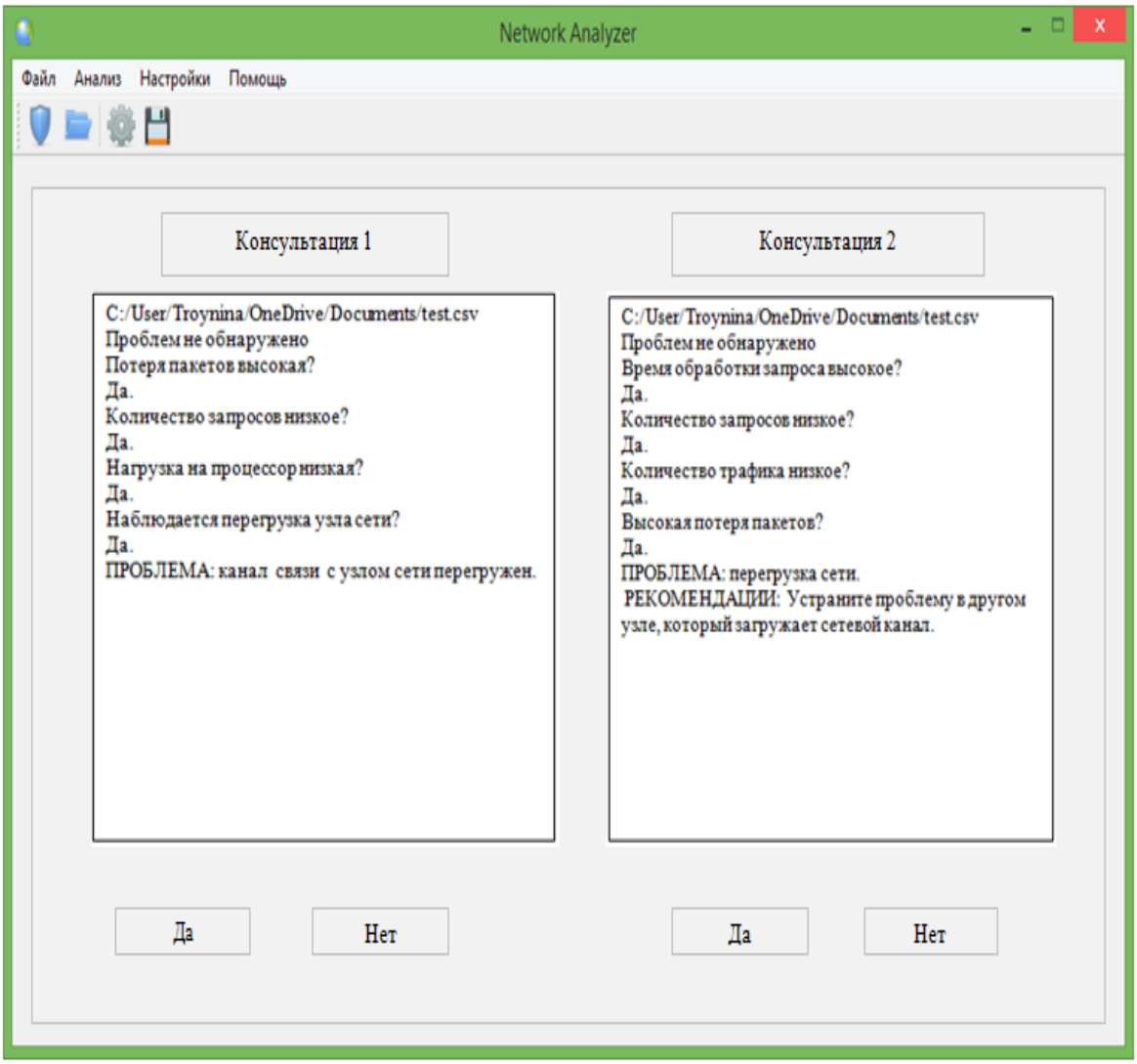

Рисунок 4.21 – Подсистема консультаций для контроля сети

Данные о проверке правил при обучении администраторов сети представлены в таблице 4.2; итоги по всем экспериментам, связанным с проверкой правил, показаны в таблице 4.3.

Сделаны выводы о том, что при проверке на противоречивость правил ошибки появлялись и исправлялись в среднем в 10% правил, при проверке на достижимость состояний и избыточность удалось удалить в среднем 5.1% неверных правил, при проверке полноты база правил была увеличена в среднем на 11.69%.

Время обучения на основе тренажеров удалось уменьшить по сравнению с традиционной методикой обучения, основанной на лекциях и проверке знаний на основе тестов, на 20% без изменения качества обучения.

Таблица 4.2 – Проверки правил контроля при обучении администраторов сети

Количество обучаемых администраторов	Количество правил созданных обучаемым	Количество правил с противоречивостью	% правил с противоречивостью	количество правил с ошибками достижимости	% правил с ошибками достижимости	Количество дополненных правил	% дополненных правил
1	9	2	22,2	1	11,1	3	33,33
2	11	2	18,2	1	9,1	2	18,2
3	9	1	11,1	0	0	2	22,2
4	6	0	0	0	0	0	0
5	10	2	20	1	10	1	10
6	8	1	12,5	0	0	1	12,5
7	7	0	0	0	0	1	14,3
8	6	1	16,6	1	16,7	1	16,6
9	6	0	0	0	0	0	0
10	9	1	11,1	0	0	1	11,1
11	10	2	20	1	10	2	20
12	11	3	27, 3	1	9,1	2	18,2
13	7	1	14,3	1	14,3	0	0
14	8	1	12,5	0	0	2	25
15	8	0	0	0	0	0	0
16	9	1	11,1	1	11,1	1	11,1
17	7	0	0	0	0	0	0
18	6	0	0	0	0	0	0
19	9	1	11,11	0	0	2	22,2
20	8	1	12,5	1	12,5	1	12,5
Среднее значение в %			9,9		5,2		12,4

Таблица 4.3 – Итоговая таблица проверки правил контроля

Область, для которой был проведен эксперимент	Количество созданных правил	Количество правил с противоречивостью	% правил с противоречивостью	количество правил с ошибками достижимости	% правил с ошибками достижимости	Количество дополненных правил	% дополненных правил
Обучение студентов	218	25	10	13	5,1	25	11,1

Продолжение таблицы 4.3

Обучение администраторов	164	20	9,9	9	5,2	22	12,3
Электробезопасность	50	5	10	2	4	6	12
Сеть	30	3	10	1	3,3	3	10
Суммарные значения	462	53	10	25	5,1	56	11,69

4.4.3 Принятие решений на основе знаниеориентированных систем контроля

ЭС по безопасной работе с электроустановками используется при принятии решений диспетчером о возможности проведения работ с электроустановками. Данные о параметрах объектов вводятся диспетчером вручную либо применяются автоматические средства для контроля, измеряющие температуру, влажность в помещении и другие параметры. ЭС определяет, выполняются ли требования техники безопасности и, в случае нарушений, сообщает об этом диспетчеру.

Достоверность принимаемых решений проверялась на примере ЭС по безопасной работе с электроустановками. В организации, где внедрялась ЭС, экспертом был проанализирован журнал за два месяца о работе диспетчера до внедрения системы и за два месяца после внедрения. Эксперт просматривал в отчетах решения диспетчера и выявлял, сколько было сделано ошибок первого рода, когда диспетчер выносил неправильные решения о том, что работы выполнять нельзя, и второго рода, когда диспетчер выносил неправильные решения о том, что работы выполнять можно. Также была вычислена достоверность правильных решений как вероятностная мера, определяющая отношение правильных решений к общему числу решений (до внедрения и после внедрения ЭС).

Сведения о результатах работы диспетчера за два месяца (март, апрель 2015 г.) до внедрения ЭС представлены в таблицах Д.1 и Г.2 в приложении Д.

За два месяца до внедрения нашей системы диспетчер принял 200 неправильных решений из них ошибок первого рода – 10, что составляет 5% от общего количества неправильных решений, а ошибок второго рода 190 – 95% (рис. 4.2.6).

Сведения о результатах работы диспетчера за два месяца (июнь, июль 2015 года) после внедрения ЭС системы представлены в таблицах Д.3, Д.4 приложения Д.

За два месяца после внедрения ЭС диспетчер принял всего 45 неправильных решений из них ошибок первого рода – 3, что составляет 6,7% от общего количества неправильных решений, а ошибок второго рода 42 – 93,3% (рис. 4.2.6). Результаты изображены на рис. 4.22-4.25.

Рисунок 4.22 – Решение диспетчера до внедрения ЭС, март

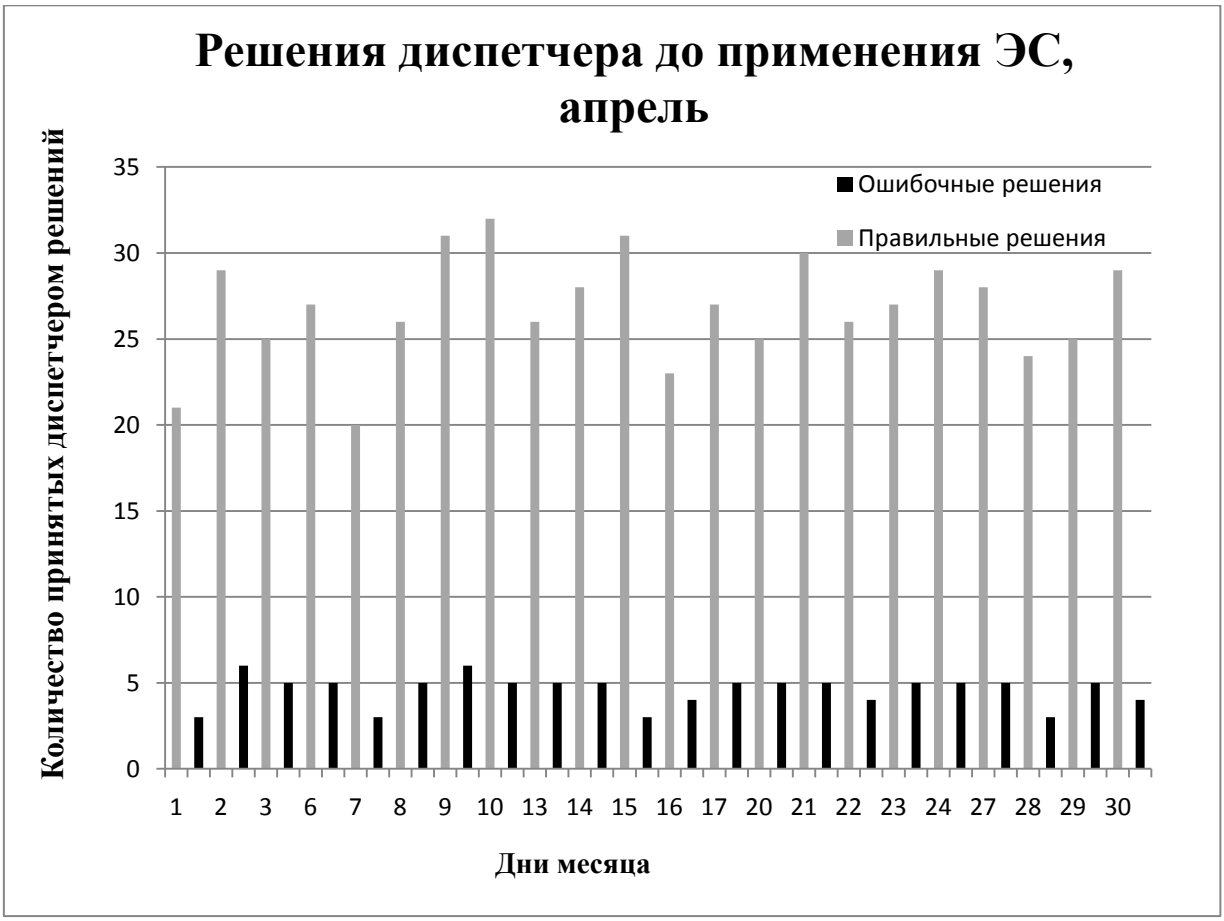

Рисунок 4.23 – Решение диспетчера до внедрения ЭС, апрель

Рисунок 4.24 – Решение диспетчера после внедрения ЭС, июнь

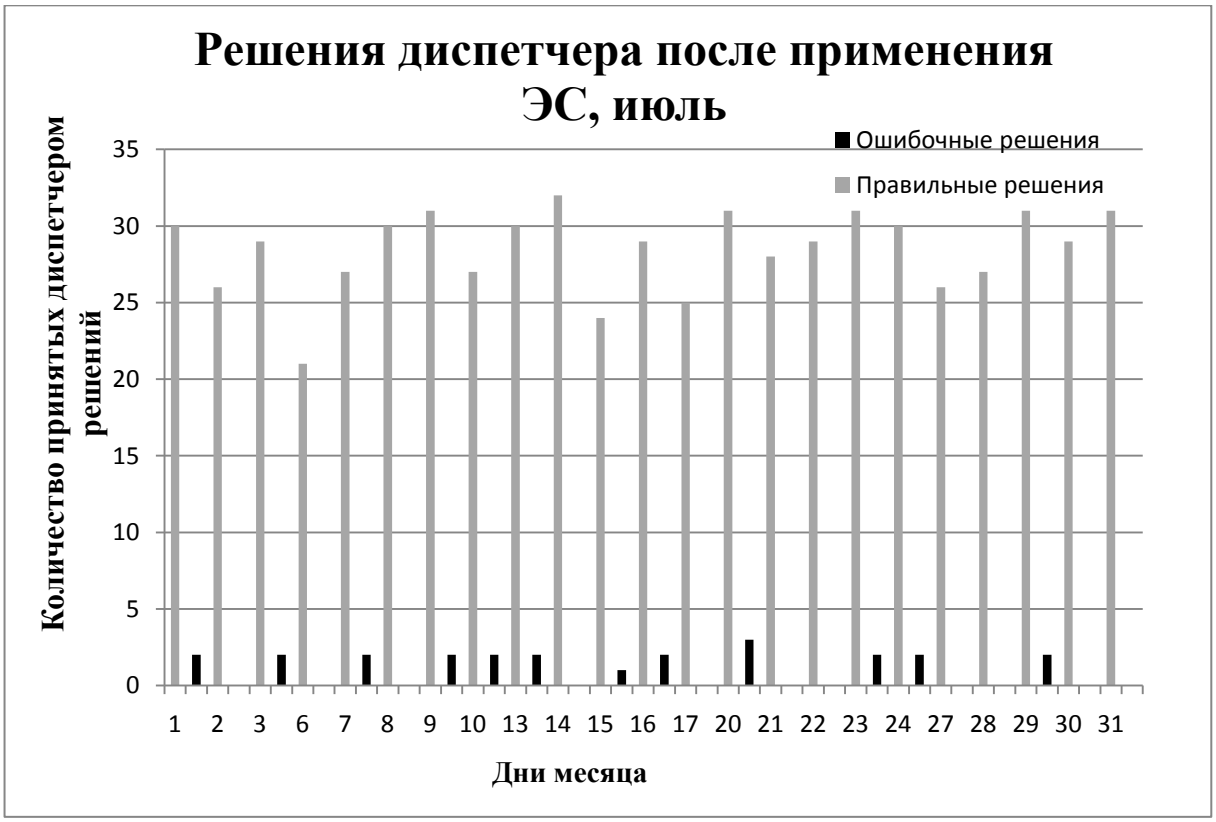

Рисунок 4.25 – Решение диспетчера после внедрения ЭС, июль

Достоверность правильно принятых решений диспетчером до применения ЭС в течении двух месяцев – 0, 85; достоверность правильно принятых решений после применения в течении двух месяцев ЭС - 0, 96 (рис.4.26). Таким образом достоверность принимаемых диспетчером решений увеличилась на 12%: (0, 96:0, 85 -1)*100=12.

