МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южный федеральный университет»
Инженерно- технологическая академия

Н.Е. Сергеев

Системы искусственного интеллекта

Часть 1

Учебное пособие

Таганрог
Издательство Южного федерального университета
2016

УДК 004.8(075.8) *Печатается по решению*

ББК 32.813я73 *редакционно-издательского совета*

С 322 *Южного федерального университета*

Рецензенты:

доктор технических наук, профессор, зав. кафедрой информатики
Таганрогского института имени А.П. Чехова
Ростовского государственного университета (РИНХ)
Ромм Я.Е.;
доктор технических наук, профессор, заведующий кафедрой САПР
ИКТИБ ИТА ЮФУ
Курейчик В.В.

Сергеев, Н.Е.

С 322 Системы искусственного интеллекта: учебное пособие Часть1 /Сергеев Н.Е; Южный федеральный университет. – Таганрог: Издательство Южного федерального университета, 2016. – 118 с.

ISBN 978-5-9275-2113-5

Приведены теоретические положения, необходимые для решения некоторых практических задач для систем, основанных на знаниях. Приведены также задания, методические рекомендации и примеры выполнения лабораторных работ.

Пособие предназначено для подготовки бакалавров направлений 09.03.01 «Информатика и вычислительная техника» и инженеров специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения» по специализациям «Применение и эксплуатация автоматизированных систем управления специальными радиотехническими средствами», «Математическое и программное обеспечение вычислительной техники и автоматизированных систем», «Автоматизированные системы обработки информации и управления» всех форм обучения по курсам «Системы искусственного интеллекта и нейрокомпьютеры», «Системы искусственного интеллекта», «Представление и использование знаний в информационных системах»

УДК 004.8(075.8)

ББК 32.813я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Термин «искусственный интеллект» (ИИ) появился в 1956 г. и до конца XX в. являлся модным и часто используемым и иногда не к месту. В настоящее время он используется крайне осторожно, а для некоторых является синонимом несбывшихся надежд. Это произошло, на мой взгляд, в основном из-за двух причин: первая – завышенные первоначальные ожидания (с этого начинают почти все новые открытия и идеи); вторая – ИИ является исследовательской областью компьютерной науки и был обречен на то, что все его достижения быстро становилось практическими приложениями и отчуждались в компьютерные приложения, драйверы различных устройств и опции «гаджетов».

Так, уже ставшие обыденными распознавание сотовыми телефонами голосовых команд, сканирование и распознавание текста, в том числе и рукописного, «проговаривание» текста электронными устройствами, автоматическое распознавание автомобильных номеров, машинный (подстрочный) перевод текста, нахождение лиц при автоматических режимах цифрового фотографирования, «интеллектуальные» бытовые устройства некоторое время назад являлись нерешенными задачами ИИ.

Получены и куда более серьёзные результаты, которые нашли своё приложение в советующих и экспертных системах, системах диагностики, управления, проектирования, поддержки принятия решений и т.д.

Настоящее учебное пособие ориентировано не только на некоторые теоретические аспекты систем ИИ, но и на решение связанных с ними практических задач для систем, основанных на знаниях. Приведены также задания на лабораторные работы, примеры и алгоритмы решения таких задач. В их числе описаны и элементы реально работающих программных комплексов.

Глава 1

Историческое предпосылки появления систем, основанных на знаниях в ИИ

Существуют многочисленные определения ИИ, вот одно из них: «Искусственный интеллект можно определить как область компьютерной науки, занимающуюся автоматизацией разумного поведения» [Люгер].

Проблема определения искусственного интеллекта сводится к проблеме определения интеллекта (И) вообще: является ли И чем-то единым, или же этот термин объединяет набор разрозненных способностей? В какой мере интеллект можно воссоздать, а в какой он существует? Что именно происходит при таком создании? Что такое творчество, интуиция? Можно ли судить о наличии интеллекта только по наблюдаемому поведению или же требуется свидетельство наличия некоего скрытого механизма?

Как представляются знания в нервных тканях живых существ и как можно применить это в проектировании интеллектуальных устройств? Необходимо ли создавать интеллектуальную компьютерную программу по образу и подобию человеческого разума или же достаточно строго "инженерного" подхода? Возможно ли вообще достичь разумности посредством компьютерной техники или же сущность интеллекта требует богатства чувств и опыта, присущего лишь биологическим существам? И главный вопрос: ЗАЧЕМ?

Отправной точкой документально оформленным идеям о мышлении человека можно считать логику Аристотеля (384–322 гг. до н.э.). Аристотель рассматривает вопросы истинности суждений на основе их взаимосвязи с другими истинными утверждениями. Хотя формальная аксиоматика логических рассуждений в полном объеме сформулирована лишь в работах Готлоба Фреге, Бертрана Рассела, Курта Геделя, Алана Тьюринга и других, корни этих работ можно проследить вплоть до Аристотеля. Уникальность работ Аристотеля, на мой взгляд, заключается в том, что, возможно, впервые предпринята попытка представления механизма

мышления человека вне его мозга, т.е. на бумаге. Потом задача состояла лишь в замене бумаги компьютером.

Ещё более революционная идея относится к началу XIV в. н.э., высказанная Раймондом Лурием: «Полезные рассуждения можно фактически проводить с помощью механического артефакта» (артефакт – искусственный объект). Вот и первая мысль о возможности создания ИИ. Им предложены «концептуальные колёса» – подобие арифмометра. Далее Томас Гоббс (1588–1679) предположил, что рассуждения аналогичны числовым расчётам. Следующий шаг на пути к ИИ – первая известная вычислительная машина Вильгельма Шиккарда (1623).

Длительное время интеллектуальные способности человека были предметом изучения философов. Материализм утверждал, что разумное поведение складывается из операций, выполняемых мозгом в соответствии с законами физики. Эмпиризм – Джон Локк: «В человеческом понимании нет ничего, что не появлялось бы прежде в ощущениях» (Знания!). Дэвид Юм (1739) предложил принцип, впоследствии названный принципом индукции: общие правила вырабатываются путём изучения повторяющихся ассоциаций между элементами, рассматриваемыми в этих правилах. Джорж Буль в 1847 г. создал логику высказываний, Готтлоб Фреге в 1879 г. логику первого порядка…

Впоследствии ученые и философы поняли, что мышление само по себе как образ представления знаний является трудным, но принципиальным предметом для научного изучения. Поскольку мышление стало рассматриваться как форма вычислений, последующими шагами в его изучении стали формализация и окончательная механизация. В XVIII в. Готфрид Вильгельм фон Лейбниц в работе "CalculusPhilosophicus" представил первую систему формальной логики, а также соорудил машину для автоматизации ее вычислений. Эйлер в начале XVIII в. в своем анализе задачи о кенигсбергских мостах создал учение о представлениях, которые абстрактно отражают структуру взаимосвязей реального мира. Формализация теории графов также сделала возможным поиск в пространстве состояний – основной концептуальный инструмент искусственного

интеллекта. Графы можно использовать для моделирования скрытой структуры задачи. Узлы графа состояний представляют собой возможные стадии решения задачи, ребра графа отражают умозаключения, ходы в игре или другие шаги в решении.

Как один из основоположников науки исследования операций, а также разработчик первых программируемых механических вычислительных устройств, математик XIX в. Чарльз Бэббидж может также считаться одним из первых практиков искусственного интеллекта. "Разностная машина" Бэббиджа являлась специализированным устройством для вычисления значений некоторых полиномиальных функций и была предшественницей его "аналитической машины". Аналитическая машина, спроектированная, но не построенная при жизни Бэббиджа, была универсальным программируемым вычислительным устройством, которое предвосхитило многие архитектурные положения современных компьютеров. Описывая аналитическую машину, Ада Лавлейс, друг Бэббиджа, его помощница и единомышленница, отмечала:

"Можно сказать, что аналитическая машина плетет алгебраические узоры подобно тому, как станок Жаккарда ткет узоры из цветов и листьев. В этом, как нам кажется, заключается куда больше оригинальности, чем в том, на что могла бы претендовать разностная машина". Бэббиджа вдохновляло желание применить технологию его времени для освобождения людей от рутины арифметических вычислений. В этом отношении, как и в представлении о вычислительных машинах как механических устройствах, Бэббидж рассуждал всецело с позиций XIX в. Тем не менее его аналитическая машина также основывалась на многих идеях современности, таких как разделение памяти и процессора ("склад" и "мельница" в терминах Бэббиджа), концепция цифровой, а не аналоговой машины и программировании её, основанном на выполнении серий операций, закодированных на картонных перфокартах. Отличительная черта описания Ады Лавлейс и работы Бэббиджа в целом – это отношение к "узорам" алгебраических взаимосвязей как сущностям, которые могут быть изучены, охарактеризованы, наконец, реализованы и подвергнуты

механическим манипуляциям без заботы о конкретных значениях, которые проходят через "мельницу" вычислительной машины. Это и есть реализация принципа "абстракции и манипуляции формой", впервые описанного Аристотелем.

Целью создания формального языка для описания мышления задавался также Джордж Буль, математик XIX столетия, чью работу необходимо упомянуть при рассмотрении истоков искусственного интеллекта. Хотя Буль внес вклад во множество областей математики, его наиболее известным открытием стала математическая формализация законов логики – свершение, сформировавшее самую сердцевину современных компьютерных наук. Роль булевой алгебры в проектировании логических цепей хорошо всем известна, однако цели самого Буля в разработке его системы по духу ближе к современному ИИ. В первой главе книги "Исследование законов мышления, на которых основываются математические теории логики и вероятностей", Буль описывает свои цели следующим образом:

«Исследовать фундаментальные законы таких операций разума, какими совершается рассуждение, дать им выражение в символическом языке исчисления и на этом основании воздвигнуть науку логики и обучать логическому методу, ...наконец, из различных элементов истины, усмотренной в этих изысканиях, составить некоторые вероятные догадки касательно природы и склада человеческого ума».

Значимость работы Буля состоит в необычайной силе и простоте предложенной им системы. Три операции: "И", "ИЛИ" и "НЕ" составляют ядро его логического исчисления. Эти операции стали базой для последующего развития формальной логики, включая разработку современных компьютеров. Сохраняя значения этих символов практически идентичными, соответствующим логическим операциям, Буль отмечал, что "символы логики относятся к специальному закону, к которому символы количества как таковые не имеют отношения". Булева система не только легла в основу двоичной арифметики, но и показала, что необы-

чайно простая формальная система может передать полную мощь логики. Это предположение и система, разработанная Булем для демонстрации этого факта, стали фундаментом для всех попыток современности формализовать логику и последующих работ Тьюринга.

Готлоб Фреге (Frege) в своих "Основах арифметики" создал ясный и точный язык спецификации для описания основ арифметики. С помощью этого языка Фреге формализовал многие вопросы, затронутые ранее в аристотелевской логике. Язык Фреге, сейчас именуемый исчислением предикатов первого порядка, служит инструментом для записи теорем и задания значений истинности, которые образуют элементы математических умозаключений и описывают аксиоматический базис "смысла" этих выражений. Предполагалось, что формальная система исчисления предикатов, которая включает символы предикатов, теорию функций и квантированных переменных, станет языком для описания математики и ее философских основ. Она также сыграла принципиальную роль в создании теории представления для искусственного интеллекта. Исчисление предикатов первого порядка обеспечивает средства автоматизации рассуждений: язык для построения выражений, теорию, позволяющую судить об их смысле, и логически безупречное исчисление для вывода новых истинных выражений.