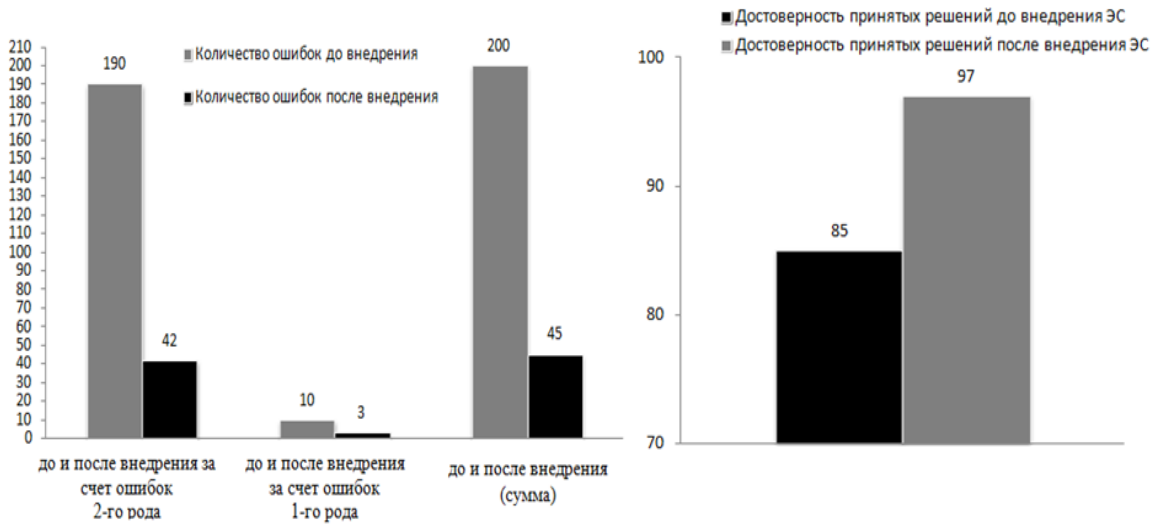

Рисунок 4.26 –Достоверность принятых решений до и после внедрения ЭС

На рис. 4.27, 4.28 изображена статистика несчастных случаев на предприятии до внедрения ЭС и после внедрения.

Количество несчастных случаев до применения ЭС системы в течении двух месяцев – 16:43=0,37; количество несчастных случаев после применения в течении двух месяцев – 13:45=0,28. Таким образом, количество несчастных случаев уменьшилось при выполнении работ в течение нескольких месяцев наблюдения на 32%: (0, 37:0, 28 -1)*100=32.

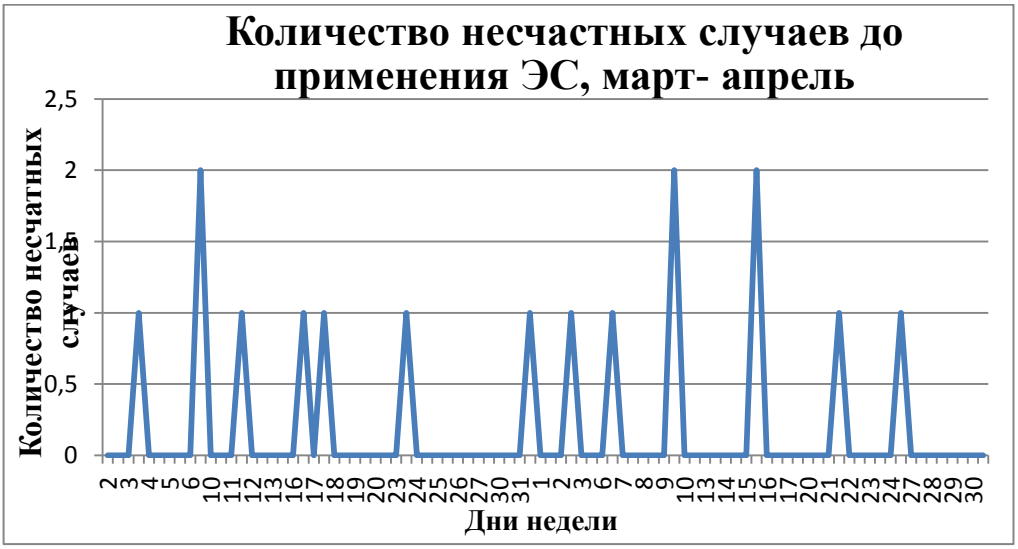

Рисунок 4.27 –Количество несчастных случаев до применения ЭС

Рисунок 4.28 – Количество несчастных случаев после применения ЭС

4.5 Выводы по четвертому разделу

Предложенные модели и методы легли в основу разработанной информационной технологии создания и сопровождения знаниеориентированных систем контроля с использованием инструментального средства в виде редактора правил для создания, проверки и поддержки правил.

1. На первом этапе предложено, автоматизировано составлять словарь предметной области для использования при составлении правил, затем выявлять входные данные в виде параметров и их значений, а также отслеживаемых состояний объекта, определять, целесообразно ли использовать знаниеориентированные системы для конкретной ПрО.

2. Второй этап предполагает разбиение знаний на группы (подгруппы) в соответствии с отслеживаемым состоянием объекта, построение правил для каждой группы (подгруппы) на естественном языке в текстовом виде либо виде И/ИЛИ-графа, разбиение объектов и состояний на терминальные и промежуточные, а также проверку правил итеративно с использованием разработанного редактора правил.

3. На третьем этапе производится автоматизированное преобразование правил, представленных в виде И/ИЛИ-графа, в язык продукционного программирования для конкретной платформы с учетом эффективности их дальнейшей работы.

4. Четвертый этап предполагает программирование знаниеориентированной системы на основе правил, полученных на предыдущем этапе, и включает настройку машины вывода, реализацию отдельных групп (подгрупп) и объединение их в единую модульную структуры, разработку компонентов для получения параметров и графического интерфейса пользователя и их интеграцию с машиной вывода и базами правил.

5. На пятом этапе производится отладка для каждой группы и всего набора правил, предлагаются способы тестирования.

6. Шестой этап предполагает сопровождение системы и изменение правил контроля с помощью редактора правил.

7. Рассматривается разработанное инструментальное средство для создания и поддержки правил знаниеориентированной системы контроля, названное редактором правил. Описываются функции системы в виде вариантов использования, строится концептуальная модель классов, предлагаются типовые шаблоны проектирования, описывается их применение при разработке. Проводится выбор инструментов разработки, а именно, языка программирования, интегрированной среды, системы контроля версий, сторонних библиотек.

8. Показано применение разработанной ИТ при создании двух ЭС контроля: работы компьютерной сети и выполнения правил безопасности при работе с электроустановками.

9. Проведены эксперименты по использованию редактора правил и ЭС для обучения студентов, а также ЛПР, таких как диспетчеры по безопасной работе с электроустановками и системные администраторы компьютерных сетей. При проверке на противоречивость правил ошибки появлялись и исправлялись в среднем в 10% правил, при проверке на достижимость состояний и избыточность удалось удалить в среднем 5% неверных правил, при проверке полноты база правил была увеличена в среднем на 12%. Количество студентов, вовремя сдавших курсовые работы, увеличилось при использовании редактора правил в среднем на 8%. Время обучения ЛПР удалось уменьшить по сравнению с традиционной методикой обучения, основанной на лекциях и проверке знаний на основе тестов, на 20% без изменения качества обучения.

10. Предложено использовать ЭС по безопасной работе с электроустановками в составе АРМ диспетчера организации, эксплуатирующей электрические сети, вместе с системой визуализации объектов в помещениях, где проводятся работы, на основе сенсорных датчиков. Рассмотрена возможность применения такого рода ЭС вместе с системами машинного зрения, предло-

жены подходы к их созданию. ЭС система применяется при контроле и мониторинга безопасности выполнения работ с электроустановками. Проведены эксперименты: в течение 2 месяцев оценивалась достоверность контроля с помощью расчета процента правильно принятых диспетчером решений до внедрению ЭС и 2 месяца после и сделан вывод, что достоверность контроля увеличилась в среднем на 12%, в том числе, за счет ошибок второго рода – на 11% и первого рода на – 1%, а количество несчастных случаев за тот же период времени уменьшилось на 32%.

Результаты исследований раздела 4 опубликованы в работах автора [71, 72, 73, 75, 77, 79].

ВЫВОДЫ

В диссертации разработаны и научно обоснованы новые модели, методы и ИТ разработки зняниеориентированных систем контроля. Основные результаты работы следующие:

1. По результатам анализа существующих систем контроля и мониторинга с целью безаварийного функционирования объектов и необходимостью оперативного внесения изменений в правила контроля показана целесообразность комплексного анализа объектов и разработки знаниеориентированных систем на основе правил. Но из-за противоречий между возможностями интеллектуальных систем, основанных на знаниях, и несовершенством поддержки ведения БЗ и разработки систем сделан вывод о том, что актуальным является автоматизация процесса создания и поддержки знаниеориентированных систем контроля.

2. Определение особенностей структуры знаний для контроля позволило разбить правила на группы, в выводах которых присутствует одно отслеживаемое состояние объекта. Приведенный способ позволяет обрабатывать правила на ранних этапах проектирования.

3. Предложенная специальная разметка И/ИЛИ-графа в качестве модели группы правил контроля для построения, визуализации, интерактивной работы с правилами дает возможность обработки правил методами теории графов и математической логики, а также разработки методов проверки правил.

4. С использованием предложенной модели правил контроля в виде булевых выражений, которые представляют «прямой» и «инверсный» наборы правил для аварийных и нормальных состояний объектов, разработаны методы проверки правил на противоричивость, полноту и достижимость состояний.

5. В результате разработанного метода проверки противоречивости посылок правил контроля на основе задачи SAT становится возможной проверка противоречий между посылками каждого правила и между правилами;

при проведении экспериментов такого рода ошибки обнаруживались и исправлялись в среднем в 10% правил.

6. Предложенный метод проверки правил контроля на полноту на основе визуализации для эксперта «инверсных» правил позволяет эксперту оценить, каких правил не хватает, и интерактивно дополнить базу недостающими знаниями; при проведении экспериментов база знаний для пополнения ее недостающими правилам была увеличена в среднем на 12%.

7. Создание метода проверки достижимости в И/ИЛИ-графе вершин состояний объекта контроля позволило на этой основе осуществлять поиск и удаление компонент связности, не содержащие такого рода вершин; при проведении экспериментов удалось изъять в среднем 5% правил, содержащих ошибки такого рода.

8. При применении разработанного редактора правил контроля для обеспечения их создания и редактирования, возможно, визуализировать правила, интерактивно их разрабатывать и проверять; предложенная ИТ для создания и сопровождения знаниеориентированных систем контроля и редактор правил дали возможность разработки и проверки правил на ранних этапах, что обеспечило оперативное изменение правил за несколько часов, в то время как внесение изменений в программный код занимает не менее нескольких суток.

9. Разработанный на базе ИТ действующий прототип ЭС контроля безопасной работы с электроустановками содержит 54 правила в пяти подгруппах, а действующий прототип ЭС контроля за работой компьютерной сети - 32 правила в трех подгруппах.

10. Созданные ЭС применяются на первом этапе для обучения, на втором - для помощи при принятии решений по результатам контроля. ЭС дает возможность уменьшить время обучения по сравнению с традиционными методиками на 20%. При проведении экспериментов на втором этапе при использовании для безопасной работы с электроустановками достоверность принятых диспетчером решений увеличилась на 12% по сравнению с перио-

дом, когда диспетчер принимал решения без ее помощи, а количество несчастных случаев за тот же период времени уменьшилось на 32%.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Anderson, J.R. Cognitive psychology and intelligent tutoring [Text] / J.R. Anderson // In Proceedings of the Cognitive Science Society Conference. – Colo., June 28 – 30 1984. – P. 37– 43.
2. A Standard Smart Transducer Interface – IEEE 1451 [Electronic resource] / Robert Johnson, Kang Lee, James Wiczer, Stan Woods Sensors Expo, Philadelphia, Oct. 2001 – access mode:

http://ieee1451.nist.gov/Workshop_04Oct01/1451_overview.pdf – 14.05.2015 –title from the screen
3. Bacharach, J. Localization in Sensor Networks [Text] / J. Bachrach, C. Taylor. – Handbook of Sensor Networks, 2005. – 38p.
4. Cacti® - The Complete RRDTool-based Graphing Solution [Electronic resource] – access mode: www/URL: http://www.cacti.net/– 17.03.2014 – title from the screen
5. Davies, E.R. Machine Vision : Theory, Algorithms, Practicalities. [Text] / E.R. Davies. – Morgan Kaufmann, 2004. – 934p
6. Easy, proactive monitoring of processes, programs, files, directories, file systems and hosts / Monit [Electronic resource] – access mode: www/URL: https://mmonit.com/monit/ – 8.05.2014 – title from the screen
7. George, K. Zipf. Human Behavior and the Principle of Least-Effort [Text] / K. Zipf. George. – Addison-Wesley, P. 268–270.
8. Homepage of Zabbix :: An Enterprise-Class Open Source Distributed Monitoring Solution [Electronic resource] – access mode: www/URL: http://www.zabbix.com/ru/ – 23.04.2014 – title from the screen
9. Icinga / Open Source Monitoring [Electronic resource] – access mode: www/URL: https://www.icinga.org// – 05.15.2014 – title from the screen
10. Intel® Core™ Desktop 4th Gen Processor Family: Thermal Guide [Electronic resource] – access mode: http://goo.gl/LAFHYW –15.05.2015 – title from the screen

11. Lavlu, S. M. Cacti 0.8 Network Monitoring [Text]/ S. M. Lavlu. – Packt Publishing, 2009. – 132p.

12. Luediger, H. An ultra-wideband approach towards autonomous radio control and positioning systems in manufacturing & logistics processes [Text] / H. Luediger, B. Kull, M. D. Perez-Guirao// in Proc. 4th Workshop on Positioning, Navigation and Communication. – Hannoversche Beitrage zur Nachrichtentechnik, 2007. – vol. 0.4. – P. 291-296

13. Meseguer, P Assessing the Role of Formal Specifications in Verification and Validation of Knowledge-Based Systems [Text] / P. Meseguer, A.D. Preece // Proceedings of the Third International Conference on Achieving Quality in Software.– London (England), 1996. – P. 317-328.

14. Nagios - The Industry Standard in IT Infrastructure Monitoring [Electronic resource] – access mode: www/URL: https://www.nagios.org/ – 15.06.2014 – title from the screen

15. NetXMS [Electronic resource] – access mode: www/URL: http://www.netxms.org/– 24.07.2014 – title from the screen

16. Olups, Rihards Zabbix 1.8 Network Monitoring [Text]/ Rihards Olups. – Packt Publishing, 2010. – 428p.

17. Preece, A.D. Foundation and Application of Knowledge Base Verification [Text] / A.D. Preece, R. Shinghal // International Journal of Intelligent Systems, 1994. – Vol. 9. – P. 683-701.

18. RRDtool – About RRDtool [Electronic resource] – access mode: www/URL : http://oss.oetiker.ch/rrdtool/ – 10.02.2014 – title from the screen

19. Stephen, A. Cook The Complexity of Theorem-Proving Procedures [Text] / A. Cook Stephen // Proceedings of the third annual ACM symposium on Theory of computing. STOC '71. – New York, NY, USA ©1971 – P. 151-158.

20. Swamy, N. Algorithms for Networked Control and Communication Systems [Text]. PhD Thesis, Dept. of Elect. Eng. / N. Swamy – Texas: USA The University of Texas at Arlington, 2003. – 165p.

21. Tennina, S. Distributed and Cooperative Localization Algorithms for WSNs in GPS-less Environments [Text] / F. Graziosi, F. Santucci, Renzo Di. – M.:ATTI of Italian Navigation Institute, July 2008. – P.870 – 876

22. Troynina, A. Rules of Expert System for Safety Monitoring: Checking on Completeness and Consistency [Текст] /A. Troynina, V. Ruvinska, E. Berkovich, A. Bilovzorov // Праці Одеського політехнічного університету, 2015. – Вип.2(46). – С. 103 – 110.