Работы Рассела и Уайтхеда особенно важны для фундаментальных принципов ИИ, поскольку заявленной ими целью было вывести из набора аксиом путем формальных операций всю математику. Хотя многие математические системы строились на основе аксиом, интересно отношение Рассела и Уайтхеда к математике как к чисто формальной системе. Это означает, что аксиомы и теоремы должны рассматриваться исключительно как наборы символов, доказательства должны выводиться лишь посредством применения строго определенных правил для манипулирования такими строками. При этом исключается использование интуиции или "смысла" теорем в качестве основы доказательств. Каждый шаг доказательства следует из строгого применения формальных (синтаксических) правил к аксиомам или уже выведенным теоремам, даже если в

традиционных доказательствах этот шаг назывался "очевидным". Смысл, содержащийся в теоремах и аксиомах системы, имеет отношение только к внешнему миру и совершенно не зависит от логического вывода. Такой полностью формальный (реализуемый техническими средствами) подход к математическим умозаключениям предоставил существенную основу для его автоматизации в реальных вычислительных машинах. Логический синтаксис и формальные правила вывода, разработанные Расселом и Уайтхедом, лежат в основе систем автоматического доказательства теорем.

Хотя в XVIII-XIX вв. и начале XX в. формализация науки и математики создала интеллектуальные предпосылки для изучения искусственного интеллекта, он не стал жизнеспособной научной дисциплиной до появления цифровых вычислительных машин. К концу 1940-х гг. электронные цифровые компьютеры продемонстрировали свои возможности в предоставлении памяти и процессорной мощности, требуемой для интеллектуальных программ. Стало возможным реализовать формальные системы рассуждений в машине и эмпирически испытать их достаточность для проявления разумности. Существенной составляющей теории искусственного интеллекта является взгляд на цифровые компьютеры как на средство создания и проверки теорий интеллекта. Но цифровые компьютеры – не только рабочая лошадка для испытания теорий интеллекта. Их архитектура наталкивает на специфичное представление таких теорий: интеллект – это способ обработки информации. Например, концепция поиска как методики решения задач обязана своим появлением в большей степени последовательному характеру компьютерных операций, нежели какой-либо биологической модели интеллекта. Большинство программ ИИ представляют знания на некотором формальном языке, а затем обрабатывают их в соответствии с алгоритмами, следуя заложенному еще фон Нейманом принципа разделения данных и программы. Формальная логика возникла как важный инструмент представления для исследований ИИ, равно как теория графов играет неоценимую

роль в анализе пространства, а также предоставляет основу для семантических сетей и схожих моделей.

Мы часто забываем, что инструменты, которые мы создаем для своих целей, влияют своим устройством и ограничениями на формирование наших представлений о мире. Можно даже сказать, что вещи меняют людей. Такое, казалось бы, стесняющее наш кругозор взаимодействие является важным аспектом развития человеческого знания: инструмент (а научные теории, в конечном счете, тоже инструменты) создается для решения конкретной проблемы. По мере применения и совершенствования инструмент подсказывает другие способы его использования, которые приводят к новым вопросам и, в конце концов, разработке новых инструментов.

Одна из первых работ, посвященных вопросу о машинном разуме в отношении современных цифровых компьютеров–"Вычислительные машины и интеллект", была написана в 1950 г. британским математиком Аланом Тьюрингом. Он задался вопросом: можно ли заставить машину думать? Придуманный им тест сравнивает интеллектуальные способности компьютера имитатора и человека. Об этом он говорил в статье «Могут ли машины мыслить?» (раздел «Игра в имитацию»). Судья задаёт различные вопросы с целью выяснить «кто, есть кто?».

Стандартная интерпретация этого теста звучит следующим образом: «Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы – ввести человека в заблуждение, заставив сделать неверный выбор». Все участники теста не видят друг друга. Если судья не может сказать определенно, кто из собеседников является человеком, то считается, что машина прошла тест. Чтобы протестировать именно интеллект машины, а не её возможность распознавать устную речь, беседа ведется в режиме «только текст», например, с помощью клавиатуры и экрана (компьютера-посредника). Переписка должна производиться че-

рез контролируемые промежутки времени, чтобы судья не мог делать заключения исходя из скорости ответов. Во времена Тьюринга компьютеры реагировали медленнее человека. Сейчас это правило необходимо, потому что они реагируют гораздо быстрее, чем человек.

По состоянию на 2016 г. ни одна из существующих компьютерных систем не приблизилась к прохождению теста, хотя бездоказательные сообщения такого рода появлялись в сети.На самом деле тест носит гипотетический смысл. И зачем же компьютер должен полностью выполнять функции человеческого мозга. Достаточно чтобы он был «умным» там и тогда, когда человеку это действительно нужно.

Благодаря этим преимуществам, тест Тьюринга представляет собой хорошую основу для многих схем, которые используются на практике для испытания современных интеллектуальных программ. Программа, потенциально достигшая разумности в какой- либо предметной области, может быть испытана сравнением ее способностей по решению данного множества проблем со способностями человеческого эксперта. Этот метод испытания всего лишь вариация на тему теста Тьюринга: группу людей просят сравнить "вслепую" ответы компьютера и человека. Как видим, эта методика стала неотъемлемым инструментом как при разработке, так и при проверке современных экспертных систем.

Тест Тьюринга, несмотря на свою интуитивную притягательность, уязвим для многих оправданных нападок. Одно из наиболее слабых мест – пристрастие в пользу чисто символьных задач. Тест не затрагивает способностей, требующих навыков перцепции или «ловкости рук», хотя подобные аспекты являются важными составляющими человеческого интеллекта. Иногда же, напротив, тест Тьюринга обвиняют в попытках втиснуть машинный интеллект в форму интеллекта человеческого. Быть может, машинный интеллект просто настолько отличается от человеческого, что проверять его человеческими критериями – фундаментальная ошибка? Нужна ли нам, в самом деле, машина, которая бы решала математические задачи так же медленно и неточно, как человек? Не должна ли разумная машина извлекать выгоду из своих преимуществ, таких как

большая, быстрая, надежная память, и не пытаться имитировать человеческое познание? На самом деле, многие современные практики ИИ говорят, что разработка систем, которые бы выдерживали всесторонний тест Тьюринга, – это ошибка, отвлекающая нас от более важных, насущных задач: разработки универсальных теорий, объясняющих механизмы интеллекта людей и машин и применение этих теорий к проектированию инструментов для решения конкретных практических проблем. Все же тест Тьюринга представляется м важной составляющей в тестировании и "аттестации" современных интеллектуальных программ.

Тьюринг также затронул проблему осуществимости построения интеллектуальной программы на базе цифрового компьютера. Размышляя в терминах конкретной вычислительной модели (электронной цифровой машины с дискретными состояниями), он сделал несколько хорошо обоснованных предположений касательно ее объема памяти, сложности программы и основных принципов проектирования такой системы. Наконец, он рассмотрел множество моральных, философских и научных возражений возможности создания такой программы средствами современной технологии. Отсылаем читателя к статье Тьюринга за познавательным и все еще актуальным изложением сути споров о возможностях интеллектуальных машин.

Два возражения, приведенных Тьюрингом, стоит рассмотреть детально. "Возражение леди Лавлейс", впервые сформулированное Адой Лавлейс, сводится к тому, что компьютеры могут делать лишь то, что им укажут, и, следовательно, не могут выполнять оригинальные (читай: разумные) действия. Однако экспертные системы, особенно в области диагностики, могут формулировать выводы, которые не были заложены в них разработчиками. Многие исследователи считают, что творческие способности можно реализовать программно.

Другое возражение, "аргумент естественности поведения", связано с невозможностью создания набора правил, которые бы говорили индивидууму, что в точности нужно делать при каждом возможном стечении обстоятельств. Действительно, гибкость, позволяющая биологическому

разуму реагировать практически на бесконечное количество различных ситуаций приемлемым, если даже и не оптимальным образом,– отличительная черта разумного поведения. Справедливо замечание, что управляющая логика, используемая в большинстве традиционных компьютерных программ, не проявляет великой гибкости или силы воображения, но неверно, что все программы должны писаться подобным образом. Большая часть работ в сфере ИИ за последние 30 лет была направлена на разработку таких языков программирования и моделей, призванных устранить упомянутый недостаток, как продукционные системы, объектные системы, сетевые представления и другие модели.Следует также отметить, что Тьюринг заблуждался в духе своего времени в том, что ИИ будет создан при достижении некоторых объёмов памяти и мощности вычислителей. По его прогнозам, это должен быть 2015 год. И как мы видим, дело обстоит не только в количественных характеристиках.

Современные программы ИИ обычно состоят из набора модульных компонентов или правил поведения, которые не выполняются в жестко заданном порядке, а активизируются по мере надобности в зависимости от структуры конкретной задачи. Системы обнаружения совпадений позволяют применять общие правила к целому диапазону задач. Эти системы необычайно гибки, что позволяет относительно маленьким программам проявлять разнообразное поведение в широких пределах, реагируя на различные задачи и ситуации.

Можно ли довести гибкость таких программ до уровня живых организмов, все еще предмет жарких споров. Нобелевский лауреат Герберт Саймон сказал, что большей частью своеобразие и изменчивость поведения, присущие живым существам, возникли скорее благодаря сложности их окружающей среды, чем благодаря сложности их внутренних "программ". Саймон описывает муравья, петляющего по неровной, пересеченной поверхности. Хотя путь муравья кажется довольно сложным, Саймон утверждает, что цель муравья очень проста: вернуться как можно скорее в колонию. Изгибы и повороты его пути вызваны встречаемыми препятствиями. Саймон заключает, что: "Муравей, рассматриваемый в

качестве проявляющей разумное поведение системы, на самом деле очень прост. Кажущаяся сложность его поведения в большей степени отражает сложность среды, в которой он существует". Здесь, на мой взгляд, свойство разумности, приданное биологическим организмам, стоящим на ступени эволюции ниже человека и даже группам таких особей, не является продуктивным на пути создания ИИ. Как и включение эволюционных алгоритмов и им подобным в совокупность методов ИИ. Однако, если такие алгоритмы показывают хорошие результаты по тесту Тьюринга, то можно и забыть о их происхождении. Возможно выяснится, что они близки к «алгоритмам» мышления человека. А может мы хотим пройти на пути к ИИ от простейших дорогой эволюции?

Эта идея, если удастся доказать применимость ее к организмам с более сложным интеллектом, составит сильный аргумент в пользу простоты, а следовательно, постижимости интеллектуальных систем. Любопытно, что, применив эту идею к человеку, мы придем к выводу об огромной значимости культуры в формировании интеллекта. Интеллект, похоже, не взращивается во тьме, как грибы. Для его развития необходимо взаимодействие с достаточно богатой окружающей средой. Культура так же необходима для создания человеческих существ, как и человеческие существа для создания культуры. Эта мысль не умаляет могущества наших интеллектов, но подчеркивает удивительное богатство и связь различных культур, сформировавших жизни отдельных людей. Фактически на идее о том, что интеллект возникает из взаимодействий индивидуальных элементов общества, основывается подход к ИИ.

Одним из наиболее ранних проектов в создании аппаратной функциональной модели мозга человека является создание персептрона. Персептрон или персептрон (англ. perceptron от лат. perceptio– восприятие) – математическая и компьютерная модели восприятия информации мозгом, предложенная Фрэнком Розенблаттом в 1957 г. и реализованная в виде электронной машины Марк-1 в 1960 г. Перцептрон стал одной из первых моделей нейросетей, а Марк-1 – первым в мире нейрокомпьютером. Не-

смотря на свою простоту, перцептрон способен обучаться и решать довольно сложные задачи. Марк-1 представлял собой модель зрительного анализатора и способен был распознавать некоторые из букв английского алфавита (!). В персептроне поступающие от сенсоров сигналы передаются ассоциативным элементам, а затем реагирующим элементам. Таким образом, перцептроны позволяют создать набор «ассоциаций» между входными стимулами и необходимой реакцией на выходе. В биологическом плане это соответствует преобразованию, например, зрительной информации в физиологический ответ от двигательных нейронов. Согласно современной терминологии, персептроны могут быть классифицированы как искусственные нейронные сети. На фоне роста популярности нейронных сетей, в 1969 г. вышла книга Марвина Минского и Сеймура Паперта, которая показала принципиальные ограничения персептронов. Это привело к смещению интереса исследователей искусственного интеллекта в противоположную от нейросетей область символьных вычислений. Кроме того, из-за сложности математического анализа персептронов, а также отсутствия общепринятой терминологии, возникли различные неточности и заблуждения. Впоследствии интерес к нейросетям, и в частности, работам Розенблатта, возобновился [Люгер].