23. Troynina, A. Wsparcie informatyczne procesów podejmowania decyzji podczas obsługi kontenerów chłodniczych w portach morskich [Текст] / A. Troynina, L. Filina-Dawidowicz // Logistyka , 2015. – № 4. – S. 3181-3191.

24. Urban, Thomas Cacti 0.8 Beginner's Guide [Text]/ Thomas Urban. – Packt Publishing, 2011. – 348p.

25. Wang, Yu Topology Control for Wireless Sensor Networks. In Wireless Sensor Networks and Applications [Text] / Yu Wang. – Springer: US, Feb. 2008. – P. 113–147

26. ZigBee Standards Organisation: ZigBee Specification[Electronic resource], December 2007 – access mode: www.zigbee.org – title from the screen

27. Автоматизированные системы мониторинга судоходства [Текст] / [А.Н. Маринич, И.Г. Проценко, В.Ю. Резников, Ю.М. Устинов, А.Р. Шигабутдинов]; под общ. ред. д.т.н., проф. Ю.М. Устинова. – СПб: Судостроение, 2003. – 248 с.

28. Адаменко, А. Логическое программирование и Visual Prolog. [Текст] / А. Адаменко, А. Кучуков – СПб.: БХВ-Петербург, 2003. – 992 с.

29. Айвазян, С.А. Прикладная статистика и основы эконометрики [Текст] / С.А. Айвазян, В Мхитарян. – М.: ЮНИТИ, 1998. – 1000 с.

30. Алгоритмы. Построение и анализ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; пер. с англ. – М.: «Вильямс», 2014. –1328 с.

31. Асаи, К. Прикладные нечеткие системы [Текст] / Асаи К., Ватада Д., Иван С.; под ред. Т. Тэрано, К. Асаи, М. Сугэно. – М.: Мир, 1993. – 368с.

32. Большакова, Е.И. Искусственный интеллект: методы и алгоритмы эвристического поиска [Текст] / Большакова Е.И., Мальковский М.Г., Пильщиков В.Н. – М.: МГУ, 2002. – 84 с.

33. Братко, И. Язык Prolog:алгоритмы искусственного интеллекта [Текст] / И. Братко. – М.: «Вильямс», 2001. – 640 с.

34. Бровкина, Н.Д. Контроль и ревизия: учеб. пособие. [Текст] / Н.Д. Бровкина, – Москва: ИНФРА-М, 2007. – 346 с.

35. Гаврилова, Т.А. Базы знаний интеллектуальных систем [Текст] / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2005г. – 384с.

36. Гладун, В.П. Процессы формирования новых знаний [Текст] / Виктор Поликарпович Гладун. – София: СД Педагог, 1994. – 192 с.

37. Грекул, В.И. Проектирование информационных систем [Текст] / Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. – М.: БИНОМ , 2005. – 304 с.

38. Дал, У. Структурное программирование [Текст] / Дал У., Дэйкстра Э., Хоар К. – М.: Мир, 1975. – 248 с.

39. Джаратанно, Джозеф Экспертные системы: принципы разработки и программирования [Текст]. 4-е издание.: Пер.с англ. / Джозеф Джаратанно, Гари Райли. – М.: ООО «И. Д. Вильямс», 2007. – 1152 с.: ил. –Парал. тит. англ.

40. Джексон, Питер Введение в экспертные системы [Текст] / Питер Джексон. – М.: «Вильямс», 2001. — 624 с.

41. Джошуа, Блох Effecting Java Programmig Language Guide [Текст] / Блох Джошуа – Лори: Java из первых рук, 2014. – 294с. – ISBN 978 – 5 – 85582– 347 – 9

42. Долинина, О.Н. Применение методов технической диагностики для отладки баз знаний экспертных систем [Текст] / О.Н. Долинина, А.К. Кузьмин. – Вестник СГТУ, 2008. – № 2 (33). – С. 266-272.

43. Зайцева, Т.В. Применение экспертной системы контроля знаний “REXPERT” в учебном процессе [Текст] / Т.В. Зайцева, Н.Н. Смородина, Н.В. Васина. – Научные ведомости, 2013. – № 22(165). – вып. 28/1 – С. 231 – 235.

44. Иванов, А.С. Математические модели и алгоритмы функционирования продукционных баз знаний [Текст]: автореф. дис. на получение научн. степени к. ф.-м. н.: спец. 05.13.18/ Иванов Александр Сергеевич; Сарат. гос. техн. ун-т. – Саратов, 2007. – 20 с.

45. Кватрани, Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. [Текст] / Т. Кватрани – М.: ДМК-Пресс, 2001. – 176 с.

46. Кирпичев, И.Г. Разработка методологии построения и функционального развития информационно-аналитической системы мониторинга жизненного цикла компонентов воздушных судов [Текст]: автореф. дис. на получение научн. степени к.т.н.: спец. 05.02.22/ Кирпичев, Игорь Геннадьевич; Московском Государственном техническом университете гражданской авиации – Москва, 2006. – 20с.

47. Колышев, Л. К. Основы теории нечетких множеств: учебное пособие. [Текст] / Л. К. Колышев, Д. М. Назаров. – Спб:Питер, 2001. –192с.

48. Кофман, А. Введение в теорию нечетких множеств: Перевод с англ. [Текст] / А. Кофман. – М.: радио и связь, 1982. – 432 с.

49. Кунгурцев, А.Б. Метод построения словарей предметных областей для извлечения фактов на естественном языке [Текст] / А.Б. Кунгурцев, С.Н. Бородавкин // Восточно-Европейский журнал передовых технологий, 2010. – Т. 1, №4 (43). – С. 32 – 36.

50. Куценко В.П. Микроволновая экспертная система контроля температуры в стекловаренной печи [Текст] / В.П. Куценко. – Искусственный интеллект, 2014. – № 3. – С. 155–161.

51. Леоненков, А.Б. Нечеткое моделирование в среде Matlab и fuzzyTech [Текст] / А.Б. Леоненков. – Спб:БХВ – Петербург, 2005. – 736с.

52. Ляшевская, О.Н. Частотный словарь современного русского языка на материалах Национального корпуса русского языка [Текст] / О.Н. Ляшевская, С.А. Шаров. – М.: Издате льский центр «Азбуковник», 2009. —1087 с.

53. Онлайн энциклопедия Glossary [Электронный ресурс] /Служба тематических толковых словарей – Режим доступа: URL: www.glossary.ru– 15.06.2014 – Загл. с экрана

54. Орлов, А.И. Прикладная статистика [Текст] /Александр Иванович Орлов. – М.: Экзамен, 2006.– 672 с.

55. Особенности режимных тренажеров диспетчера энергосистем, разработанных в России, Европе и США [Электронный ресурс] / Ежемесячный производственно-массовый журнал "Энергетик" № 9. – 2013. – С. 47-51. – Режим доступа: http://www.monitel.ru/company/articles/203-osobennosti-rezhimnykh-trenazherov-dispetchera-energosistem-razrabotannykh-v-rossii-evrope-i-ssha.html – 24.06.2014 – Загл. с экрана

56. Положение об экспертной системе контроля и оценки состояния и условий эксплуатации воздушных линий электропередачи напряжением 110 кВ и выше: РД 153-34.3-20.524-00/ РАО “ЕЭС России” от 24.02.2000 № 100

57. Поспелова, Л.Я. Поиск противоречий в продукционных базах знаний [Текст] / Л.Я. Поспелова, О.В. Чуканова // Научн. сессия МИФИ2009. Сб. научн. трудов. Т. V. Информационно-телекоммуникационные системы. – М.: НИЯУ МИФИ, 2009. – С. 23–27.

58. Правила технической эксплуатации электроустановок потребителей — [Дійсні від 2006—25—07]. — К. : Міністерства палива та енергетики України, 2006. — № 258 — (Національні стандарти України). — Текст: рос., укр.

59. Проверка графа на ацикличность и нахождение цикла [Электронный ресурс] – Режим доступа: http://e-maxx.ru/algo/finding_cycle – 12.06.2015 – Загл. с экрана

60. Программный пакет синтаксического разбора и машинного перевода. [Электронный рессурс] – Режим доступа : http://cs.isa.ru:10000/dwarf/ – 15.06.2013 – Загл. с экрана

61. Проектирование пользовательского интерфейса на персональных компьютерах. Стандарт фирмы IBM / Под ред. Дадашова. – М. – Вильнюс: DBS LTD., 1992. – 180 с

62. Рожкин, Б.В. Экспертная система контроля работоспособности рельсовых цепей [Текст] / Рожкин Б.В. // Проблемы управления эксплуатационной работой на железнодорожном транспорте; развитие телекоммуникаций и информатизации: Материалы научно-технической конференции, посвященной 125-летию Свердловской железной дороги. –Екатеринбург, 2003. – Т.2. – УрГУПС.

63. Савельев, А.Я. Основы информатики [Текст] / А. Я. Савельев; Информатика в техническом университете. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2001. – С. 232 –239. – ISBN 5703815150

64. Смирнов, В.В. Методы и средства верификации баз знаний в интегрированных экспертных системах [Текст]: автореф. дис. на получение научн. степени канд. техн. наук.: спец. 05.13.11 / Смирнов Виталий Валерьевич. – Москва, 2006. – 20 с.

65. Советчик Диспетчера по ликвидации перегрузок [Электронный ресурс]: по данным Научно производственного центра «Приоритет»// Москва – 28 июля 2009. – Режим доступа: http://www.priortelecom.ru/Software/sovet.htm – 10.06.2015 – Загл. с экрана

66. Статические и динамические системы экспертные системы [Текст] : учеб. пособие для втузов, обуч. по спец.: Прикладная математика; Автоматизированные системы обработки информации и управления / Э.В.Попов, И.Б.Фоминых, Е.Б.Кисель, М.Д.Шапот. М.: Финансы и статистика, 1996. – 319 с.

67. Суботін, С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень [Текст] / С.О. Суботін – Запоріжжя: ЗНТУ, 2008. – 341с.

68. Тоценко, В.Г. Методы и системы поддержки принятия решений [Текст] / В.Г. Тоценко. – Киев: Наукова думка, 2002. –382 с.

69. Тройнина, А.С. 3D-визуализация положений датчиков для принятия решений [Текст] / Тройнина А.С., Рувинская В.М., Туршатов И.А.// Вісник Східноукраїнського національного університету імені Володимира Даля, 2011. – №13 (167). – С. 189–194.

70. Тройнина, А.С. Методика построения экспертных систем для мониторинга [Текст] / А.С. Тройнина, В.М. Рувинская // Радіоелектронні і комп'ютерні системи, 2012. – №6(58). – С. 276 – 280.

71. Тройніна, А.С. Редактор знань для експертних систем моніторингу [Текст] / А.С. Тройніна, В.М. Рувінська, М.С. Ніколенко // Вестник Херсонского национального технического университета (ХНТУ), 2013. – №1(46) – С. 183– 185.

72. Тройнина, А.С. Автоматизация проверки правил ЭС для мониторинга работы компьютерной сети [Текст] / А.С. Тройнина, В.М. Рувинская, Е.Л. Беркович, А.Ю. Черненко //Электротехнические и компьютерные системы. – 2014. – №14(90). – С.94 – 104.

73. Тройнина, А.С. Экспертная система по безопасной работе с электроустановками [Текст] / А.С. Тройнина, В.М. Рувинская, Л.В. Беркович // Труды двенадцатой международной научно-практической конференции «Современные информационные и электронные технологии». – Одеса, 23-27 мая 2011 г. – С.78.

74. Тройнина, А. С. Объединение групп правил при работе экспертной системы мониторинга [Текст] / А.С. Тройнина // Труды тринадцатой международной научно-практической конференции «Современные информационные и электронные технологии». – Одесса, 4-8 июня 2012 г. – С. 133.

75. Тройніна, А.С. Мониторинг и анализ показателей работы компьютерной сети на основе баз знаний [Текст] / А.С. Тройнина, В.М. Рувинская // Сучасні інформаційні технології: Матеріали четвертої міжнародній конференції студентів і молодих науковців (МІТ-2014). – Одеса, 2014. – С.42 – 43.

76. Тройнина, А.С. Словарь предметной области для разработки экспертной системы [Текст] / А.С. Тройнина, В.М. Рувинская, Д.А. Силяев// Труды Меж-

дународной научно технической конференции «Информационные технологи в металлургии и машиностроении». – Днепропетровск, 24– 26.03.2015. – С. 44.

77. Тройніна, А.С. Експертна система для аналізу даних мережевого моніторингу [Электронный ресурс] / А.С. Тройніна, В.М. Рувінська, О.О. Біловзоров //Труди Міжнародної науково-практичної Інтернет-конференції Молодь в технічних науках: дослідження, проблеми, перспективи. – Вінниця. – Режим доступа: http://conf.inmad.vntu.edu.ua/fm/index.php?page=materials&line=11&mat=188 / – 23 – 26 квітня 2015р. – Загл. с экрана.

78. Тройнина, А.С. Экспертные системы в качестве тренажера для обучения ЛПР [Текст] / А.С. Тройнина // Труди першої міжнародної конференції з адаптивних технологій управління навчанням –2015. – Одеса, 23–25.09.2015. – С.147 – 148.

79. Тройнина, А.С. 3D-визуализация положений датчиков для принятия решений [Текст] / А.С. Тройнина, В.М. Рувинская, И.А. Туршатов// I Міжнародна науково-технічна конференція «Обчислювальний інтелект (OI-2011)». – Черкаси, Україна, 2011р.– С. 367-368

80. Хает, С.И. Разработка и реализация элементов диагностического модуля для мониторинга состояния конденсационной установки паровой турбины [Текст]: дис. на получение научн. степени к. т. н.: спец 05.04.12 / Хаета Станислава Иосифовича; Ур. гос. техн. ун-т. – Екатеринбург, 2004. – 147 с.

81. Харисова, А. Р. Разработка нечеткой экспертной системы диагностики и мониторинга состояния оборудования [Текст]: дис. на получение научн. степени к. т. н.: спец 05.13.06 / Харисова Азамата Робертовича; Ур. гос. техн. ун-т. – Екатеринбург, 2007. – 144 с.

82. Экспертная система мониторинга, диагностики и управления ЭСМДУ-ТРАНС [Электронный ресурс] – Режим доступа: http://www.ztr.com.ua/files/ztr_d69-esmdy-trans--2014.pdf – 15.06.2015 – Загл. с экрана

83. Энциклопедический словарь-справочник. [Электроонный рессурс] / Под редакцией А. И. Половинкина, В. В. Попова. – Режим доступа: http://doc.unicor.ru/tt/126.html – 27.06.2015 г. – Загл. с экрана.

84. Язловецкий, Я.С. Сравнительный анализ экспертных систем контроля качества обслуживания в сетях передачи данных [Текст] / Я.С. Язловецкий, Л.Н. Величко. – Веснік сувязі, 2015. – № 2(130). – С. 49–54

85. Яловец, А.Л. Представление и обработка знаний с точки зрения математического моделирования. Проблемы и решения. [Текст]/ А.Л. Яловец / Киев, Наукова думка НАН Украины, 2011, с. 359.

ПРИЛОЖЕНИЕ А. Группы правил для двух предметных областей

ПрО «Контроль безопасной работы с электроустановками

Группа правил «Работа с приборами учета»

Если напряжение от 42в до 1000в и работа – замена прибора учета, и нет индикатора напряжения или нет диэлектрических перчаток, или нет инструмента с изолирующими рукоятками, то работу выполнять нельзя.

Если напряжение свыше 1000в и работа – замена прибора учета, и нет указателя напряжения или нет изолирующей штанги, то работу выполнять нельзя.

Если тип работы – снятие прибора учета трансформаторного включения и нет наряда на выполнение работы, то работу выполнять нельзя.

Если тип работы – снятие прибора учета трансформаторного включения и число работников в бригаде < 2 или не оформлен наряд, или не оформлено распоряжение, то работу выполнять нельзя.

Если тип работы – снятие прибора учета трансформаторного включения и квалификация каждого работника бригады < 4 или не оформлен наряд, или не оформлено распоряжение, то работу выполнять нельзя.

Если тип работы – снятие прибора учета непосредственного включения и квалификация работника < 3или не оформлено распоряжение, то работу выполнять нельзя.

Если тип работы – снятие показаний прибора учета или электроизмерительного прибора и число работников = 1, и квалификация работника < 3, то работу выполнять нельзя.

Группа «Защитные средства»

Если напряжение свыше 1000в и нет одного из дополнительных защитных средств, необходимых для данной работы, то работу выполнять нельзя.

Если не дата следующего испытания любого ЭЗС (Электрозащитные средства) из одного из двух списков, представленных ниже > CurrentDate, то работы выполнять нельзя.

Если невозможно установить ограждение токоведущих частей и вместо ограждения используются изолирующие накладки и число работников <2 или группа квалификации каждого работника<3, то работу выполнять нельзя.

Группа «Работы в охранной зоне электрических сетей»

Если объект находится в охранной зоне электрических сетей (ЭС), и работа должна проводиться по согласованию с ЭСО (Электро-снабжающая организация) и нет согласования, то работу выполнять нельзя.

Если должны производиться строительные работы, и место проведения работ находится в охранной зоне ЭС, то без согласования выполнения этих работ с организацией, эксплуатирующей ЭС, и получения разрешения на их выполнение работы выполнять нельзя.

Если должны производиться погрузочно-разгрузочные работы и место проведения работ находится в охранной зоне ЭС, то без согласования выполнения этих работ с организацией, эксплуатирующей ЭС, и получения разрешения на их выполнение работы выполнять нельзя.

ПО «Контроляь и мониторинг компьютерной сети»

Подгруппа правил «Сеть»

1. Если связь между узлами сети нестабильна, или наблюдается перегрузки сети, то налицо проблема, требующая внимания системного администратора.
2. Если потеря пакетов высока и производительность узла сети низкая, связь с узлом сети нестабильна.
3. Если количество запросов низкая, и объем трафика низкий, и время обработки запроса высокий, то производительность узла сети низкая.
4. Если потеря пакетов высокая, и нагрузка на процессор низкое, и наблюдается перегрузки узла сети, то канал связи с узлом сети перегружен.
5. Если время обработки запроса высокое, и объем трафика нормальный, и объем трафика нормальный, то наблюдается перегрузки узла сети.

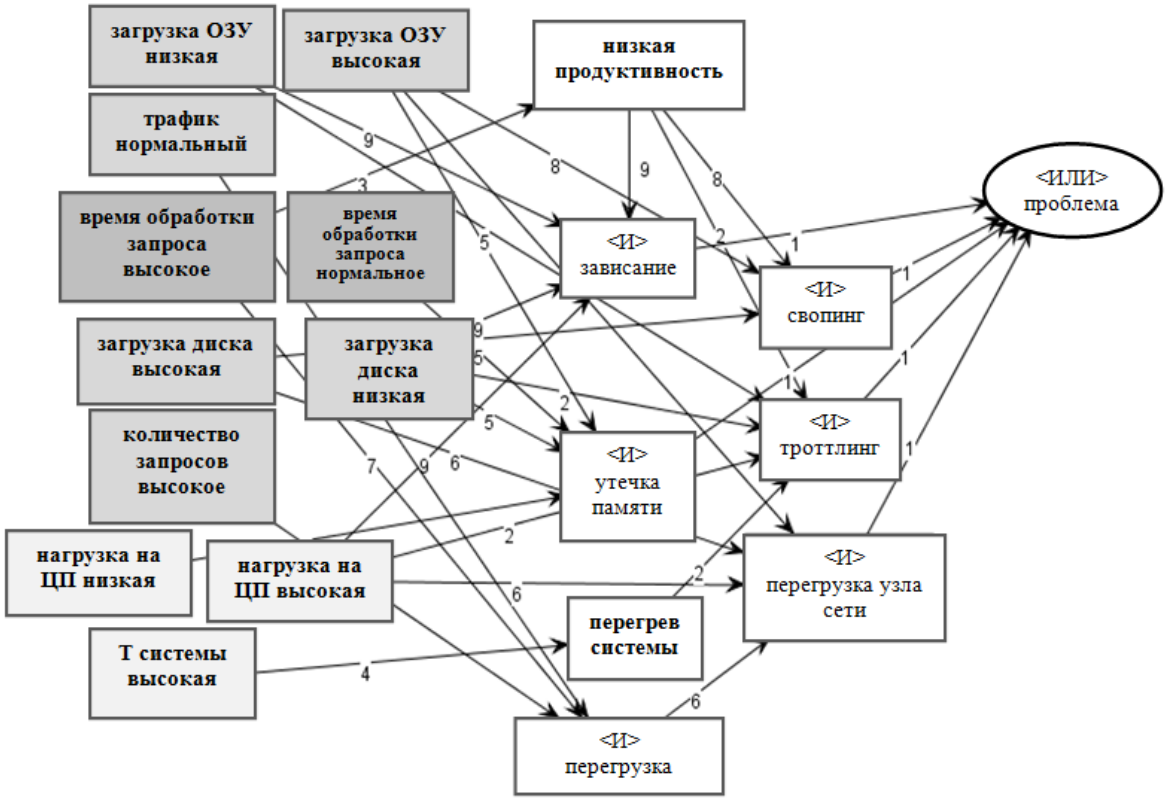

Рисунок А.1 – И/ИЛИ-граф для подгруппы правил «Узел сети»

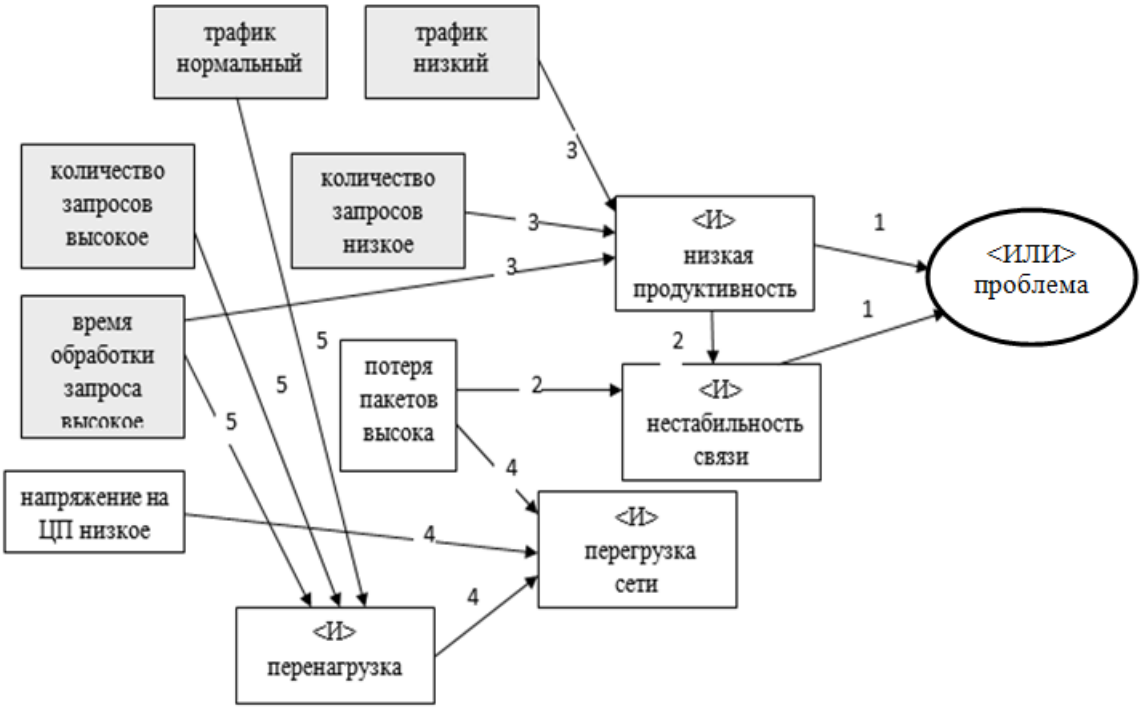

Рисунок А.2 – И/ИЛИ-граф для подгруппы правил «Сеть»

ПРИЛОЖЕНИЕ Б. Программные модули для двух ЭС

Для ЭС безопасной работе на электроустановках

```
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
        (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer)
        then
         (bind ?answer (lowcase ?answer))))
     ?answer)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question yes  no n))
(if (or (eq ?response yes) (eq ?response y))
then
TRUE
else
FALSE))

(deffunction ask_common_question (?question)
   (printout t ?question)
   (bind ?answer (read))
    ?answer
)
(deftemplate av  (field a) (field v))


(defrule dop-one
  (declare (salience -1))
 (not (av (a the_mark_of_target_instruction_presence   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "For each employee there is the mark of the target instruction (yes/no) ? ")
then
(assert (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction)))
else
(assert (av (a the_mark_of_target_instruction_presence) (v
not_every_employee_has_a_mark_of_the_target_instruction)))))

(defrule base-six
    (av (a the_mark_of_target_instruction_presence) (v
not_every_employee_has_a_mark_of_the_target_instruction))
     =>
(assert (result "    Not permitted")))

(defrule dop-two
```

```
  (declare (salience -1))
 (not (av (a fact_checing) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "For each team's  workers  the next protection of labour knowledge checking date less
WorkEnd_Time  (yes/no) ? ")
then
(assert (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time)))
else
(assert (av (a fact_checing) (v
not_for_each_teams_worker_the_next_checking_date_of_labour_protection_knowledge_less_WorkEnd_Time)))))

(defrule base-seven
    (av (a fact_checing) (v
not_for_each_teams_worker_the_next_checking_date_of_labour_protection_knowledge_less_WorkEnd_Time))
     =>
(assert (result "    Not permitted")))

(defrule dop-three
  (declare (salience -1))
  (not (av (a work_range) (v ?)))
  (not (result ?))
=>
(assert (av (a work_range) (v (ask-question  "What range is this work (by_work_order or by_direction or
by_technical_service)? "  by_work_order by_direction by_technical_service)))))


(defrule dop-four
 (declare (salience -1))
  (av (a work_range) (v by_technical_service ))
  (not (av (a the_fact_that_a_work_is_by_direction    ) (v ?)))
  (not (result ?))
  (not (av (a the_fact_that_work_is_by_work_order    ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Direction for work was registered (yes/no) ? ")
then
(assert (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_registered)))
else
(assert (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered)))
)
(if (yes-or-no-p  "Work order was registered (yes/no) ? ")
then
(assert (av (a the_fact_that_work_is_by_work_order ) (v work_order_was_registered )))
else
(assert (av (a the_fact_that_work_is_by_work_order ) (v work_order_was_not_registered)))))

(defrule dop-five
 (declare (salience -1))
 (not (av (a voltage) (v ?)))
  (not (result ?))
=>
(assert (av (a voltage) (v (ask_common_question  "Enter voltage? "))))))
```

```
(defrule base-five

  (av (a voltage) (v  ?n))
  (test (> ?n 1000))
  (av (a work_range) (v by_technical_service))
  (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered))
  (av (a the_fact_that_work_is_by_work_order) (v work_order_was_not_registered))

   =>
  (assert (result "   Not permitted")))

(defrule dop-six
  (declare (salience 0))
  (av (a work_range) (v by_direction ))
; (av (a the_fact_that_a_work_is_by_direction  ) (v ?))
  (not (av (a the_fact_that_a_work_is_by_direction   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Direction for work was registered (yes/no) ? ")
then
(assert (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_registered)))
else
(assert (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered)))))


(defrule base-four
    (av (a work_range) (v by_direction ))
  (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered))
    =>
(assert (result "   Not permitted")))


(defrule dop-seven
  (declare (salience -1))
  (av (a work_range) (v by_direction ))
  (not (av (a of_works_observer) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Is there observer of the work (yes/no) ? ")
then
(assert (av (a of_works_observer ) (v there_is_observer )))
else
(assert (av (a of_works_observer ) (v there_is_no_observer)))))


(defrule base-three
    (av (a work_range) (v by_direction ))
  (av (a of_works_observer) (v there_is_no_observer))
    =>
(assert (result "   Not permitted")))

(defrule dop-eight
  (declare (salience 0))
```

```
 (av (a work_range) (v by_work_order ))
; (av (a the_fact_that_work_is_by_work_order  ) (v ?))
  (not (av (a the_fact_that_work_is_by_work_order  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Work order was registered (yes/no) ? ")
then
(assert (av (a the_fact_that_work_is_by_work_order ) (v work_order_was_registered )))
else
(assert (av (a the_fact_that_work_is_by_work_order ) (v work_order_was_not_registered)))))

(defrule base-one
    (av (a work_range) (v by_work_order))
  (av (a the_fact_that_work_is_by_work_order) (v work_order_was_not_registered))
    =>
(assert (result "   Not permitted")))

(defrule dop-nine
 (declare (salience -1))
 (av (a work_range) (v by_work_order))
 (not (av (a voltage) (v ?)))
  (not (result ?))
=>
(assert (av (a voltage) (v (ask_common_question  "Enter voltage? "))))

(defrule dop-ten
  (declare (salience -1))
  (av (a work_range) (v by_work_order ))
  (not (av (a of_works_observer) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Is there observer of the work (yes/no) ? ")
then
(assert (av (a of_works_observer ) (v there_is_observer )))
else
(assert (av (a of_works_observer ) (v there_is_no_observer)))))


(defrule base-two
   (av (a voltage) (v  ?n))
  (test (> ?n 1000))
  (av (a work_range) (v by_work_order))
  (av (a of_works_observer ) (v there_is_no_observer))
    =>
  (assert (result "   Not permitted")))

(defrule dop-eleven
  (declare (salience -1))
  (av (a work_range) (v by_work_order))
  (not (av (a the_presence_of_the_head_of_of_works_manager) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Is there manager of the work (yes/no) ? ")
then
```

```
(assert (av (a the_presence_of_the_head_of_of_works_manager ) (v there_is_manager )))
else
(assert (av (a the_presence_of_the_head_of_of_works_manager ) (v there_is_not_manager)))))


(defrule base-two'
   (av (a voltage) (v  ?n))
  (test (> ?n 1000))
  (av (a work_range) (v by_work_order))
  (av (a the_presence_of_the_head_of_of_works_manager ) (v there_is_not_manager))

   =>
  (assert (result "   Not permitted")))

(defrule ok-result ""
(declare (salience -10))
(not (result ?))
=>
 (assert (result "    All OK")))
(defrule print-result ""
(declare (salience 10))
(result ?item)
=>
(printoutt          "Suggested Answer:")
(printoutt          crlf)
(format t"          %s%n%n%n" ?item))
```