Следующим, достаточно громким проектом, стала разработка программы GPS (не навигатора). Аллен Ньюэлл и Герберт Саймон в 1961 г. разработали программу, которую назвали универсальным решателем (GeneralProblemSolver). Исследователи её продолжали считать, что смоделировав структурно мозг человека или логику его рассуждения (как он сам эту логику представляет) можно достичь эффективности мышления человека.

«Японский вызов миру» – это приметный текст заголовков статей журналов и газет 1982 г. В ответ на этот так называемый вызов были открыты подобные проекты в Европе и США. Аналогичные работы проводились и в Советском Союзе, но назвать их «проектом» было нельзя. Можно сказать, что эти работы были самодеятельными. Вызовом являлся проект по созданию ЭВМ 5-го поколения. Ну и в чёмже состоял вызов?

После реализации этого проекта Япония намеревалась прекратить производство промышленной продукции и стать мировым хранилищем и генератором знаний и данных. Согласитесь, нешуточной являлась угроза.

Основные положения проекта были опубликованы в начале 90-х гг. прошлого века. Движение по создании ЭВМ 5-го поколения предполагался сразу в 4-х направлениях: элементная база (высокая степень интеграции микросхем), архитектура (должна позволять выполнять за один машинный такт один логический вывод), способы представления знаний и технология программирования. Обычно в те времена всё начиналось с разработки нового процессора, потом некоторое время процессор оставался бесхозным, потом разрабатывался компьютер, а технология программирования всегда жила своей собственной жизнью. Поскольку элементная база не входит в круг вопросов настоящего рассмотрения, начнём с предполагаемой архитектуры. В настоящее время быстродействие персональных компьютеров измеряется частотой процессора, что напрямую не является показателем быстродействия. На пути от частоты до реального быстродействия стоят разрядность процессора и шины данных, размеры оперативной памяти, архитектура и загрузка процессора системными приложениями… В 90-е гг. быстродействие ЭВМ оценивалось скоростью выполнения команд специальных смесей для расчётных задач (в основном работа процессора) и экономических, статистических задач (большая доля операций обращения к памяти). Это смеси Гибсон-1 и Гибсон-2. В то время быстродействие «топовых», наиболее производительных майнфреймов являлось порядка одного миллиона операций на смеси Гибсон-1. Такого же порядка была тактовая частота первых бытовых игровых компьютеров. Реально же это были совсем разные миллионы. Такой обман стал процветать при широком распространении компьютеров в быту. Быстродействие ЭВМ-5 предполагалось измерять в ЛВС – логических выводах в секунду. Один ЛВС оценивался порядка 10 000 элементарных операций обычной ЭВМ. Причём ЛВС должен был стать для ЭВМ-5 элементарной операцией. В качестве языка ассемблера был

выбран язык ПРОЛОГ. В связи с этим, архитектура ЭВМ должна осуществлять выполнение в качестве элементарных машинных команд «команды» ПРОЛОГа. Причём реализация этого амбициозного проекта должна пройти в несколько этапов, на каждом из которых должен быть создан полнофункциональный компьютер, на котором в свою очередь будет проектироваться компьютер следующего уровня. Результатом реализации этого амбициозного проекта явилось сообщение о создании малогабаритной высокопроизводительной ЭВМ порядка нескольких миллионов ЛВС. Дальнейшие разработки в направлении создания компьютеров, ориентированных на логические выводы либо прекращены, либо засекречены. Однако ещё до этого проекта в США существовали ЛИСП- машины- ЭВМ, в качестве языка нижнего уровня являлся язык программирования ЛИСП.

Таким образом, как видно из предыдущих примеров, многие попытки создания технических средств, моделирующих либо структурно, либо функционально мозг человека в целом, не дали тех результатов, на которые были рассчитаны. Оставались надежды на создание программных структур или моделей.

Очень трудный и пока ещё не опровергнутый вывод из не совсем удачных попыток исследований в области ИИ 70- 90-х гг. прошлого века следующий: **«Достигнуть с помощью компьютера результатов, посильных возможностям человеческого мозга, можно только в узкой предметной области и только используя его, человека, знания».** Здесь и, наверное, ранее тоже возникает мысль: «А зачем нужно догонять гений человека, если он есть сам?». Подчеркнём несколько аспектов нужности переложения человеческих мыслительных способностей на компьютеры: необходимость тиражирования знаний и умений лучших всем остальным; умножение человеческих знаний и умений на большее число процессов, чем доступно человеку; тиражирование знаний и умений человека туда, где пребывание человека невозможно или опасно.

Термин «знания» в нашем подходе к созданию систем искусственного интеллекта требует предварительного уточнения. Существуют многочисленные определения термина «знание». И в силу того, что люди различных профессий (философы, лингвисты, физиологи…) дают его различные определения и сам предмет описания является только «тем, благодаря чему или вопреки чему» дают очень разные, зачастую, непонятные другим специалистам определения. Возможно, такая субстанция не существует но, человек, напуганный таинственными своими способностями, решил спрятать это таинственное и такое необходимое за термином «знание». Но всё же нам придётся с этим разбираться. Иногда под знанием подразумевают нечто, противоположное вере. «Знать» по-старославянски – знати, делать известным. «Знать»– высшее общество – например, придворное общество… «Знать»– те, кто имеют знак, возможно герб.

Существуют различные варианты употребления слова “знать”: знать (кого)– быть знакомым с кем-либо – Иванова, Петрова,…(глагол- сказуемое); знать– иметь о чём-то информацию – расписание поездов (глагол); знать как… (изъяснительный союз); знать что … (изъяснительный союз); знать (о чём?) о том что…; знать (о ком?)…; знать где…; знать куда…; знать откуда…; знать сколько…; знать зачем…; знать когда… (временной союз).

Попробуем дать определение «знания», отталкиваясь от более простого понятия «данные». Проследим эволюцию данных в программировании компьютеров от информации к знаниям. То есть здесь мы говорим не о знаниях человека, а о том, как человек сам их представляет в компьютере. Двоичный код в ячейке памяти (только программа(мист) знает, что с ним делать). Определение значения переменной или константы через операцию присваивания. Разделение описания типов данных от их значений (сами значения в конце программы в особых служебных знаках). Вынесение данных в отдельные массивы вне программы на других носителях. Создание отдельных структур для хранения данных вместе (так кажется пользователю) с их описанием – БД и возможность доступа

к ним без написания программы– СУБД (Реляционная БД– начало свойству внутренней интерпретации). И вот здесь появляется особый тип данных с особыми свойствами– «знания». Представим, чем же отличаются знания от данных через их свойства (Поспелов Д.А.).

Внутренняя интерпретация предусматривает наличие внутренней структуры связей (множество элементов, их иерархичность, наличие разнотипных связей). В этой части рассуждения будем называть такой элемент знаний фреймом.

Наличие внешней структуры связей определяет наличие взаимодействий между элементарными структурами знаний, фреймами.

Шкалирование предусматривает упорядоченность в соответствии с некоторой структурой– шкалой. Для знаний могут использоваться различные типы шкал: абсолютные метрические шкалы, например с привязкой к сотворению мира, Рождеству Христову; относительные метрические шкалы– «через 2 года вы окончите ВУЗ»; порядковые шкалы– «неудовлетворительно, удовлетворительно, хорошо, отлично»; размытые порядковые шкалы (лингвистические)– «никогда», «иногда», «часто», «всегда».

Погружение в пространство с «семантической метрикой». Упорядочение сведений в когнитивных структурах человека происходит не только благодаря применению шкал, но и погружению в некоторое пространство, метрики которого характеризуют семантическую (смысловой) близость понятий. Ещё в 20-х гг. прошлого века американский психолог Ч. Оствуд с учениками предложил построить трехмерное пространство, на одну ось которого проецируются оппозиционные шкалы, обладающие оценочным характером типа “хороший- плохой”, “добрый - злой”, “красивый - безобразный”, на другую – “тяжелый- легкий”, “сильный - слабый”, “большой -маленький”, на третью –“острый- тупой”, “быстрый-медленный”, “короткий- длинный” … Около 400 шкал. И задавая вопросы на разных языках, просили носителей языка разместить на этих шкалах слова- понятия. Получились кластеры слов в этом пространстве.

Например, отец, мать, дети попали в один кластер –ситуация семья. Стулья, парты, тетради, ручки, преподаватель–ситуация лекция. Есть так же предположение, что помимо ситуативной системы оценки близости существует частотная система появления тех или иных представителей в типовых ситуациях. Поэт– Пушкин, инструмент– молоток…. На мой взгляд, это два проявления одного и того же свойства мозга человека.

Наличие активности. Программа (процедурное знание) отделена от данных (начальный этап декларативных знаний). В компьютере процедурная часть активизирует декларативную, в голове, скорее всего, наоборот. Фрейм.

Ньюэлл (Newell) и Саймон (Simon) в лекции по случаю вручения премии Тьюринга в Ассоциации специалистов по компьютерной технике (1976) доказывают, что интеллектуальная деятельность как человека, так и машины, осуществляется с использованием следующих средств:

1. Символьные шаблоны (комбинации символов), предназначенные для описания важнейших аспектов области определения задачи.

2. Операции с этими шаблонами, позволяющие генерировать потенциальные решения проблем.

3. Поиск с целью выбора решения из числа всех возможных.

Что такое символ? Это нечто замещающее другое нечто (Питер Джексон). Описанные предположения формируют базис гипотезы о физической символьной системе (physical symbol system hypothesis). Эта гипотеза лежит в основе наших попыток создания умных машин и делает очевидными основные предположения в исследовании искусственного интеллекта. Гипотеза о физической символьной системе неявно различает понятия шаблонов (patterns), сформированных путем упорядочения символов, и среды (medium), в которой они реализованы. Если уровень интеллекта определяется исключительно структурой системы символов, то любая среда, которая успешно реализует правильные шаблоны и процессы, достигнет этого уровня интеллекта, независимо от того, составлена она из нейронов, логических цепей, или это просто механическая игрушка. Возможность построения машины, которая бы прошла тест

Тьюринга, зависит от вышеупомянутого разграничения. Согласно тезису Черча о вычислимости по Тьюрингу, компьютеры способны осуществить любой эффективно описанный процесс обработки символьной информации. Разве из этого не следует, что должным образом запрограммированный цифровой компьютер обладает интеллектом?

Знак – минимальный носитель языковой информации. Совокупность знаков образует знаковую систему, или язык (см. семиотика). Знак представляет собой двустороннюю сущность. С одной стороны, он материален (имеет план выражения, или денотат), с другой – он является носителем нематериального смысла (план содержания). Структуру знаний удобно представлять в виде так называемого треугольника Фреге (ок.1878). Готлиб Фреге – немецкий философ и логик, один из основателей логической семантики. Выглядит треугольник Фреге как показано на рис. 1.

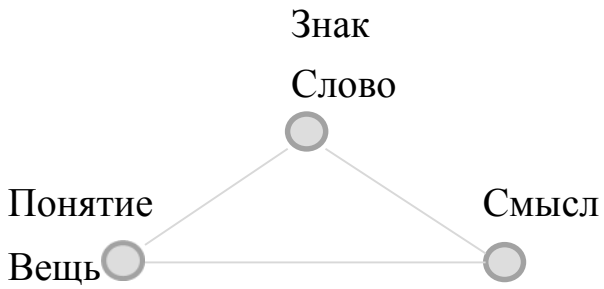


Рис. 1. Треугольник Фреге

Гипотеза о физической символьной системе также вкратце описывает главные вопросы исследования в области искусственного интеллекта и разработки его приложений. К ним относится определение структур символов и операций, необходимых для интеллектуального решения задачи, а также разработка стратегий для эффективного и правильного поиска потенциальных решений, сгенерированных этими структурами и операциями. Эти взаимосвязанные проблемы представления знания и поиска лежат в основе современных исследований в области искусственного интеллекта.

Гипотеза о физической символьной системы оспаривается критиками, которые утверждают, что интеллект является наследственно биологическим и экзистенциальным и не может быть зафиксирован с помощью символов. Ими продвигаются такие направления исследований, как развитие теории нейронных сетей, генетических алгоритмов и агентоориентированных методов (подход "агент-действие"). Несмотря на вышеуказанные возражения, предположения гипотезы физической символьной системы лежат в основе почти всех практических и теоретических работ в экспертных системах, в планировании и понимании естественного языка.

Таким образом, можно сделать вывод, что знания– это сложная структура, они должны быть предметом изучения и нам нужно научиться их использовать при проектировании, разработки информационных систем.

Глава 2
Представление темпоральных знаний в интеллектуальных системах

Во многих информационных и аналитических системах, системах поддержки принятия решений, экспертных системах, автоматизированных системах управления особое внимание уделяется учёту последовательности и взаимосвязи событий и процессов как в реальном масштабе времени, так и при планировании действий или рассмотрении исторических трендов событий и процессов. Представлению темпоральных (временных) компонентов знаний невозможно без учёта свойств времени.

2.1. Основные положения представления темпоралных характеристик процессов

Время рассматривается исследователями в многочисленных аспектах: философском (логическом, онтологическом), естественнонаучном (физическом, психологическом) и даже этическом [Домбровский]. Направление «течения» времени также вызывает философские споры,

как и многие другие аспекты времени. Если учитывать, что все больше событий из будущего становятся достоянием настоящего и далее прошлого, то можно предположить, что время «течет» (или мы движемся) в будущее, и значит, из прошлого через настоящее. Конечно, интуитивно мы считаем неизменными положение компонентов в тройке <«прошлое», «настоящее», «будущее»>, изменяется только содержимое каждой компоненты. Существование фатального «будущего» требует признания того, что события и процессы находятся в «будущем» в той же последовательности, в какой они произойдут в «настоящем» и будут «храниться» в прошлом. Если мы каким-либо образом «управляем» событиями и процессами в течение времени своей жизни, то проще представить будущее в виде хаотического множества событии, которых мы можем избежать, ускорить или замедлить их наступление и течение. Однако этот хаос частично упорядочен, и события и процессы «сведены» в подмножества в соответствии с возможностями, определяемыми законами бытия (биологическими, физическими, социальными...) и ограничениями (сословными, национальными, религиозными...). Каждое из этих событий и каждый из процессов может характеризоваться некоторой оценкой вероятности наступления и оценкой «удаленности» от «настоящего». Это справедливо только в том случае, если мы признаем существование будущего как временного «хранилища» всех событии и процессов.

Если «содержание» «будущего» составляют еще не произошедшие события «прошлого», то «настоящее» вообще существует благодаря объединению результатов работы органов чувств. «Настоящим» человек называет самое близкое «прошлое». И в самом деле, то, что можно отнести к категории «сейчас», уже произошло и, естественно, относится к «прошлому». Итак, «будущее», скорее всего, не существует, а если и существует, то состоит из элементов «прошлого», «настоящее» есть тоже хоть и ближайшее, но скорее «прошлое», т.е. то, о чем можно реально рассуждать, это только о «прошлом» и его подмножествах в виде «настоящего» и «будущего».

Для рассмотрения времени как свойства некоторых сущностей известны следующие подходы: 1) рассматривать время наряду с другими атрибутами без учета его характерных свойств; 2) рассматривать время «через призму» его характерных свойств.

Используя первый подход, можно считать, что есть некоторый терминальный (ни от чего не зависящий) процесс– «хранитель» времени, результатом которого является значение некоторых атрибутов-носителей текущего времени. В свою очередь, атрибуты-носители текущего времени могут быть носителями абсолютного поясного времени, носителями оперативного и относительного времени, а также носителями времени в пределах хранимых отрезков времени. Сложные технические и технологические системы содержат подсистемы «единого времени» (СЕВ), обеспечивающие синхронизацию событий и процессов. Примеры таких систем являются АСУ ТП (автоматизированные системы управления технологическими процессами), в них существуют многочисленные иерархические процессы, распределенные в пространстве и времени. Однако для обозначения времени используются обычные переменные или поля тегов. Время используется как атрибут событий или как уставное значение для контролируемых процессов. Временная шкала, используемая для представления трендов, обычно масштабируется. Но существуют приложения, в которых время играет первостепенную роль. В частности, для выявления некоторых закономерностей в сложных процессах требуется анализ временных компонентов событий.

Временные диаграммы используются в основном как средство представления синхронизации процессов чаще всего в не программируемых цифровых и аналоговых устройствах. Это преимущественно детерминированные процессы.

Для анализа сложных процессов с целью выявления возможных отношений между событиями были бы полезны сведенные вместе временные тренды нескольких событий. Для этого необходимо выявить возможные отношения, которые могут существовать между временными компо-

нентами. Такие событийные или «временные» тренды должны быть доступны как экспертному, так и автоматизированному или автоматическому анализу.

Исторические тренды представляют собой отображение изменения значений некоторого атрибута в течение отрезка времени, не включающего текущий момент времени. Текущие тренды включают в качестве последнего по времени значения атрибута его значение в настоящим момент времени. Тренды будущего представляют либо предполагаемые значения некоторого информационного атрибута в будущем, либо запланированные изменения значения управляющего атрибута.

Основными свойствами времени, порожденными представлениями человека о времени, многие авторы объявляют следующие [Домбровский]:

1) структура времени (линейная, ветвящаяся, циклическая);
2) свойства шкалы времени (непрерывность, дискретность);
3) обусловленность моментов времени (детерминизм, индетерминизм).

Некоторые авторы выделяют иные свойства времени: упорядоченность, возможность определить отношение «раньше/позже» между событиями/процессами, течение времени, универсальность времени, необратимость, не фиксированность будущего. Что же собой представляет шкала времени? Любая шкала (особенно в метрическом ее понимании) является переносом представления человека о пространстве на другие атрибуты, воспринимать которые непосредственно с помощью органов чувств человек не может.

Для иллюстрации терминов, которые человек использует при описании темпоральных компонентов, представим группы терминов (темпоральные теории) событий и процессов на рис. 2. Событие представляется только одной точкой на временной шкале– временем, когда оно произошло, а процесс–двумя: временем начала и окончания. Будем использовать следующие группы терминов: «раньше–позже», «прошлое–настоящее–будущее», «было–есть–будет».

Фактически на рисунке представлены темпоральные связи между терминами и по ним можно восстановить следующие высказывания человека относительно представленных событий,например:

1. «Событие 2 (всегда) «будет» «позже» события 1».
2. «Событие 2 «есть» в «будущем»».
3. «Событие 2 (когда-нибудь) «будет» в «настоящем», но чуть «позже».
4. «Событие 2 (когда-нибудь) «будет» в «прошлом»».

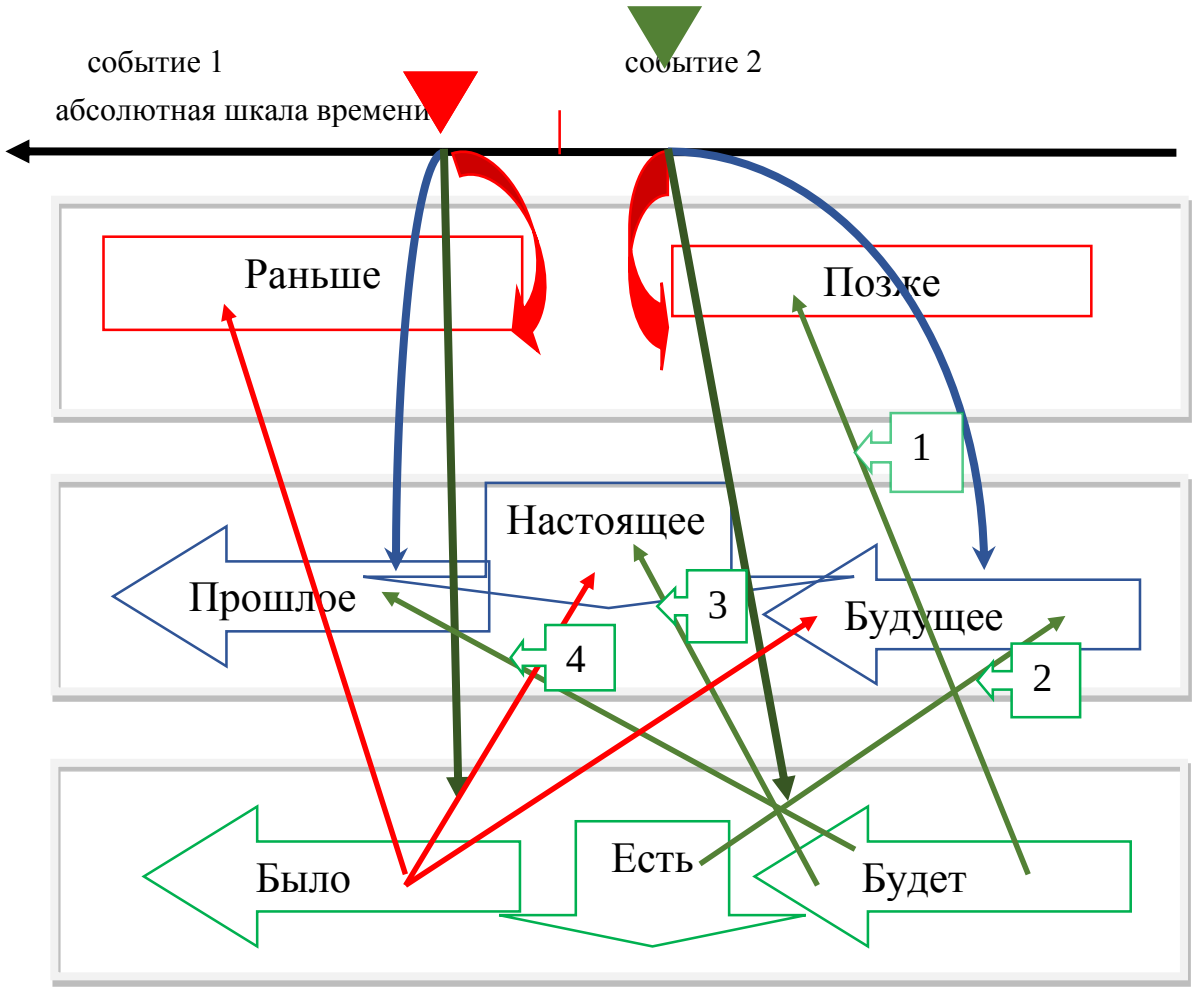


Рис. 2. Позиционирование событий на временных шкалах

В случае с пространством линейная шкала – это уложенные друг за другом одинаковые (эталонные) линейные элементы самого пространства. А само измерение заключается в подсчете элементов, которые можно разместить между двумя точками. Со временем дела обстоят еще сложнее. Само время мы измеряем с помощью событий или процессов,

которые обладают стабильностью по отношению к данному их свойству. Время, как таковое, может не существовать, а лишь являться некой родственной общностью свойств разнотипных сущностей. Сравнивая идентичные события или процессы, мы используем категорию времени для того, чтобы их отличить: одно из них уже произошло, другое еще происходит (длится). Поскольку существование времени само по себе считается спорным, мы будем рассматривать его как свойство или атрибут некоторого события или процесса.