ЭС для контроля и мониторинга сети

```
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
        (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer)
         then
          (bind ?answer (lowcase ?answer))))
     ?answer)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question yes  no n))
(if (or (eq ?response yes) (eq ?response y))
then
TRUE
else
FALSE)
)

(deffunction ask_common_question (?question)
```

```
   (printout t ?question)
   (bind ?answer (read))
    ?answer)

(deftemplate av  (field a) (field v))


(defrule dop-2-r1
  (declare (salience -1))
  (not (av (a speed_cooling_fan ) (v ?)))
  (not (result ?))
=>
(assert (av (a speed_cooling_fan ) (v (ask-question  "Speed cooling fan  ( s_c_absent  or s_c_normal or s_c_high )? "
s_c_absent s_c_normal s_c_high)))))

(defrule dop-1-r1
  (declare (salience -1))
  (not (av (a temperature_system    ) (v ?)))
  (not (result ?))
=>
(assert (av (a temperature_system  ) (v (ask-question  "Temperature system  ( t_s_low  or t_s_high or t_s_very_high
)? "  t_s_low  t_s_high  t_s_very_high)))))

(defrule dop-3-r1
  (declare (salience -1))
  (not (av (a ambient_temperature    ) (v ?)))
  (not (result ?))
=>
(assert (av (a ambient_temperature  ) (v (ask-question  "Ambient temperature  ( a_t_low  or a_t_high or
a_t_very_high )? "  a_t_low  a_t_high  a_t_very_high)))))



(defrule finel-8
      (and(av (a ambient_temperature)(v a_t_very_high))
      (av (a temperature_system) (v t_s_very_high)))
       (not (result ?))
        =>
(printout t"№8 FIRE!!!!" crlf)
(assert (av (a fire) (v Fire_yes))))

(defrule finel-3
      (or(av (a temperature_system)(v t_s_very_high))
      (av (a temperature_system) (v t_s_high)))
       (not (result ?))
        =>
 (printout t"№3 Overheating of the system !!!!" crlf)
(assert (av (a overheating_of_the_system ) (v overheating_of_the_system_yes))))

(defrule finel-4
      (and(av (a overheating_of_the_system)(v overheating_of_the_system_yes))
      (av (a ambient_temperature) (v a_t_low)))
       (not (result ?))
        =>
```

```
(printout t "№4 Cooling system not effective !!!!" crlf)
(assert (av (a cooling_system_effective ) (v c_s_not_effective))))

(defrule finel-9
      (exists(av (a speed_cooling_fan )(v s_c_absent)))
       (not (result ?))
        =>
(printout t"№9 Cooling system is out of order!!!!" crlf)
(assert (av (a cooling_system_is_out_of_order ) (v c_s_is_out_of_yes))))

(defrule finel-7
      (and(av (a temperature_system)(v t_s_low))
      (av (a speed_cooling_fan ) (v s_c_high )))
       (not (result ?))
        =>
(printout t "№7 Necessary to reduce the speed of the cooling fan!!!!" crlf)
(assert (av (a need_for_action_for_the_c_f ) (v reduce_speed_for_the_c_f))))

(defrule finel-6
      (and(av (a overheating_of_the_system)(v overheating_of_the_system_yes))
      (av (a speed_cooling_fan ) (v s_c_normal)))
       (not (result ?))
        =>
(printout t "№6 Necessary to increase the speed of the cooling fan!!!!" crlf)
(assert (av (a need_for_action_for_the_c_f ) (v increase_turnover_for_the_c_f))))

(defrule finel-5
      (or(av (a need_for_action_for_the_c_f)(v reduce_speed_for_the_c_f))
      (av (a need_for_action_for_the_c_f ) (v increase_turnover_for_the_c_f)))
       (not (result ?))
        =>
 (printout t"№5 Cooling system needs in building!!!!" crlf)
(assert (av (a need_for_action_for_the_c_f) (v the_need_for_settina_tg_for_the_c_f))))

(defrule finel-2
      (and(av (a overheating_of_the_system)(v overheating_of_the_system_yes))
      (av (a ambient_temperature) (v a_t_high)))
       (not (result ?))
        =>
(printout t "№2 Conditioning system is inefficient facilities!!!!" crlf)
(assert (av (a air_conditioning_system) (v a_c_s_not_effective ))))

(defrule finel-1
      (or(av (a fire)(v Fire_yes))
      (av (a cooling_system_is_out_of_order) (v c_s_is_out_of_yes))
      (av (a cooling_system_effective) (v c_s_not_effective))
      (av (a air_conditioning_system) (v a_c_s_not_effective))
      (av (a need_for_action_for_the_c_f) (v the_need_for_setting_for_the_c_f)))
       (not (result ?))
        =>
 (assert (result "№1 There was a problem requiring intervention system administrator!!!!")))

(defrule ok-result ""
  (declare (salience -10))
```

```
  (not (result ?))
  =>
  (assert (result "    All OK")))

(defrule print-result ""
  (declare (salience 10))
  (result ?item)
   =>
  (printout          t          "Suggested Answer:")
  (printout          t          crlf)
  (format t          "          %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
(printout t crlf crlf)
(printout t "*******************************************" crlf)
(printout t "            * Expert System  *" crlf)
(printout t "*******************************************" crlf)
(printout t crlf crlf))
```

ПРИЛОЖЕНИЕ В. Протоколы тестирования ЭС

Протоколы тестирования ЭС по безопасной работе на электроустановках

```
CLIPS> (run)
==> Focus MAIN
FIRE    1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? no
==> f-1     (av (a the_mark_of_target_instruction_presence) (v
not_every_employee_has_a_mark_of_the_target_instruction))
FIRE    2 base-six: f-1
==> f-2     (result "   Not permitted")
FIRE    3 print-result: f-2
Suggested Answer:
            Not permitted
```

Правило 7

```
TRUE
CLIPS> (run)
==> Focus MAIN
FIRE    1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1     (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE    2 dop-two: *,*
For each team's  workers  the next protection of labour knowledge checking date less WorkEnd_Time  (yes/no) ? no
==> f-2     (av (a fact_checing) (v
not_for_each_teams_worker_the_next_checking_date_of_labour_protection_knowledge_less_WorkEnd_Time))
FIRE    3 base-seven: f-2
==> f-3     (result "   Not permitted")
FIRE    4 print-result: f-3
Suggested Answer:
            Not permitted
```

Правило 5

```
TRUE
CLIPS> (run)
FIRE    1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1     (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE    2 dop-two: *,*
For each team's  workers  the next protection of labour knowledge checking date less WorkEnd_Time  (yes/no) ?
yes
==> f-2     (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE    3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_technical_service
==> f-3     (av (a work_range) (v by_technical_service))
FIRE    4 dop-five: f-3,*,*
Enter voltage? 1111
==> f-4     (av (a voltage) (v 1111))
FIRE    5 dop-four: f-3,*,*,*,*
Direction for work was registered (yes/no) ? no
==> f-5     (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered))
Work order was registered (yes/no) ? no
==> f-6     (av (a the_fact_that_work_is_by_work_order) (v work_order_was_not_registered))
FIRE    6 base-five: f-4,f-3,f-5,f-6
==> f-7     (result "   Not permitted")
```

FIRE 7 print-result: f-7
Suggested Answer:
Not permitted
Правило 4
TRUE
CLIPS> (run)
FIRE 1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1 (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE 2 dop-two: *,*
For each team's workers the next protection of labour knowledge checking date less WorkEnd_Time (yes/no) ?
yes
==> f-2 (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE 3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_direction
==> f-3 (av (a work_range) (v by_direction))
FIRE 4 dop-six: f-3,*,*
Direction for work was registered (yes/no) ? no
==> f-4 (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_not_registered))
FIRE 5 base-four: f-3,f-4
==> f-5 (result " Not permitted")
FIRE 6 print-result: f-5
Suggested Answer:
Not permitted
Правило 3
TRUE
CLIPS> (run)
FIRE 1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1 (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE 2 dop-two: *,*
For each team's workers the next protection of labour knowledge checking date less WorkEnd_Time (yes/no) ?
yes
==> f-2 (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE 3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_direction
==> f-3 (av (a work_range) (v by_direction))
FIRE 4 dop-six: f-3,*,*
Direction for work was registered (yes/no) ? yes
==> f-4 (av (a the_fact_that_a_work_is_by_direction) (v direction_for_work_was_registered))
FIRE 5 dop-seven: f-3,*,*
Is there observer of the work (yes/no) ? no
==> f-5 (av (a of_works_observer) (v there_is_no_observer))
FIRE 6 base-three: f-3,f-5
==> f-6 (result " Not permitted")
FIRE 7 print-result: f-6
Suggested Answer:
Not permitted
Правило 1
TRUE
CLIPS> (run)
FIRE 1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1 (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE 2 dop-two: *,*
For each team's workers the next protection of labour knowledge checking date less WorkEnd_Time (yes/no) ?
yes

==> f-2 (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE 3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_work_order
==> f-3 (av (a work_range) (v by_work_order))
FIRE 4 dop-eight: f-3,*,*
Work order was registered (yes/no) ? no
==> f-4 (av (a the_fact_that_work_is_by_work_order) (v work_order_was_not_registered))
FIRE 5 base-one: f-3,f-4
==> f-5 (result " Not permitted")
FIRE 6 print-result: f-5
Suggested Answer:
Not permitted

Правило 2'

TRUE
CLIPS> (run)
FIRE 1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1 (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE 2 dop-two: *,*
For each team's workers the next protection of labour knowledge checking date less WorkEnd_Time (yes/no) ?
yes
==> f-2 (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE 3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_work_order
==> f-3 (av (a work_range) (v by_work_order))
FIRE 4 dop-eight: f-3,*,*
Work order was registered (yes/no) ? yes
==> f-4 (av (a the_fact_that_work_is_by_work_order) (v work_order_was_registered))
FIRE 5 dop-nine: f-3,*,*
Enter voltage? 1111
==> f-5 (av (a voltage) (v 1111))
FIRE 6 dop-eleven: f-3,*,*
Is there manager of the work (yes/no) ? no
==> f-6 (av (a the_presence_of_the_head_of_of_works_manager) (v there_is_not_manager))
FIRE 7 base-two': f-5,f-3,f-6
==> f-7 (result " Not permitted")
FIRE 8 print-result: f-7
Suggested Answer:
Not permitted

Правило 2

TRUE
CLIPS> (run)
FIRE 1 dop-one: *,*
For each employee there is the mark of the target instruction (yes/no) ? yes
==> f-1 (av (a the_mark_of_target_instruction_presence) (v
for_each_employee_there_is_the_mark_of_the_target_instruction))
FIRE 2 dop-two: *,*
For each team's workers the next protection of labour knowledge checking date less WorkEnd_Time (yes/no) ?
yes
==> f-2 (av (a fact_checing) (v
for_each_teams_workers_the_next_protection_of_labour_knowledge_checking_date_less_WorkEnd_Time))
FIRE 3 dop-three: *,*
What range is this work (by_work_order or by_direction or by_technical_service)? by_work_order
==> f-3 (av (a work_range) (v by_work_order))
FIRE 4 dop-eight: f-3,*,*
Work order was registered (yes/no) ? yes
==> f-4 (av (a the_fact_that_work_is_by_work_order) (v work_order_was_registered))
FIRE 5 dop-nine: f-3,*,*
Enter voltage? 1111

```
==> f-5     (av (a voltage) (v 1111))
FIRE    6 dop-eleven: f-3,*,*
Is there manager of the work (yes/no) ? yes
==> f-6     (av (a the_presence_of_the_head_of_of_works_manager) (v there_is_manager))
FIRE    7 dop-ten: f-3,*,*
Is there observer of the work (yes/no) ? no
==> f-7     (av (a of_works_observer) (v there_is_no_observer))
FIRE    8 base-two: f-5,f-3,f-7
==> f-8     (result "   Not permitted")
FIRE    9 print-result: f-8
Suggested Answer:
            Not permitted
```

Таблица В.1 - Тестирование алгоритма «Добавление фактов»

№	Тестовый набор	Желаемый результат	Полученный результат
1	ping = 50, ping_st = 60, q_number = 3000, q_number_st = 4000, traffic = 500, traffic_st = 5000, cpu_load = 40, cpu_load_st = 75, pkg_loss = 0, pkg_loss_st = 5	добавленные факты: “ping low”, “queries number low”, “traffic low”, “cpu load low”, “package lost low”	добавленные факты: “ping low”, “queries number low”, “traffic low”, “cpu load low”, “package lost low”
2	ping = 70, ping_st = 60, q_number = 4500, q_number_st = 4000, traffic = 500, traffic_st = 5000, cpu_load = 80, cpu_load_st = 75, pkg_loss = 7, pkg_loss_st = 5	добавленные факты: “ping high”, “queries number high”, “traffic low”, “cpu load high”, “package lost high”	добавленные факты: “ping high”, “queries number high”, “traffic low”, “cpu load high”, “package lost high”
3	ping = 50, ping_st = 60, q_number = 3000, q_number_st = 4000, traffic = 5500, traffic_st = 5000, cpu_load = 40, cpu_load_st = 75, pkg_loss = 0, pkg_loss_st = 5	добавленные факты: “ping low”, “queries number low”, “traffic high”, “cpu load low”, “package lost low”	добавленные факты: “ping low”, “queries number low”, “traffic high”, “cpu load low”, “package lost low”

ПРИЛОЖЕНИЕ Г. Таблицы для экспериментов с ЭС при обучении студентов

Таблица Г.1 – Контрольная группа

№	Баллы по дисциплине	Оценки по дисциплине	Оценки по курсовой
1	87	4	4
2	75	4	3
3	61	3	3
4	85	4	5
5	80	4	4
6	70	3	4
7	95	5	5
8	100	5	5
9	65	3	3
10	63	3	3
11	98	5	5
12	82	4	4
13	78	4	3
14	60	3	3
15	60	3	3
16	71	3	3
17	66	3	3
18	92	5	4
19	94	5	4
20	69	3	3
21	79	4	3

Таблица Г.2 – Экспериментальная группа

№	Баллы по дисциплине	Оценки по дисциплине	Оценки по курсовой
1	88	4	4
2	99	5	5
3	65	3	3
4	67	3	3
5	84	4	4
6	96	5	5
7	75	4	4
8	77	4	4
9	65	3	3
10	60	3	3
11	81	4	4
12	60	3	4
13	60	3	4
14	80	4	5
15	100	5	5
16	64	3	3
17	100	5	5

Продолжение таблицы Г.2

18	62	3	3
19	96	5	5
20	83	4	4
21	95	5	5

Таблица Г.3 – Вариационный ряд успеваемости в двух группах

№ позиции	1	2	3	4	5	6	7	8	9	10	11
x	3	3	3	3	3	3	3	3	3	4	4
y	3	3	3	3	3	3	3	3	4	4	4
№ позиции	12	13	14	15	16	17	18	19	20	21	
x	4	4	4	4	4	5	5	5	5	5	
y	4	4	4	4	5	5	5	5	5	5	

Таблица Г.4 –Ряд распределения баллов в выборках

Оценка	"5"	"4"	"3"
X	0,24	0,33	0,43
Y	0,29	0,33	0,38

Таблица Г.5 – Проверки правил контроля студентами

№ студентов в группе	Количество правил созданных обучаемым	Количество правил с противоречивостью	% правил с противоречивостью	количество правил с ошибками достижимости	% правил с ошибками достижимости	Количество дополненных правил	% дополненных правил
1	6	0	0	0	0	1	16,7
2	15	3	20	2	13,3	2	13, 3
3	8	1	12,5	1	12,5	2	25
4	7	0	0	0	0	1	14,3
5	14	3	21,4	2	14,3	2	14,3
6	9	1	11,1	0	0	1	11,1
7	12	2	16,7	1	8,3	1	8,3
8	15	2	13,3	0	0	3	20

Продолжение таблицы Г.5

9	8	0	0	0	0	1	12,5
10	7	0	0	0	0	0	0
11	14	2	14,3	1	7,1	2	14,3
12	8	1	12,5	0	0	1	12,5
13	9	1	11,1	1	11,1	0	0
14	9	0	0	0	0	1	11,1
15	12	2	16,7	1	8,3	1	8,3
16	9	1	11,1	0	0	0	0
17	12	2	16,7	1	8,3	1	8,3
18	9	1	11,1	1	11,1	1	11,1
19	15	2	13,3	1	6,7	3	20
20	8	0	0	0	0	1	12,5
21	12	1	8,3	1	8,3	0	0
Среднее значение в %%			10		5,2		11

ПРИЛОЖЕНИЕ Д. Таблицы для экспериментов с ЭС при обучении ЛПР

Таблица Д.1 – Результаты работы диспетчера - март

№ дня	Всего решений	Правильных решений	Неправильных решений	Из них ошибок первого рода	Из них ошибок второго рода	Количество несчастных случаев
2 марта	28	24	4	0	4	0
3 марта	36	31	5	1	4	1
4 марта	32	27	5	0	5	0
5 марта	24	21	3	0	3	0
6 марта	40	34	6	1	5	2
10 марта	33	28	5	0	5	0
11 марта	35	30	5	0	5	1
12 марта	32	27	5	0	5	0
13 марта	28	25	3	0	3	0
16 марта	33	28	5	1	4	1
17 марта	32	27	5	0	5	1
18 марта	29	25	4	0	4	0
19 марта	31	26	5	0	5	0
20 марта	31	27	4	0	4	0
23 марта	36	30	6	0	6	1
24 марта	34	29	5	0	5	0
25 марта	33	28	5	0	5	0
26 марта	32	27	5	0	5	0
27 марта	28	24	4	0	4	0
30 марта	33	28	5	0	5	0
31 марта	35	30	5	1	4	1
Всего						

Таблица Д.2 – Результаты работы диспетчера – апрель

№ дня	Всего реше-ний	Правильных решений	Неправильных решений	Из них ошибок первого рода	Из них ошибок второго рода	Количество несчастных случаев
1 апреля	24	21	3	0	3	0
2 апреля	35	29	6	1	5	1
3 апреля	30	25	5	0	5	0
6 апреля	32	27	5	0	5	0
7 апреля	23	20	3	1	2	2
8 апреля	31	26	5	0	5	0
9 апреля	37	31	6	1	5	1
10 апреля	37	32	5	0	5	0
13 апреля	31	26	5	0	5	0
14 апреля	33	28	5	1	4	1
15 апреля	34	31	3	0	3	1
16 апреля	27	23	4	0	4	0
17 апреля	32	27	5	0	5	0
20 апреля	30	25	5	0	5	0
21 апреля	35	30	5	1	4	1

Продолжение таблицы Д.2

22 апреля	30	26	4	0	4	0
23 апреля	32	27	5	0	5	0
24 апреля	34	29	5	0	5	0
27 апреля	33	28	5	0	5	0
28 апреля	27	24	3	0	3	0
29 апреля	30	25	5	1	4	1
30 апреля	33	29	4	0	4	0

Таблица Д.3 – Результаты работы диспетчера – июнь

№ дня	Всего реше-ний	Правильных решений	Неправильных решений	Из них ошибок первого рода	Из них ошибок второго рода	Количество несчастных случаев
1 июня	26	24	2	0	2	0
2 июня	37	35	2	0	2	1
3 июня	28	27	1	0	1	0
4 июня	21	21	0	0	0	0
5 июня	31	31	0	0	0	0
8 июня	29	29	0	0	0	0
9 июня	32	30	2	1	1	1
10 июня	28	28	0	0	0	0
11 июня	25	25	0	0	0	0
12 июня	29	28	1	0	1	1
15 июня	27	27	0	0	0	0
16 июня	25	25	0	0	0	0
17 июня	28	26	2	0	2	0
18 июня	28	28	0	0	0	0
19 июня	33	31	2	0	2	1
22 июня	30	30	0	0	0	0
23 апреля	31	29	2	0	2	0
24 июня	31	28	3	0	3	1
25 июня	24	24	0	0	0	0
26 июня	29	29	0	0	0	0
29 июня	34	31	3	1	2	1
30 июня	23	22	1	0	1	0

Таблица Д.4 – Результаты работы диспетчера – июль

№ дня	Всего реше-ний	Правильных решений	Неправильных решений	Из них ошибок первого рода	Из них ошибок второго рода	Количество несчастных случаев
1 июля	32	30	2	0	2	1
2 июля	26	26	0	0	0	0
3 июля	31	29	2	0	2	1
6 июля	21	21	0	0	0	0

Продолжение Д.4

7 июля	29	27	2	0	2	0
8 июля	30	30	0	0	0	0
9 июля	33	31	2	0	2	1
10 июля	29	27	2	0	2	0
13 июля	32	30	2	0	2	0
14июля	32	32	0	0	0	0
15 июля	25	24	1	0	1	0
16 июля	31	29	2	0	2	1
17 июля	25	25	0	0	0	0
20 июля	34	31	3	1	2	1
21 июля	28	28	0	0	0	0
22 июля	29	29	0	0	0	0
23 июля	33	31	2	0	2	1
24 июля	32	30	2	0	2	0
27 июля	26	26	0	0	0	0
28 июля	27	27	0	0	0	0
29 июля	33	31	2	0	2	1
30 июля	29	29	0	0	0	0
31 июля	31	31	0	0	0	0

ПРИЛОЖЕНИЕ Ж. Листинг кода редактора правил

Клас MinimizedRuleSetForm

```
package com.achernenko.editor.ui;

import com.achernenko.editor.model.graph.Edge;
import com.achernenko.editor.model.graph.Vertex;
import com.achernenko.editor.ui.graph.RuleProcessor;
import com.intellij.uiDesigner.core.GridConstraints;
import com.intellij.uiDesigner.core.GridLayoutManager;
import com.intellij.uiDesigner.core.Spacer;
import edu.uci.ics.jung.algorithms.layout.Layout;
import edu.uci.ics.jung.visualization.GraphZoomScrollPane;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import org.jdesktop.swingx.JXLabel;

import javax.swing.*;
import java.awt.*;
import java.util.Map;

import static com.achernenko.editor.ui.Utils.convertBooleanFormula;

public class MinimizedRuleSetForm {
    private String booleanExpression;
    private Map<String, Vertex> termToVertexMap;
    private String conclusionVertexName;
    private JXLabel labelBooleanExpression;
    private GraphZoomScrollPane graphZoomScrollPane;
    private JPanel panelRoot;

    public MinimizedRuleSetForm(String booleanExpression, Map<String, Vertex> termToVertexMap, String
conclusionVertexName) {
        this.booleanExpression = booleanExpression;
        this.termToVertexMap = termToVertexMap;
        this.conclusionVertexName = conclusionVertexName;

        $$$setupUI$$$();
        labelBooleanExpression.setText(convertBooleanFormula(booleanExpression, termToVertexMap).replace("",
"" ").replace("+", " + ").replace("  ", " ").replace("*", " "));
    }

    private void createUIComponents() {
        labelBooleanExpression = new JXLabel();
        labelBooleanExpression.setLineWrap(true);

        Layout<Vertex, Edge> graphLayout = Utils.createGraph(booleanExpression, termToVertexMap,
conclusionVertexName);
        VisualizationViewer<Vertex, Edge> vv = Utils.createVisualizationViewer(graphLayout);
        graphZoomScrollPane = new GraphZoomScrollPane(vv);
        new RuleProcessor().processGraph(graphLayout.getGraph());
        vv.repaint(); }

    public void show() {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Мінімальний набір правил");
                frame.setContentPane(panelRoot);
                frame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                frame.pack();
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);    } });  }

    /**
     * @noinspection ALL
     */
    public JComponent $$$getRootComponent$$$() {
        return panelRoot;    }}
```

Клас RuleSetComparisonForm

```
package com.achernenko.editor.ui;

import com.achernenko.editor.model.graph.Edge;
import com.achernenko.editor.model.graph.PredicateGroup;
import com.achernenko.editor.model.graph.Vertex;
```

```
import com.achernenko.editor.ui.graph.RuleProcessor;
import com.achernenko.editor.ui.graph.EdgeFactory;
import com.achernenko.editor.ui.graph.EdgeLabelClosenessTransformer;
import com.achernenko.editor.ui.graph.EdgeLabeller;
import com.achernenko.editor.ui.graph.EdgeShapeTransformer;
import com.achernenko.editor.ui.graph.VertexFactory;
import com.achernenko.editor.ui.graph.VertexFillPaintTransformer;
import com.achernenko.editor.ui.graph.VertexLabeller;
import com.achernenko.editor.ui.graph.VertexShapeTransformer;
import com.achernenko.editor.ui.plugins.GraphMouse;
import com.intellij.uiDesigner.core.GridConstraints;
import com.intellij.uiDesigner.core.GridLayoutManager;
import com.intellij.uiDesigner.core.Spacer;
import edu.uci.ics.jung.algorithms.layout.Layout;
import edu.uci.ics.jung.algorithms.layout.StaticLayout;
import edu.uci.ics.jung.graph.DirectedSparseMultigraph;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.graph.util.EdgeType;
import edu.uci.ics.jung.visualization.GraphZoomScrollPane;
import edu.uci.ics.jung.visualization.RenderContext;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import edu.uci.ics.jung.visualization.control.ModalGraphMouse;
import edu.uci.ics.jung.visualization.renderers.Renderer;
import org.jdesktop.swingx.JXLabel;

import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.util.List;

import static com.achernenko.editor.model.graph.Vertex.DEFAULT_TYPE;
import static com.achernenko.editor.ui.Utils.convertBooleanFormula;

public class RuleSetComparisonForm {
    private JPanel panelRoot;
    private GraphZoomScrollPane expressionGraphScrollPane;
    private GraphZoomScrollPane negExpressionGraphScrollPane;
    private JSplitPane splitPane;
    private JPanel panelLeft;
    private JPanel panelRight;
    private JXLabel labelNegativeBooleanExpression;
    private JXLabel labelBooleanExpression;
    private final String booleanExpression;
    private final String negativeBooleanExpression;
    private Map<String, Vertex> termToVertexMap;
    private String conclusionVertexName;

    public RuleSetComparisonForm(String booleanExpression, String negativeBooleanExpression, Map<String, Ver-
tex> termToVertexMap, String conclusionVertexName) {
        this.booleanExpression = booleanExpression;
        this.negativeBooleanExpression = negativeBooleanExpression;
        this.termToVertexMap = termToVertexMap;
        this.conclusionVertexName = conclusionVertexName;

        $$$setupUI$$$();
        labelBooleanExpression.setText(convertBooleanFormula(booleanExpression, termToVertexMap).replace("",
"' ").replace("+", " + ").replace("  ", " ").replace("*", " "));
        labelNegativeBooleanExpression.setText(convertBooleanFormula(negativeBooleanExpression,
termToVertexMap).replace("", "' ").replace("+", " + ").replace("  ", " "));   }

    public void show() {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Перевірка повноти");
                frame.setContentPane(panelRoot);
                frame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                frame.pack();
                frame.setLocationRelativeTo(null);
                frame.setVisible(true); } }); }

    private void createUIComponents() {
        Layout<Vertex, Edge> graphLayout = Utils.createGraph(booleanExpression, termToVertexMap,
conclusionVertexName);
        VisualizationViewer<Vertex, Edge> vv = Utils.createVisualizationViewer(graphLayout);
        expressionGraphScrollPane = new GraphZoomScrollPane(vv);
        new RuleProcessor().processGraph(graphLayout.getGraph());
        vv.repaint();
```

```
        Layout<Vertex, Edge> negGraphLayout = Utils.createGraph(negativeBooleanExpression, termToVertexMap,
"HE " + conclusionVertexName);
        VisualizationViewer<Vertex, Edge> negVv = Utils.createVisualizationViewer(negGraphLayout);
        negExpressionGraphScrollPane = new GraphZoomScrollPane(negVv);
        new RuleProcessor().processGraph(negGraphLayout.getGraph());
        negVv.repaint();

        labelBooleanExpression = new JXLabel();
        labelBooleanExpression.setLineWrap(true);
        labelNegativeBooleanExpression = new JXLabel();
        labelNegativeBooleanExpression.setLineWrap(true); }

    /**
     * @noinspection ALL
     */
    public JComponent $$$getRootComponent$$$() {
        return panelRoot;    }}
```