Темпоральные отношения могут быть формализованы в конечное множество отношений либо представлять высказывания на естественном языке. Объектами темпоральных отношений могут быть события или процессы. Под событиями будем понимать факт или результат. Событие принято характеризовать одним отсчетом времени (темпором) – временем, когда оно произошло. Процесс принято характеризовать моментом (темпором) начала и окончания и, как следствие, длительностью.

2.2. Темпоральные отношения

Определение события как результата действий единичного субъекта [Кандрашина и др.] не совсем полно, поскольку исключает из рассмотрения действия двух или нескольких субъектов, объектов и систем. Событием может быть не обязательно (только) смена состояний объекта, но и интересующее нас сочетание текущих значений атрибутов, характеризующих объект. Иногда принадлежность события объекту(ам) и субъектам трудно определить, или она нас не интересует.

Некоторые элементы события/процесса могут быть описаны нечеткими темпорами, т.е. такими, границы которых нечетко определены. В этом случае следует рассматривать отношения четкого и нечеткого или отношения двух или нескольких нечетких темпоров (τ_i, τ_j). Ниже приведены два типа элементарных темпоральных высказываний:

1. E$R\tau$– между событием и темпором.
2. $\tau_i R \tau_j$– между двумя темпорами.

Для практических целей полезно использовать составные темпоральные высказывания, которые представляют совокупность элементарных темпоральных высказываний, объединенных при помощи логических связок.

Рассмотрим отношения между интервальными темпорами. Точечные темпоры можно рассматривать как интервальные с совпадающими началами и окончаниями и нулевом длительностью. Все рассматриваемые отношения будут двухместными. В основе рассматриваемых отношений лежит группа интервально-временных отношений ECT [Кандрашина и др.] темпоральной логики. Темпоры можно сравнивать по длительности и по взаимному расположению.

Два темпора могут находиться в следующих отношениях:

– (rts) – последовательны с паузой, если интервал первого процесса закончился раньше, чем начался интервал второго (они не одновременны, и первый интервал раньше второго)

– (rtsn) –последовательны без паузы;

– (rtes)– пересекаются;

– (rtel) –вложенные с примыканием к началу;

– (rter) –вложенные с примыканием к окончанию;

– (rte) –вложенные без примыканий;

– (rtU) – отношение несравнимости темпоров.

Будем считать, что два темпора находятся между собой в отношении rtU, если:

1) ничего не известно о процессах, которые характеризуются этими темпорами, что свидетельствовало бы о других темпоральных отношениях между этими темпорами;

2) необходимо предпринять некоторые действия, чтобы темпоральные отношения стали более определенными.

2.3. Разработка и исследование «темпорального процессора» для информационно- аналитических систем

Цель работы: Приобрести навыки представления и использования темпоральных компонентов знаний в информационных и управляющих системах.

Задачи: Научиться представлять темпоральные атрибуты знаний; разрабатывать алгоритмы поиска темпоральных характеристик в реляционных базах данных; разрабатывать интерфейсы пользователей для имитационных моделей; разрабатывать алгоритмы анализа темпоральных отношений для различных позиций наблюдения (наблюдателя).

Порядок выполнения работы: Рассмотрение теоретической части задания; представление общей структуры теморального процессора; разработка алгоритмов моделирования темпоральных процессов; разработка алгоритмов анализа темпоральных процессов (темпорального процессора), выполнение чертежей (эскизов) экранных форм; написание и отладка программы моделирования темпоральных процессов; написание и отладка программы темпорального процессора; выполнение темпоральных (вычислительных) экспериментов; анализ правдоподобности результатов вычислительного эксперимента; выполнение отчетных материалов.

Задание на выполнение работы: Разработать формат представления темпоральных данных для последующего анализа в информационных и управляющих системах; разработать алгоритм поиска темпоральных характеристик в реляционных базах данных; разработать интерфейс пользователя имитационной модели темпоральных характеристик процессов; разработать имитационную программу для имитации пяти тем-

поральных процессов и позиции наблюдателя; разработать алгоритм темпоральных отношений пар процессов; разработать формат записи и хранения темпоральных отношений пар процессов.

Позиция наблюдателя представляет момент рассмотрения процессов. Позиция наблюдателя представляется не протяженным моментом времени и, следовательно, аналогична событию.

Форма отчетности. По результатам выполнения лабораторной работы оформляется отчёт, состоящий из титульного листа с указанием города, года, организации и подразделения, в котором выполнена работа, названия работы, фамилии И.О. и учебной группы исполнителя, фамилии И.О. руководителя работы; описания целей и задач работы; описания методов, используемых при решении задач; алгоритмов и форматов данных; указанием выбранных языков программирования; скриншотов экранных форм и рисунков форматов данных; описания экспериментов, подтверждающих (демонстрирующих) выполнение поставленных в работе задач; текст моделирующей программы с комментариями.

Защита работы заключается в предоставлении отчета и ответа на вопросы по содержанию отчёта, демонстрации работы программного темпорального процессора, ответа на теоретические вопросы.

Пример реализации пользовательского интерфейса «темпорального процессора»

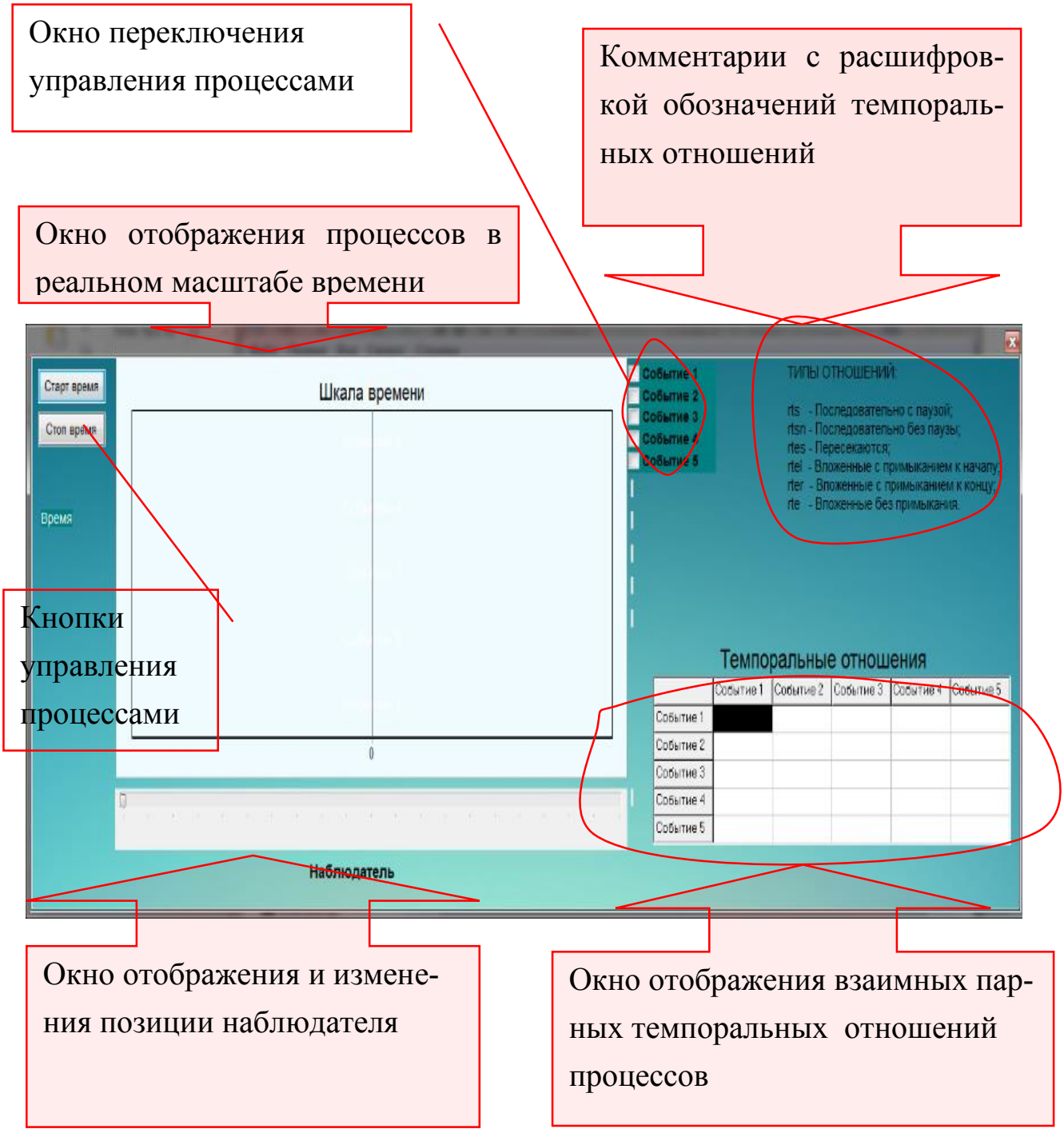


Рис. 3. Функциональное назначение компонентов экранной формы

На данном этапе мы выбираем одно или несколько событий из списка сверху и запускаем таймер времени.

Дополнительно указаны положения процессов на шкале «прошлое-настоящее-будущее»

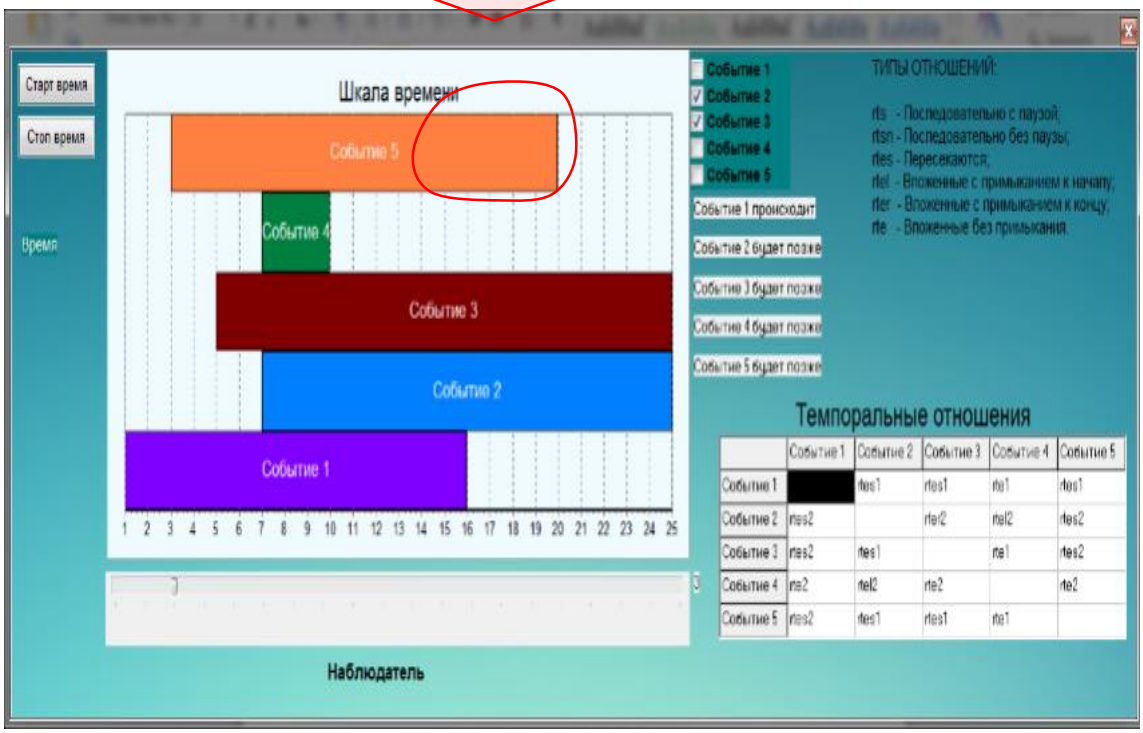


Рис. 4. Экранная форма с результатами эксперимента

В данном процессоре темпоральных отношений мы можем менять положение наблюдателя, и программа будет показывать нам, какое событие закончилось, какое идет, а какое еще не произошло.

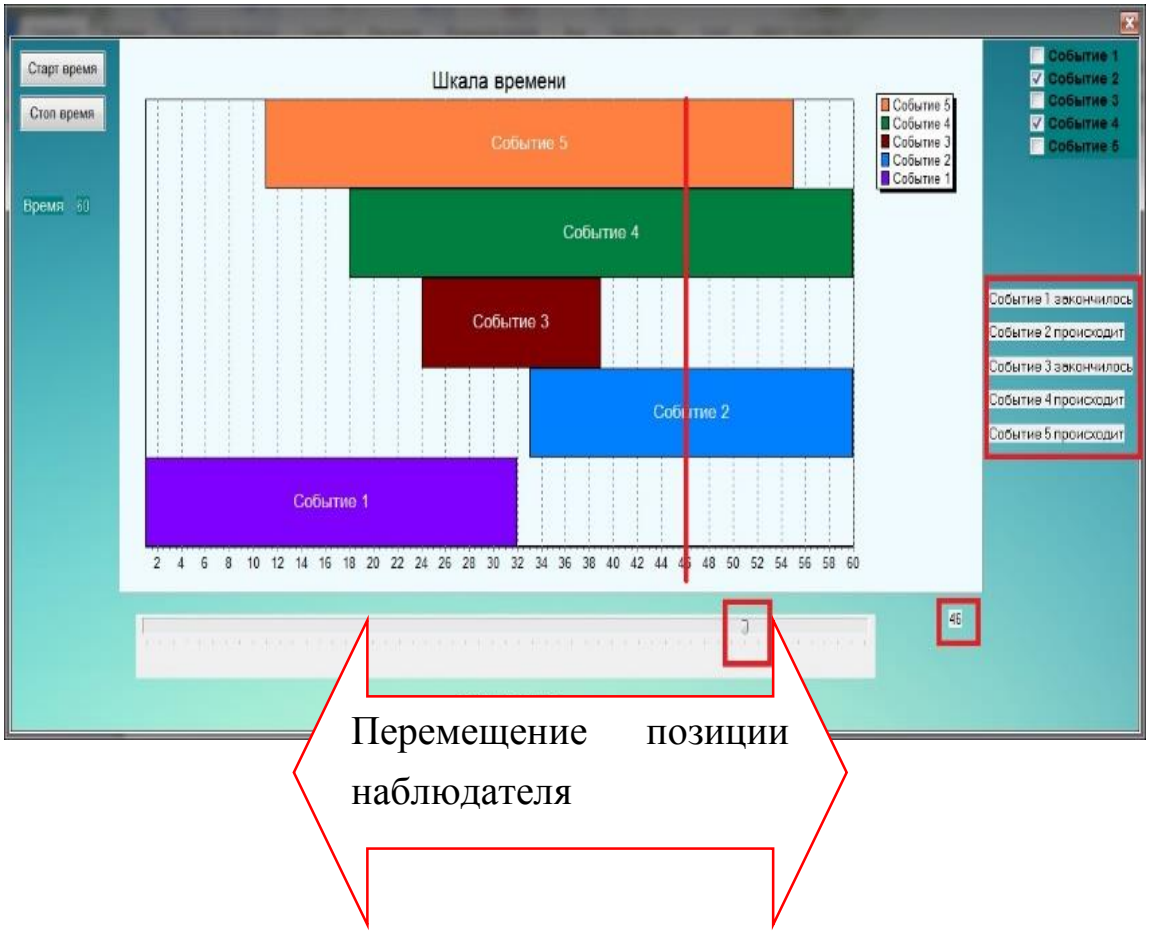


Рис. 5. Демонстрация положения и перемещения позиции наблюдателя на временной шкале

	rte1	rtel2	rte1	rtes1
rte2		rtes2	rtes1	rtes1
rtel2	rtes1		rtes1	rtsn1
rte2	rtes2	rtes2		rtes1
rtes2	rtes2	rtsn2	rtes2	

Рис. 6. Таблица попарных отношений процессов

Глава 3

Использование лингвистических переменных в системах, основанных на знаниях

3.1. Представление атрибутов в виде лингвистических переменных

Нечеткие атрибуты формализуются с помощью нечетких переменных. Нечеткой переменной называется тройка объектов $< \alpha, A, \tilde{C}(\alpha) >$ [Берштейн и др., 1990], [Борисов и др.], где α – имя некоторой переменной, А – базовое множество или область определения нечеткой переменной, $\tilde{C}(\alpha) = \{\mu_C(\alpha)/\alpha\}$ – нечеткое подмножество множества А, описывает область значений нечеткой переменной, $\mu(\alpha) \in [0; 1]$ – значения функции принадлежности. То есть, если А-некоторое дискретное числовое множество, $\tilde{C}(\alpha) = \{\mu_C(\alpha)/\alpha\}$ фактически представляет собой множество точек на плоскости с координатами $(\alpha_i, \mu_i(\alpha_i))$. А для случая если нечеткое множество определено на непрерывном числовом множестве, это множество точек представляет некоторую функцию $\mu_i(\alpha_i)$.

Для некоторых атрибутов предметной области могут потребоваться описания лингвистических значений. Лингвистические значения атрибутов формализуются с помощью лингвистических переменных. Лингвистической переменной [Борисов] называется тройка $< \beta, A, T(\beta) >$, гдеβ– имя лингвистической переменной, А – базовое множество или область определения лингвистической переменной, $T($$\beta$– терм-множество лингвистической переменной β, $T(\beta) = \{< \gamma_i, \tilde{C}_i(\alpha) >\}$, γ_i – имя терма,$\tilde{C}_i(\alpha) = \{\mu_{C_i}(\alpha)/\alpha\}$ – нечеткое множество, описывающее область значений терма γ_i.

Интерпретацию понятия нечеткого множества $\tilde{C}_i(\alpha) = \{\mu_{C_i}(\alpha)/\alpha\}$, используемого при задании нечетких и лингвистических переменных, необходимо давать исходя из процессов, лежащих в основе формирования значений описываемых атрибутов. Существует несколько способов построения функций принадлежности по результатам экспертного

опроса. Лингвистическая переменная задаётся на предметной шкале посредством определения функций принадлежности для всех термов. Преобразование числового значения некоторой (чаще всего физической) величины в некоторое векторное значение, отражающее отношение эксперта или группы экспертов к величине этого значения в рамках конкретной решаемой задачи. Например, одно и то же значение скорости для разных автомобилей и для разной дорожной обстановки может оказаться и «большим» и «маленьким». Использованные термины «большой» и «маленький» представляют результаты измерений на некоторой не метрической шкале. Эти термины будем называть термами. Совокупность всех термов принятых для нашего «словесного измерения» будем называть терм- множеством или множеством термов.

В работе [Борисов] сформулированы требования к виду функций принадлежности лингвистической переменной, соблюдение которых в значительной степени избавит от некорректности представления переменных. Эти требования универсальны для всех предметных областей. При проектировании редактора функций принадлежности необходимо обеспечить выдачу предупреждений пользователю при нарушении этих требований или создать условия, исключающие такие ошибки.

После того как мы определили, что конкретное значение скорости является «больш**и**м» («**и**» указывает ударение) или «маленьким», возникает естественный вопрос, насколько «большой» или «маленькой» является эта величина.

Если все эксперты утверждают, что значение является «больш**и**м», то будем считать, что некая величина, определяющая соответствие величины терму, будет равна 1. Эту величину можно называть достоверностью. А определить конкретное её значение можно, разделив, например, число экспертов, согласных с утверждением «значение величины «X» является «больш**и**м» для решаемой задачи», на общее число участвующих в опросе экспертов. В данном случае используемую методику можно назвать методом голосования. В противоположном случае если ни один

из экспертов не высказался в пользу того, что значение величины является «больш**и**м», используя уже известную процедуру, получим достоверность высказывания о том, что выбранное значение скорости является «больш**и**м», равна 0. И таким же образом, перебирая последовательно все значения предметной шкалы, получим некоторую функцию, которую назовём функцией принадлежности значений предметной шкалы (шкала значений измеряемой величины) множеству, описанному указанным термом. Предметная шкала, как правило, является осью абсцисс. Отрезок предметной шкалы с ненулевыми значениями функции принадлежности называется областью определения выбранного терма. При построении функций принадлежности лингвистической переменной могут возникнуть некоторые ошибки, связанные с неправильной постановкой вопросов для экспертов, с самой организацией опроса или с обработкой экспертных данных. Эти ошибки в дальнейшем могут привести к неоднозначности или противоречивости получаемых решений, а также невозможности получения таковых.

3.2. Требования к виду функций принадлежности лингвистической переменной

1.Требование к упорядоченности термов означает, что функции принадлежности на предметной шкале должны размещаться в последовательности, в которой понимают эксперты возрастание значения представляемой величины. Например, «очень малое», «малое», «среднее», «большое», «очень большое». Методика проверки выполнения этого требования в автоматическом режиме сложна и в данной работе не приводится.

2.Требование к виду «крайних» функций принадлежности лингвистической переменной или требование к значениям функций принадлежности для граничных точек области определения ЛП.

$$\mu_{t1}(x=0)= \mu_{tmax}(x=x_{max})=1.$$

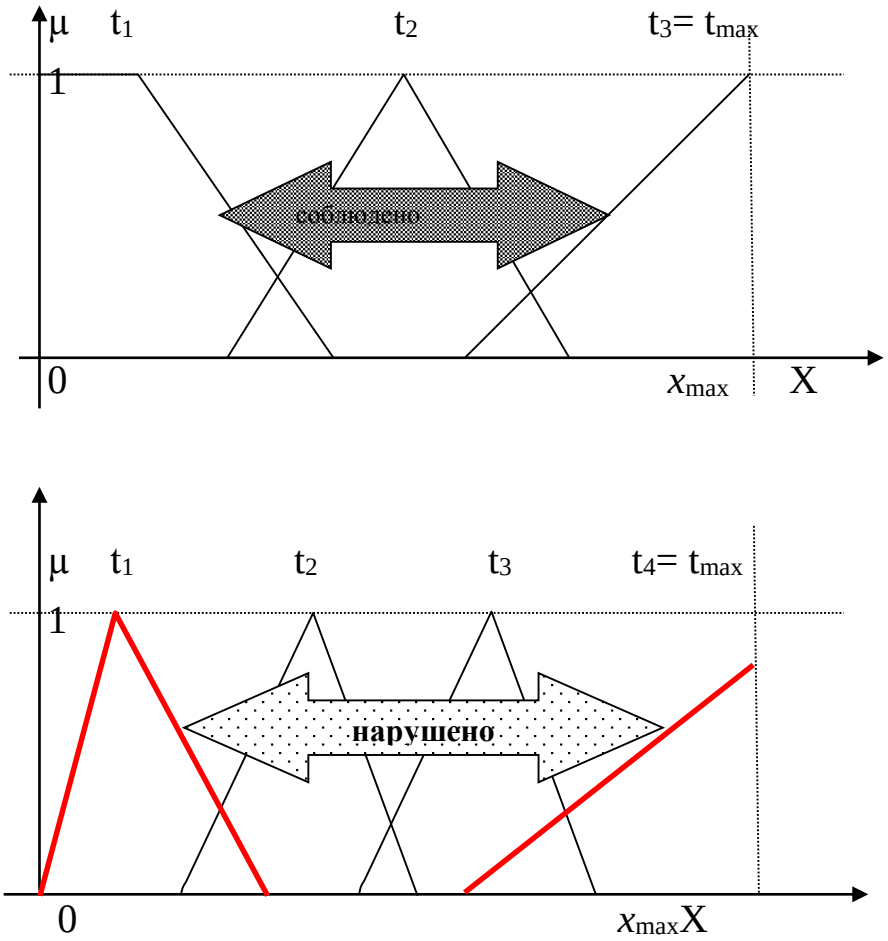


Рис. 7. Вид крайних функций принадлежности

3.Требование к полноте покрытия предметной области областями определения функций принадлежности лингвистической переменной.