Клас Utils

```
package com.achernenko.editor.ui;

import com.achernenko.editor.bool_operations.TruthTable;
import com.achernenko.editor.model.graph.Edge;
import com.achernenko.editor.model.graph.PredicateGroup;
import com.achernenko.editor.model.graph.Vertex;
import com.achernenko.editor.ui.graph.EdgeFactory;
import com.achernenko.editor.ui.graph.EdgeLabelClosenessTransformer;
import com.achernenko.editor.ui.graph.EdgeLabeller;
import com.achernenko.editor.ui.graph.EdgePaintTransformer;
import com.achernenko.editor.ui.graph.EdgeShapeTransformer;
import com.achernenko.editor.ui.graph.VertexFactory;
import com.achernenko.editor.ui.graph.VertexFillPaintTransformer;
import com.achernenko.editor.ui.graph.VertexLabeller;
import com.achernenko.editor.ui.graph.VertexShapeTransformer;
import com.achernenko.editor.ui.plugins.GraphMouse;
import edu.uci.ics.jung.algorithms.layout.Layout;
import edu.uci.ics.jung.algorithms.layout.StaticLayout;
import edu.uci.ics.jung.graph.DirectedSparseMultigraph;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.graph.util.EdgeType;
import edu.uci.ics.jung.visualization.RenderContext;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import edu.uci.ics.jung.visualization.control.ModalGraphMouse;
import edu.uci.ics.jung.visualization.renderers.Renderer;

import java.awt.*;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.util.List;
import java.util.*;

import static com.achernenko.editor.model.graph.Vertex.DEFAULT_TYPE;

public class Utils {
    private static final int SPACING_Y = 120;

    @SuppressWarnings(value = "unchecked")
    public static void clearGraph(Graph graph) {
        ArrayList vertices = new ArrayList(graph.getVertices());
        for (Object vertex : vertices) {
            graph.removeVertex(vertex);  }  }

    public static String readRawData(File file) throws IOException {
        StringBuilder accumulator = new StringBuilder();
        Scanner scanner = new Scanner(file, "UTF-8");
        while (scanner.hasNextLine()) {
            accumulator.append(scanner.nextLine()); }

        scanner.close();
        return accumulator.toString();  }

    public static void writeRawData(String rawData, File file) throws IOException {
        PrintStream printStream = new PrintStream(new FileOutputStream(file), true, "UTF-8");
        printStream.println(rawData);
```

```
        printStream.close();  }

    public static String convertBooleanFormula(String booleanFormula, Map<String, Vertex> varToVertexMap) {
        StringBuilder output = new StringBuilder();

        for (int i = 0; i < booleanFormula.length(); i++) {
            Character token = booleanFormula.charAt(i);
            Character nextToken = i + 1 < booleanFormula.length() ? booleanFormula.charAt(i + 1) : null;
            if (Character.isWhitespace(token)) {
                continue;    }

            if (Character.isLetter(token)) {
                Vertex vertex = varToVertexMap.get(String.valueOf(token));
                output.append(vertex.toString());

                if (nextToken!=null && Character.isLetter(nextToken)) {
                    output.append('*');
                }
            } else {
                output.append(token);  }}
        return output.toString().replace("\n", " ");   }

    public static Layout<Vertex, Edge> createGraph(String booleanExpression, Map<String, Vertex>
termToVertexMap, String conclusionVertexName) {
        Graph<Vertex, Edge> graph = new DirectedSparseMultigraph<>();

        String[] minterms = booleanExpression.split("\\+");
        Set<Vertex> firstLevelVertices = new HashSet<>();
        Set<Vertex> secondLevelVertices = new HashSet<>();

        Vertex orVertex = null;
        if (minterms.length > 1) {
            orVertex = new Vertex(Vertex.Type.OR, conclusionVertexName);
            graph.addVertex(orVertex);
        }
        boolean hasValidMinterms = false;

        for (String minterm : minterms) {
            minterm = minterm.trim();
            java.util.List<Vertex> mintermVertices = new ArrayList<>();

            for (int i = 0; i < minterm.length(); i++) {
                String vertexName = String.valueOf(minterm.charAt(i));
                Vertex vertexTitle = termToVertexMap.get(vertexName);
                if (i < minterm.length() - 1 && minterm.charAt(i + 1) == '\'') {
                    vertexName = vertexName + '\'';
                    i++;

                    if (!termToVertexMap.containsKey(vertexName)) {
                        termToVertexMap.put(vertexName, new Vertex(Vertex.Type.AND, "HE\n" + vertexTitle)); } }
                Vertex vertex = termToVertexMap.get(vertexName);
                graph.addVertex(vertex);
                mintermVertices.add(vertex);
                firstLevelVertices.add(vertex);  }

            if (mintermVertices.size() == 1) {
                if (orVertex != null) {
                    graph.addEdge(new Edge(), mintermVertices.get(0), orVertex, EdgeType.DIRECTED);
                }
            } else {
                Vertex andVertex = new Vertex(Vertex.Type.AND, "");
                graph.addVertex(andVertex);
                secondLevelVertices.add(andVertex);
                if (orVertex != null) {
                    graph.addEdge(new Edge(), andVertex, orVertex, EdgeType.DIRECTED); }

                for (Vertex vertex : mintermVertices) {
                    graph.addEdge(new Edge(), vertex, andVertex, EdgeType.DIRECTED);  }  }

            hasValidMinterms = true;     }

        int graphWidth = Math.max(firstLevelVertices.size(), secondLevelVertices.size()) * 120;
        Layout<Vertex, Edge> graphLayout = new StaticLayout<>(graph, new Dimension(2500, 2500));
        if (!firstLevelVertices.isEmpty()) {
            java.util.List<Vertex> firstLevelVerticesList = new ArrayList<>(firstLevelVertices);
            Collections.sort(firstLevelVerticesList, new Comparator<Vertex>() {
                @Override
                public int compare(Vertex o1, Vertex o2) {
```

```
                PredicateGroup predicateGroup1 = o1.getPredicateGroup();
                PredicateGroup predicateGroup2 = o2.getPredicateGroup();
                int id1 = predicateGroup1 == null ? 0 : predicateGroup1.getId();
                int id2 = predicateGroup2 == null ? 0 : predicateGroup2.getId();

                return id1 - id2;
            }
        });
        int firstLevelXSpacing = graphWidth / firstLevelVertices.size();
        int x = firstLevelXSpacing / 2;
        for (Vertex vertex : firstLevelVerticesList) {
            graphLayout.setLocation(vertex, new Point(x, SPACING_Y));
            x += firstLevelXSpacing; } }

    if (!secondLevelVertices.isEmpty()) {
        int secondLevelXSpacing = graphWidth / secondLevelVertices.size();
        int x = secondLevelXSpacing / 2;
        for (Vertex vertex : secondLevelVertices) {
            graphLayout.setLocation(vertex, new Point(x, SPACING_Y * 2));
            x += secondLevelXSpacing;  } }

    if (hasValidMinterms) {
        if (orVertex != null) {
            graphLayout.setLocation(orVertex, new Point(graphWidth / 2, SPACING_Y * 3));
        }
    } else {
        graph.removeVertex(orVertex);   }

    return graphLayout;   }

public static boolean isSatisfiable(TruthTable truthTable) {
    boolean isSatisfiable = false;

    boolean[] truthValues = truthTable.getTruthValues();
    if (truthValues != null) {
        for (Boolean value : truthValues) {
            isSatisfiable |= value;  } }

    return isSatisfiable;  }

public static VisualizationViewer<Vertex, Edge> createVisualizationViewer(Layout<Vertex, Edge>
graphLayout) {
    VisualizationViewer<Vertex, Edge> vv = new VisualizationViewer<>(graphLayout);
    vv.setBackground(Color.WHITE);

    setupRenderContext(vv, vv.getRenderer(), vv.getRenderContext());

    GraphMouse<Vertex, Edge> graphMouse = new GraphMouse<>(new VertexFactory(DEFAULT_TYPE), new
EdgeFactory());
    vv.setGraphMouse(graphMouse);
    vv.addKeyListener(graphMouse.getModeKeyListener());
    graphMouse.setMode(ModalGraphMouse.Mode.PICKING);

    return vv;  }

public static void setupRenderContext(VisualizationViewer<Vertex, Edge> vv, Renderer<Vertex, Edge> render-
er, RenderContext<Vertex, Edge> renderContext) {
    renderer.getVertexLabelRenderer().setPosition(Renderer.VertexLabel.Position.CNTR);

    renderContext.setVertexLabelTransformer(new VertexLabeller(vv));
    EdgePaintTransformer edgePaintTransformer = new EdgePaintTransformer(vv);
    renderContext.setEdgeLabelTransformer(new EdgeLabeller());
    renderContext.getEdgeLabelRenderer().setRotateEdgeLabels(false);
    renderContext.setLabelOffset(RenderContext.LABEL_OFFSET);
    renderContext.setEdgeLabelClosenessTransformer(new EdgeLabelClosenessTransformer());
    renderContext.setEdgeShapeTransformer(new EdgeShapeTransformer());
    renderContext.setEdgeDrawPaintTransformer(edgePaintTransformer);
    renderContext.setArrowDrawPaintTransformer(edgePaintTransformer);
    renderContext.setArrowFillPaintTransformer(edgePaintTransformer);
    renderContext.setVertexShapeTransformer(new VertexShapeTransformer(vv));
    renderContext.setVertexFillPaintTransformer(new VertexFillPaintTransformer(vv)); }

public static ArrayList<Vertex> getConclusionVertices(Graph<Vertex, Edge> graph) {
    ArrayList<Vertex> conclusionVertices = new ArrayList<>();

    for (Vertex vertex : graph.getVertices()) {
        if (graph.getOutEdges(vertex).isEmpty()) {
            conclusionVertices.add(vertex);   }  }
```

```
        return conclusionVertices;  }

    public static List<List<Vertex>> findCycles(Graph<Vertex, Edge> currentGraph) {
        List<List<Vertex>> suspects = new ArrayList<>();
        for (Vertex conclusionVertex : getConclusionVertices(currentGraph)) {
            depthFirstSearch(currentGraph, conclusionVertex, new ArrayList<Vertex>(), new HashSet<Edge>(), sus-
pects);   }

        return suspects; }

    private static void depthFirstSearch(Graph<Vertex, Edge> graph, Vertex vertex, List<Vertex> path,
HashSet<Edge> exploredEdges, List<List<Vertex>> cycles) {
        if (path.contains(vertex)) {
            path.add(vertex);
            for (int i = 0; i < path.size();) {
                if (!path.get(i).equals(vertex)) {
                    path.remove(i);
                } else {
                    break; }}
            Collections.reverse(path);
            cycles.add(path);   }
 else {
            List<Vertex> newPath = new ArrayList<>(path);
            newPath.add(vertex);
            for (Vertex predecessor : graph.getPredecessors(vertex)) {
                boolean isExplored = true;
                Collection<Edge> inEdges = graph.findEdgeSet(predecessor, vertex);
                for (Edge edge : inEdges) {
                    isExplored &= exploredEdges.contains(edge);  }

                if (!isExplored) {
                    exploredEdges.addAll(inEdges);
                    depthFirstSearch(graph, predecessor, newPath, exploredEdges, cycles);    } }  }  } }
```

Клас GraphMouse

```
package com.achernenko.editor.ui.plugins;

import com.achernenko.editor.ui.plugins.editor.EditorListener;
import com.achernenko.editor.ui.plugins.editor.EditorPlugin;
import com.achernenko.editor.ui.plugins.picker.PickedVertexStateListener;
import com.achernenko.editor.ui.plugins.picker.PickerPlugin;
import edu.uci.ics.jung.visualization.control.AbstractModalGraphMouse;
import edu.uci.ics.jung.visualization.control.CrossoverScalingControl;
import edu.uci.ics.jung.visualization.control.TranslatingGraphMousePlugin;
import org.apache.commons.collections15.Factory;

import java.awt.event.ItemEvent;

public class GraphMouse<V, E> extends AbstractModalGraphMouse {
    private final Factory<V> vertexFactory;
    private final Factory<E> edgeFactory;
    private EditorPlugin<V, E> editorPlugin;

    public GraphMouse(Factory<V> vertexFactory, Factory<E> edgeFactory) {
        super(1.1f, 1 / 1.1f);
        this.vertexFactory = vertexFactory;
        this.edgeFactory = edgeFactory;
        loadPlugins();  }

    @Override
    protected void loadPlugins() {
        editorPlugin = new EditorPlugin<>(vertexFactory, edgeFactory);
        pickingPlugin = new PickerPlugin<V, E>();
        scalingPlugin = new ZoomPlugin(new CrossoverScalingControl(), in, out);
        translatingPlugin = new TranslatingGraphMousePlugin();

        add(pickingPlugin);
        add(scalingPlugin);
        add(editorPlugin);
        add(translatingPlugin);  }

    @Override
    public void setMode(Mode mode) {
        if (this.mode != mode) {
            fireItemStateChanged(new ItemEvent(this, ItemEvent.ITEM_STATE_CHANGED, this.mode,
```

```
ItemEvent.DESELECTED));

        this.mode = mode;
        switch (mode) {
            case PICKING:
                remove(editorPlugin);
                break;
            case EDITING:
                add(editorPlugin);
                break;    }

        fireItemStateChanged(new ItemEvent(this, ItemEvent.ITEM_STATE_CHANGED, mode,
ItemEvent.SELECTED));  } }

    @SuppressWarnings("unchecked")
    public void setPickedStateListener(PickedVertexStateListener<V> pickedVertexStateListener) {
        PickerPlugin<V, E> pickerPlugin = (PickerPlugin<V, E>) pickingPlugin;
        if (pickerPlugin != null)
            pickerPlugin.setPickedVertexStateListener(pickedVertexStateListener);  }

    public void setEditorListener(EditorListener<V, E> editorListener) {
        if (editorPlugin != null)
            editorPlugin.setEditorListener(editorListener);    }}
```

Клас Vertex

```
package com.achernenko.editor.model.graph;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.Collection;
import java.util.HashSet;
import java.util.Set;

public class Vertex implements Comparable<Vertex> {
    public static Type DEFAULT_TYPE = Type.AND;
    public static IdGenerator ID_GENERATOR = new IdGenerator();

    private int id;
    private String title;
    private Type type;
    private PredicateGroup predicateGroup;
    private Set<RuleGroup> ruleGroups = new HashSet<>();
    {
        ruleGroups.add(RuleGroup.ALL);  }

    public Vertex(Type newPredicateType) {
        type = newPredicateType;
        id = ID_GENERATOR.nextId();
        title = String.format("p%d", id);  }

    public Vertex(Type newPredicateType, String title) {
        type = newPredicateType;
        id = ID_GENERATOR.nextId();
        this.title = title;  }

    public Vertex(JSONObject jsonObject) throws JSONException {
        type = Type.valueOf(jsonObject.getString("type"));
        title = jsonObject.getString("title");
        id = jsonObject.getInt("id");

        ID_GENERATOR.reportIdUsed(id); }

    public static boolean areVerticesEqual(Vertex vertex1, Vertex vertex2) {
        return vertex1 != null && vertex2 != null && vertex1.getId() == vertex2.getId();  }

    @Override
    public String toString() {
        return title;   }

    public JSONObject toJsonObject() throws JSONException {
        JSONObject jsonObject = new JSONObject();

        jsonObject.put("id", id);
        jsonObject.put("title", title);
```

```
        jsonObject.put("type", type.toString());
        if (predicateGroup != null) {
            jsonObject.put("predicateGroupId", predicateGroup.getId());    }

        JSONArray groupsJsonArray = new JSONArray();
        for (RuleGroup ruleGroup : getRuleGroups()) {
            if (ruleGroup != null) {
                groupsJsonArray.put(ruleGroup.getId()); } }

        return jsonObject;  }

    public String getTitle() {
        return title;  }

    public Type getType() {
        return type;  }

    public void setTitle(String title) {
        this.title = title;  }

    public void setType(Type type) {
        this.type = type;  }

    public int getId() {
        return id;  }

    public void setId(int id) {
        this.id = id; }

    public int getHeigth() {
        int lineCount = getTitle().split("\\n").length + 1;

        return 10 + lineCount * 18;  }

    public int getWidth() {
        String[] lines = getTitle().split("\\n");
        int longestLineLength = 0;
        for (String line : lines) {
            longestLineLength = Math.max(longestLineLength, line.length());  }

        return Math.max(10 + longestLineLength * 8, 80);  }

    public static void restartIdGenerator() {
        ID_GENERATOR.restart();  }

    public Collection<RuleGroup> getRuleGroups() {
        return ruleGroups;    }

    public void addRuleGroup(RuleGroup ruleGroup) {
        ruleGroups.add(ruleGroup);  }

    public void removeRuleGroup(RuleGroup ruleGroup) {
        ruleGroups.remove(ruleGroup);  }

    @Override
    public int compareTo(Vertex o) {
        return this.id - o.id; }

    public PredicateGroup getPredicateGroup() {
        return predicateGroup;  }

    public void setPredicateGroup(PredicateGroup predicateGroup) {
        this.predicateGroup = predicateGroup;  }

    public enum Type {
        AND("І"),
        OR("АБО");

        public String getDisplayedName() {
            return displayedName;  }

        private String displayedName;

        Type(String displayedName) {
            this.displayedName = displayedName;    } }}
```

Клас GraphSerializer

```
package com.achernenko.editor.model.graph;

import edu.uci.ics.jung.algorithms.layout.AbstractLayout;
import edu.uci.ics.jung.algorithms.layout.Layout;
import edu.uci.ics.jung.algorithms.layout.StaticLayout;
import edu.uci.ics.jung.graph.DirectedSparseGraph;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.graph.util.EdgeType;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.awt.*;
import java.awt.geom.Point2D;
import java.util.*;
import java.util.List;

public class GraphSerializer {
    public void deserialize(String rawData, Map<RuleGroup, Layout<Vertex, Edge>> layoutsMap,
List<PredicateGroup> predicateGroups, Dimension dimension) throws JSONException {
        validateData(rawData);

        Map<Integer, RuleGroup> ruleGroups = new HashMap<>();
        Map<Integer, PredicateGroup> predicateGroupsMap = new HashMap<>();
        ruleGroups.put(RuleGroup.ALL.getId(), RuleGroup.ALL);
        layoutsMap.clear();
        layoutsMap.put(RuleGroup.ALL, new StaticLayout<>(new DirectedSparseGraph<Vertex, Edge>(), dimen-
sion));
        predicateGroups.clear();

        Map<Integer, Vertex> vertices = new HashMap<>();
        Vertex.restartIdGenerator();
        RuleGroup.restartIdGenerator();
        PredicateGroup.restartIdGenerator();

        JSONObject rootObject = new JSONObject(rawData);

        JSONArray ruleGroupsJsonArray = rootObject.optJSONArray("groups");
        if (ruleGroupsJsonArray != null) {
            for (int i = 0; i < ruleGroupsJsonArray.length(); i++) {
                JSONObject ruleGroupJsonObject = ruleGroupsJsonArray.getJSONObject(i);
                RuleGroup ruleGroup = new RuleGroup(ruleGroupJsonObject.getInt("id"),
ruleGroupJsonObject.getString("title"));
                if (ruleGroup.getId() != -1) {
                    ruleGroups.put(ruleGroup.getId(), ruleGroup);
                    layoutsMap.put(ruleGroup, new StaticLayout<>(new DirectedSparseGraph<Vertex, Edge>(), dimen-
sion));  }  }  }
        JSONArray predicateGroupsJsonArray = rootObject.optJSONArray("predicateGroups");
        if (predicateGroupsJsonArray != null) {
            for (int i = 0; i < predicateGroupsJsonArray.length(); i++) {
                JSONObject groupJsonObject = predicateGroupsJsonArray.getJSONObject(i);
                PredicateGroup predicateGroup = new PredicateGroup(groupJsonObject.getInt("id"),
groupJsonObject.getString("title"), groupJsonObject.getInt("color"));
                predicateGroupsMap.put(predicateGroup.getId(), predicateGroup);
                predicateGroups.add(predicateGroup); } }
        JSONArray verticesArray = rootObject.getJSONArray("vertices");
        for (int i = 0; i < verticesArray.length(); i++) {
            JSONObject vertexJsonObject = verticesArray.getJSONObject(i);
            Vertex vertex = new Vertex(vertexJsonObject);

            vertices.put(vertex.getId(), vertex);

            //predicate group
            if (vertexJsonObject.has("predicateGroupId")) {
                PredicateGroup predicateGroup = predicateGroupsMap.get(vertexJsonObject.getInt("predicateGroupId"));
                vertex.setPredicateGroup(predicateGroup);
                predicateGroup.addVertex(vertex);
            }
            //reading location
            JSONArray locationsJsonArray = vertexJsonObject.optJSONArray("locations");
            if (locationsJsonArray != null) {
                for (int j = 0; j < locationsJsonArray.length(); j++) {
                    JSONObject locationJsonObject = locationsJsonArray.getJSONObject(j);
                    Point2D location = new Point(locationJsonObject.getInt("x"), locationJsonObject.getInt("y"));
                    RuleGroup ruleGroup = ruleGroups.get(locationJsonObject.getInt("groupId"));
                    vertex.addRuleGroup(ruleGroup);
                    Layout<Vertex, Edge> groupLayout = layoutsMap.get(ruleGroup);
                    groupLayout.getGraph().addVertex(vertex);
```

```
                groupLayout.setLocation(vertex, location);
            }
        } else {
            Point2D location = new Point(vertexJsonObject.getInt("x"), vertexJsonObject.getInt("y"));
            layoutsMap.get(RuleGroup.ALL).setLocation(vertex, location);
        }
    }
    JSONArray edgesArray = rootObject.getJSONArray("edges");
    for (int i = 0; i < edgesArray.length(); i++) {
        JSONObject edgeObject = edgesArray.getJSONObject(i);
        Edge edge = new Edge();
        Vertex src = vertices.get(edgeObject.getInt("src"));
        Vertex dst = vertices.get(edgeObject.getInt("dst"));
        Set<RuleGroup> allRuleGroups = new HashSet<>();
        Collection<RuleGroup> srcRuleGroups = src.getRuleGroups();
        Collection<RuleGroup> dstRuleGroups = dst.getRuleGroups();
        allRuleGroups.addAll(srcRuleGroups);
        allRuleGroups.addAll(dstRuleGroups);
        for (RuleGroup ruleGroup : allRuleGroups) {
            if (srcRuleGroups.contains(ruleGroup) && dstRuleGroups.contains(ruleGroup)) {
                layoutsMap.get(ruleGroup).getGraph().addEdge(edge, src, dst, EdgeType.DIRECTED); } } } }

  public String serialize(Map<RuleGroup, Layout<Vertex, Edge>> layoutsMap, List<PredicateGroup>
predicateGroups) throws JSONException {
    JSONObject rootObject = new JSONObject();
    //saving rule groups
    JSONArray groupsJsonArray = new JSONArray();
    for (RuleGroup ruleGroup : layoutsMap.keySet()) {
        JSONObject groupJsonObject = new JSONObject();
        groupJsonObject.put("id", ruleGroup.getId());
        groupJsonObject.put("title", ruleGroup.getTitle());
        groupsJsonArray.put(groupJsonObject);
    }
    rootObject.put("groups", groupsJsonArray);
    //saving predicate groups
    JSONArray predicateGroupsJsonArray = new JSONArray();
    for (PredicateGroup predicateGroup : predicateGroups) {
        JSONObject groupJsonObject = new JSONObject();
        groupJsonObject.put("id", predicateGroup.getId());
        groupJsonObject.put("title", predicateGroup.getTitle());
        groupJsonObject.put("color", predicateGroup.getVertexColor().getRGB());
        predicateGroupsJsonArray.put(groupJsonObject);
    }
    rootObject.put("predicateGroups", predicateGroupsJsonArray);
    //saving vertices
    ///obtaining vertex coordinates
    JSONArray verticesArray = new JSONArray();
    Graph<Vertex, Edge> universalGraph = layoutsMap.get(RuleGroup.ALL).getGraph();
    for (Vertex vertex : universalGraph.getVertices()) {
        try {
            JSONObject vertexObject = vertex.toJsonObject();

            JSONArray locationsJsonArray = new JSONArray();
            for (RuleGroup ruleGroup : vertex.getRuleGroups()) {
                if (ruleGroup != null) {
                    AbstractLayout<Vertex, Edge> graphLayout = (AbstractLayout<Vertex, Edge>)
layoutsMap.get(ruleGroup);
                    JSONObject locationObject = new JSONObject();
                    locationObject.put("groupId", ruleGroup.getId());
                    locationObject.put("x", graphLayout.getX(vertex));
                    locationObject.put("y", graphLayout.getY(vertex));

                    locationsJsonArray.put(locationObject);  } }
            vertexObject.put("locations", locationsJsonArray);
            verticesArray.put(vertexObject);
        } catch (JSONException ex) {
            ex.printStackTrace();   } }
    rootObject.put("vertices", verticesArray);
    //saving edges
    JSONArray edgesArray = new JSONArray();
    for (Edge edge : universalGraph.getEdges()) {
        JSONObject edgeObject = new JSONObject();
        edgeObject.put("src", universalGraph.getSource(edge).getId());
        edgeObject.put("dst", universalGraph.getDest(edge).getId());
        edgesArray.put(edgeObject);
    }
    rootObject.put("edges", edgesArray);
    return rootObject.toString(4);
```

```
    }
    private void validateData(String rawData) throws JSONException {
        JSONObject object = new JSONObject(rawData);
        object.getJSONArray("edges");
        object.getJSONArray("vertices");   }}
```

Клас Rule

```
package com.achernenko.editor.model.graph;
import java.util.Collections;
import java.util.List;

public class Rule {
    private int id;
    private Vertex conclusion;
    private List<Vertex> predicates;
    public Rule(int ruleId, Vertex conclusion, List<Vertex> predicates) {
        id = ruleId;
        this.conclusion = conclusion;
        this.predicates = predicates;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public Vertex getConclusion() {
        return conclusion;
    }
    public void setConclusion(Vertex conclusion) {
        this.conclusion = conclusion;
    }
    public List<Vertex> getPredicates() {
        return predicates;
    }
    public void clearPredicates(List<Vertex> predicates) {
        this.predicates.clear();
   }
    public void addPredicate(Vertex predicate) {
        this.predicates.add(predicate);
    }
    public boolean isConsistent(StringBuilder complaints) {
        if (predicates == null || predicates.isEmpty()) {
            complaints.append(String.format("Rule %s has no predicates\n", id));
            return false;
        } else if (conclusion == null) {
            complaints.append(String.format("Rule %s has no conclusion\n", id));
            return false;
        }
        PredicateGroup conclusionGroup = conclusion.getPredicateGroup();
        for (int i = 0; i < predicates.size(); i++) {
            Vertex vertex = predicates.get(i);
            PredicateGroup predicateGroup = vertex.getPredicateGroup();
            if (PredicateGroup.areGroupsEqual(predicateGroup, conclusionGroup)) {
                complaints.append(String.format("Rule %s: predicate %s and conclusion belong to the same group\n", id,
vertex));
                return false;
            }
            for (int j = i + 1; j < predicates.size(); j++) {
                Vertex other = predicates.get(j);
                PredicateGroup otherPredicateGroup = other.getPredicateGroup();
                if (PredicateGroup.areGroupsEqual(predicateGroup, otherPredicateGroup)) {
                    complaints.append(String.format("Rule %s: predicates %s and %s belong to the same group\n", id, ver-
tex, other));
                    return false;  }  }  }
        return true;
    }
    public boolean equalTo(Rule other) {
        if (this.conclusion.getType() != other.conclusion.getType()) {
            return false;
        }
        if (this.predicates.size() != other.predicates.size()) {
            return false;
        }
        Collections.sort(this.predicates);
        Collections.sort(other.predicates);
```

```
        for (int i = 0; i < this.predicates.size(); i++) {
            if (!Vertex.areVerticesEqual(this.predicates.get(i), other.predicates.get(i))) {
                return false;   } }
        return true;
    }
    public boolean hasSamePredicatesWithDifferentConclusion(Rule other) {
        if (this.conclusion == null || other.conclusion == null || this.conclusion.getType() != oth-
er.conclusion.getType()) {
            return false;
        }

        if (this.predicates.size() != other.predicates.size()) {
            return false;
        } else {
            Collections.sort(this.predicates);
            Collections.sort(other.predicates);

            for (int i = 0; i < this.predicates.size(); i++) {
                if (!Vertex.areVerticesEqual(this.predicates.get(i), other.predicates.get(i))) {
                    return false;               }           }
            return true;       }    }}
```

Клас RuleProcessor

```
package com.achernenko.editor.ui.graph;

import com.achernenko.editor.model.graph.Edge;
import com.achernenko.editor.model.graph.IdGenerator;
import com.achernenko.editor.model.graph.Rule;
import com.achernenko.editor.model.graph.Vertex;
import com.achernenko.editor.ui.Utils;
import edu.uci.ics.jung.graph.Graph;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class RuleProcessor {
    public List<Rule> processGraph(Graph<Vertex, Edge> graph) {
        IdGenerator ruleIdGenerator = new IdGenerator();
        Set<Edge> annotatedEdges = new HashSet<>();
        List<Rule> rules = new ArrayList<>();

        ArrayList<Vertex> predecessorVertices = Utils.getConclusionVertices(graph);
        for (Vertex root : predecessorVertices) {
            processEdges(graph, root, rules, ruleIdGenerator, annotatedEdges);
        }
        return rules;    }
    private void processEdges(Graph<Vertex, Edge> graph, Vertex vertex, List<Rule> rules, IdGenerator
idGenerator, Set<Edge> annotatedEdges) {
        int incomingEdgesCount = graph.getPredecessorCount(vertex);

        if (incomingEdgesCount > 0) {
            int ruleId = -1;

            for (Edge edge : graph.getInEdges(vertex)) {
                if (annotatedEdges.contains(edge)) {
                    return;
                }
                if (ruleId == -1) {
                    ruleId = idGenerator.nextId();
                }
                annotatedEdges.add(edge);
                edge.setRuleId(ruleId);
            }
            List<Vertex> predecessors = new ArrayList<>(graph.getPredecessors(vertex));
            rules.add(new Rule(ruleId, vertex, predecessors));
            Collections.sort(predecessors);
            for (Vertex predecessor : predecessors) {
                processEdges(graph, predecessor, rules, idGenerator, annotatedEdges);  } } } }
```

ПРИЛОЖЕНИЕ К. Акты внедрения

К1. Акт внедрения результатов диссертационной работы на Коммунальном предприятии «Одесскоммунтранс»

Україна
Одеська міська рада
Управління житлово-комунального господарства
КОМУНАЛЬНЕ ПІДПРИЄМСТВО
“ОДЕСКОМУНТРАНС”
31185678
65031, м. Одеса, вул. Бр. Поджио 4
тел.7280612, 7280636
e-mail komuntrans@te.net.ua

Украина
Одесский городской совет
Управление жилищно-коммунального хозяйства
КОММУНАЛЬНОЕ ПРЕДПРИЯТИЕ
“ОДЕСКОММУНТРАНС”
31185678
65031, г. Одесса, ул. Бр. Поджио 4
тел. 7280612, 7280636
e-mail komuntrans@te.net.ua

№ 80 від «19» травня 2014 року

ЗАТВЕРДЖУЮ
Директор КП «ОДЕСКОМУНТРАНС»
Радченко С.М._____
«19» 05 2014 р.

Довідка
про впровадження результатів дисертаційної роботи
Тройніної Анастасії Сергіївни

Довідка видана в тому, що в Комунальному підприємстві «Одескомунтранс» прийнятий до використання програмно-апаратний комплекс експертної системи для моніторингу виконання вимог щодо безпечної експлуатації електроустановок та прийняття рішень про допустимість проведення робіт.

Експертна система застосовується при виконанні контролю за безпекою проведення робіт з електроустановками, у тому числі з можливістю застосування даних від апаратних засобів контролю для прийняття рішення про можливість проведення робіт і необхідність забезпечення додаткових умов, відповідність місця та часу проведення робіт вимогам техніки безпеки та охорони праці. Застосування програмного комплексу із системою прийняття рішень підвищує безпеку на підприємстві шляхом надання рекомендацій диспетчеру або іншій відповідальній особі.

Впроваджена інформаційна технологія на основі експертної системи для прийняття рішень у складі автоматизованого робочого місця диспетчера організації з безпечної експлуатації електричних установок дозволяє суттєво знизити ризик прийняття хибних або помилкових рішень, запобігти виробничому травматизму та скоротити час на обробку документації при плануванні робіт на 10%.

Довідка надана для подання в спеціалізовану Вчену Раду по захисту дисертацій.

Директор _____ Радченко С.М.

Головний енергетик _____ Рететий С.В

К2. Акт внедрения результатов диссертационной работы в ООО «Свитеко»

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
«СВІТЕКО»
65011, м. Одеса, вул. Єврейська, 23-А
Код ЕДРПОУ 39282608 тел. 048 7835345

ЗАТВЕРДЖУЮ
Директор ТОВ «СВІТЕКО»
Шляк О.А.
.
«11» лютого 2015 р.

Довідка
про впровадження результатів дисертаційної роботи
Тройніної Анастасії Сергіївни

Довідка видана в тому, що у ТОВ «СВІТЕКО» прийнятий до використання програмний комплекс з експертною системою для моніторингу роботи комп'ютерної мережі з можливістю інтелектуального аналізу ситуацій, що виникають у мережах.

Дана розробка може бути застосована в якості доповнення до існуючих систем моніторингу мереж для видачі рекомендацій адміністраторам. При цьому потенційно можлива інтеграція в будь-яку з таких систем у якості їх модуля. Застосування програмного комплексу із системою прийняття рішень значно знижує трудомісткість роботи системного адміністратора.

Впроваджена інформаційна технологія на основі експертної системи для прийняття рішень у складі автоматизованого робочого місця адміністратора мережі дозволяє суттєво знизити ризик прийняття хибних або помилкових рішень та скоротити час для прийняття рішень при адмініструванні комп'ютерної мережі на 5-7%.

Довідка надана для подання в спеціалізовану Вчену Раду по захисту дисертацій.

Директор _____ О.А. Шляк

К3. Акт внедрения результатов диссертационной работы в учебный процесс ОНПУ

ЗАТВЕРДЖУЮ

Проректор
з науково-педагогічної та
виховної роботи
_____д.т.н., проф. С.А.Нестеренко
«_4_» ____03____ 2015 р.

АКТ

впровадження результатів дисертаційної роботи

Тройніної Анастасії Сергіївни

«Моделі, методи та інформаційна технологія створення та супроводу знанняорієнтованих систем контролю»

у навчальному процесі ОНПУ

Цей акт складений про те, що в курсах по дисциплінам «Організація баз даних та знань», «Інтелектуальні системи» та «Інтелектуальний аналіз даних», що читаються студентам фаху 6.050103 "Програмна інженерія" на кафедрі «Системне програмне забезпечення» у 5, 8 і 9 семестрах відповідно, використовуються наукові результати, отримані в дисертаційній роботі Тройніної А. С.

Запропонована інформаційна технологія побудови експертних систем контролю станів об'єктів використовує інструментальні засоби у вигляді редактору правил для створення, перевірки та підтримки правил контролю. Створений редактор правил застосовується в лабораторних та курсових роботах та при дипломному проектуванні.

Застосування отриманих результатів в курсовій роботі по дисципліні «Інтелектуальний аналіз даних» дозволяє оперативно створювати та вносити зміни в правила контролю, підвищити достовірність контролю в середньому на 12%.

Директор
Інституту комп'ютерних
систем _____д.т.н., проф. С. Г. Антощук

Начальник навчально-методичного
відділу _____д.т.н., доц. О.С. Савельєва

Старший викладач
та аспірантка каф. СПО _____А.С. Тройніна