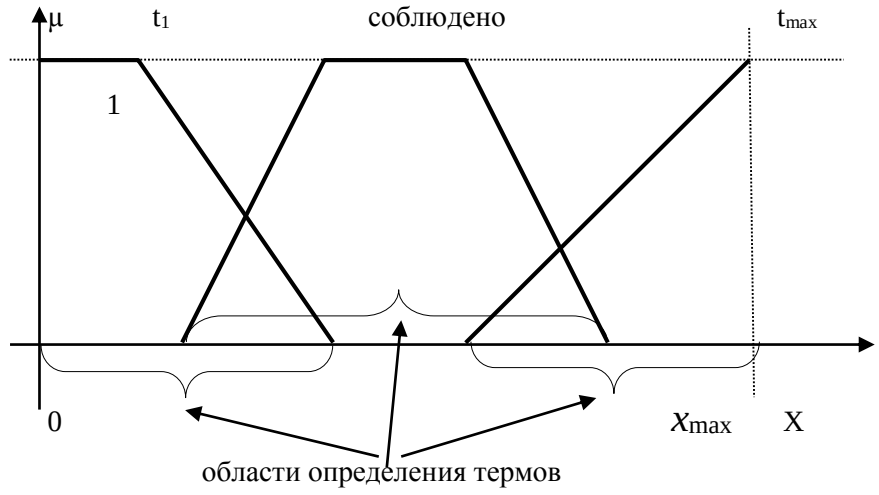


Рис. 8. Полнота покрытия предметной области

Это требование означает, что для любой точки предметной шкалы X должен существовать хотя бы один терм, значение функции принадлежности которого будет отлично от нуля: $\forall x_i \in X, \exists t_j | t_j(x_i) > 0$, под t_j будем понимать не имя терма, а функцию его принадлежности.

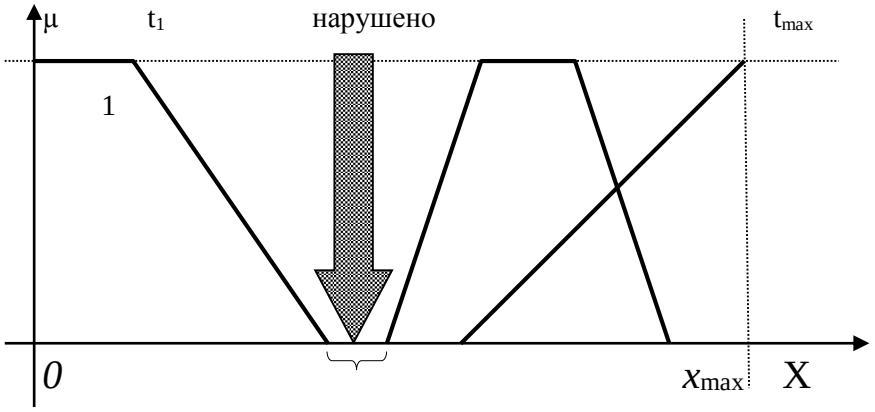


Рис. 9. Нарушение требования полноты

4. Требование к разграничению понятий, описанных функциями принадлежности термов лингвистической переменной означает, что ни какие два терма не должны иметь единичные значения функций принадлежности ни для какого значения предметной шкалы.

$$\forall t_i, t_j \urcorner \exists x_k \in X | t_i(x_k) = t_j(x_k) = 1.$$

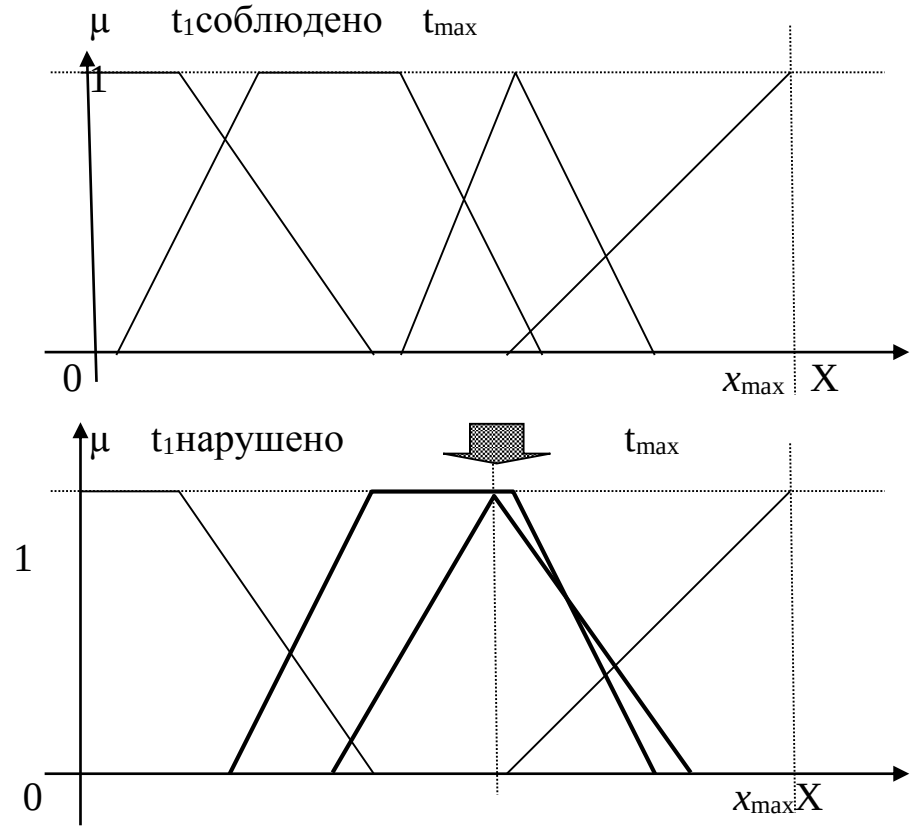


Рис. 10. Разграничение понятий

6. Наличие типового элемента. Для каждого терма должно существовать хотя бы одно значение на предметной шкале со значением достоверности, равным единице.

$$\forall t_i \exists x_k \in X | t_i (x_k) = 1.$$

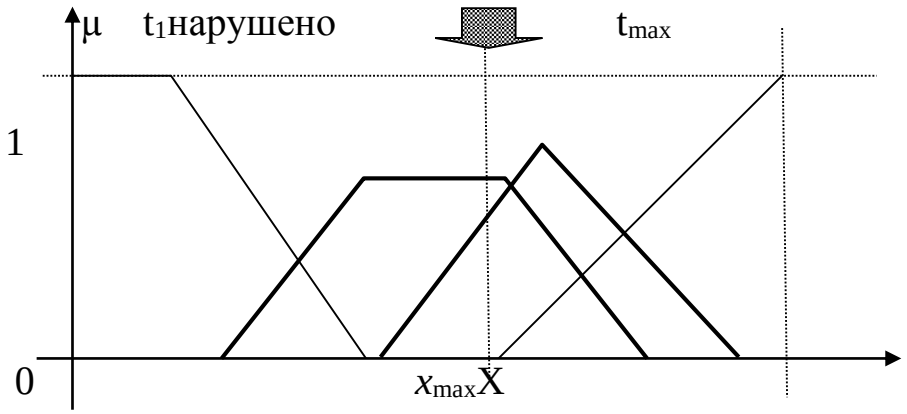


Рис. 11. Наличие типового элемента

7. Ограничение предметной шкалы. Такое требование вводится потому, что многие алгоритмы на лингвистических переменных основаны на вычислении площадей под модифицированными функциями принадлежности. На всех предыдущих рисунках показано ограничение предметной шкалы x_{max}.

3.3. Разработка и исследование редактора функций принадлежности лингвистических переменных для представления экспертных знаний в информационно-аналитических системах

Цель работы: Приобрести навыки представления и использования нечетких атрибутов как компонентов знаний в информационных и управляющих системах.

Задачи: Научиться представлять нечетко определенные атрибуты знаний; разработать алгоритмы создания и редактирования компонентов лингвистических переменных в базах знаний; разрабатывать интерфейсы

пользователей для редактора лингвистических переменных; разработать форматы представления значений лингвистических переменных.

Задание и порядок выполнения работы: Рассмотрение теоретической части задания; проектирование общей структуры редактора; разработка алгоритмов создания и редактирования переменных; разработка алгоритмов анализа выполнения требований к виду функций принадлежности; выполнение чертежей (эскизов) экранных форм; написание и отладка программы редактора; создание тестовых баз знаний с лингвистическими переменными; анализ удобства редактирования и корректности представления переменных в базе знаний; выполнение отчетных материалов.

Форма отчетности. По результатам выполнения лабораторной работы оформляется отчёт, состоящий из титульного листа с указанием города, года, организации и подразделения, в котором выполнена работа, названия работы, фамилии И.О. и учебной группы исполнителя, фамилии И.О. руководителя работы; описания целей и задач работы; описания методов используемых при решении задач; алгоритмов и форматов данных; указанием выбранных языков программирования; скриншотов экранных форм и рисунков форматов данных; описания экспериментов, подтверждающих (демонстрирующих) выполнение поставленных в работе задач; текст моделирующей программы с комментариями.

Защита работы заключается в предоставлении отчета и ответа на вопросы по содержанию отчёта, демонстрации работы редактора, ответа на теоретические вопросы.

Пример реализации пользовательского интерфейса (вариант 1)

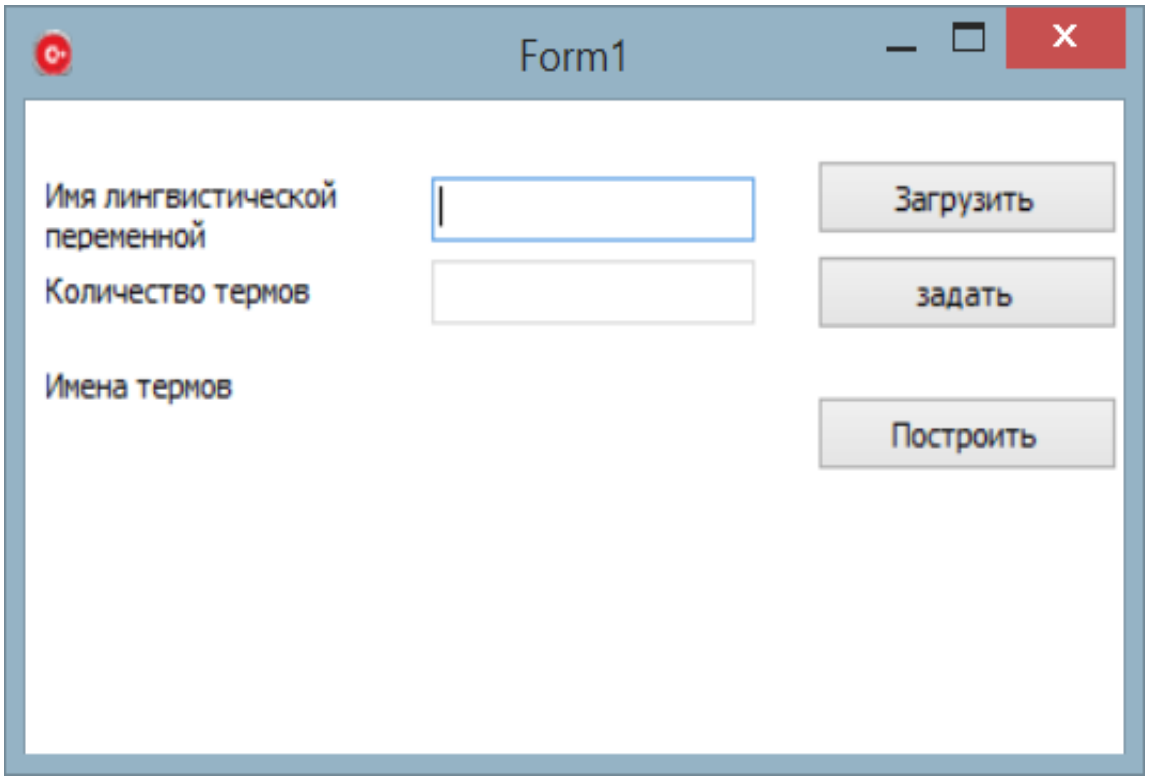


Рис. 12. Первоначальный экран редактора

Производится выбор имени ЛП для загрузки из числа созданных ранее либо ввод нового имени.

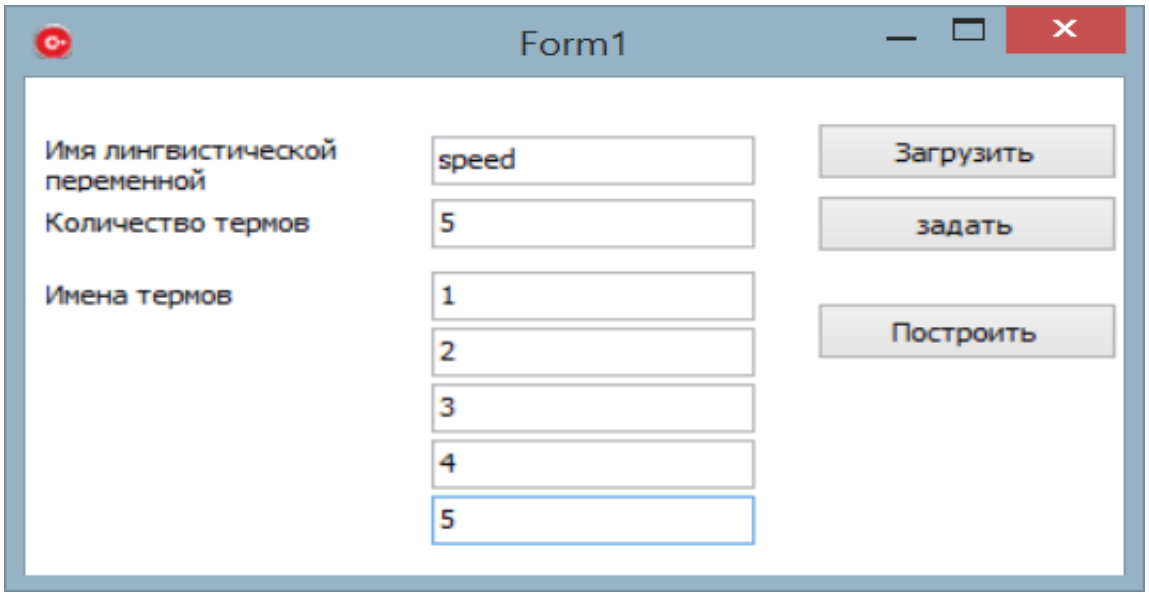


Рис. 13. Перечисление термов

Далее производится указание числа термов и перечисление их имён. В режиме редактирования производится ввод максимального значения предметной шкалы, редактор автоматически строит функции принадлежности трапециевидной формы. В отдельных окнах указаны точки предметной шкалы с «единичными» и «нулевыми» значениями функций принадлежности, которые перед построением можно изменить.

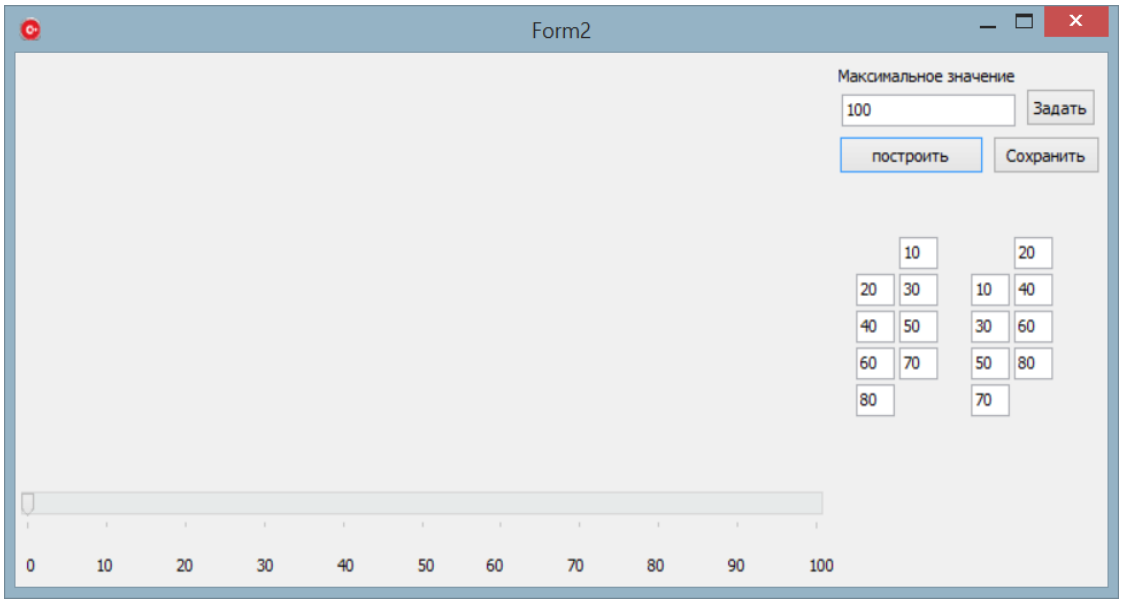


Рис. 14. Ввод граничных точек функций определения и максимального значения предметной шкалы

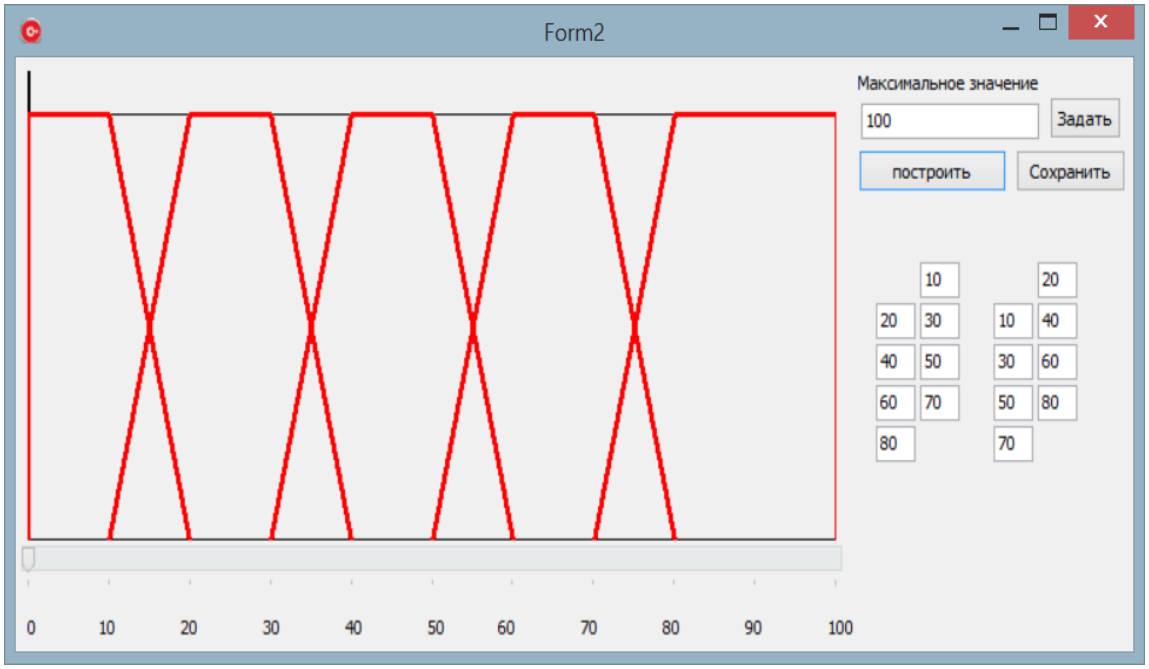


Рис. 15. Предъявление функций принадлежности по введенным значениям

После получения необходимого описания лингвистической переменной можно сохранить полученную редакцию ЛП. Лингвистическая переменная сохраняется в отдельном файле с именем лингвистической переменной.

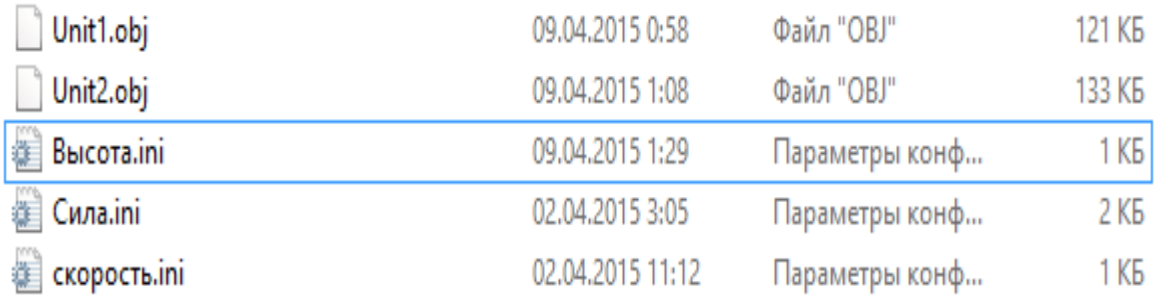


Рис.16. Сохранение значение описывающих ЛП

Пример реализации пользовательского интерфейса (вариант 2)

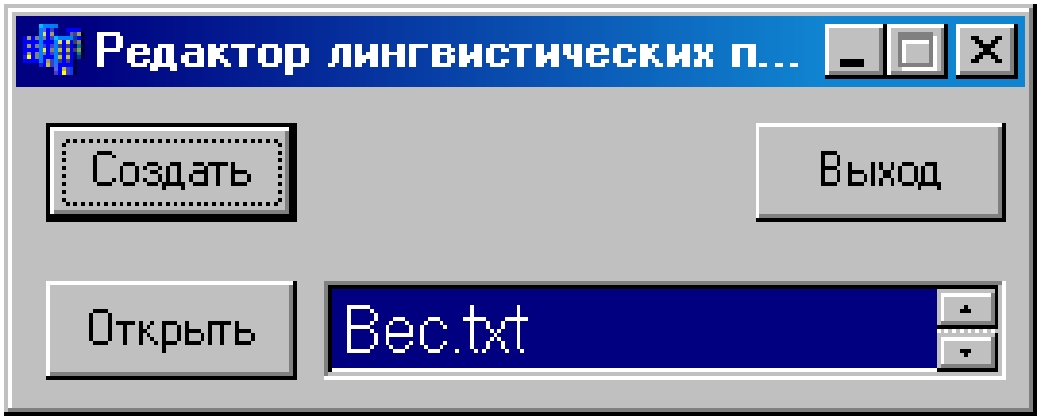


Рис. 17. Создание файла для сохранения значений для ЛП

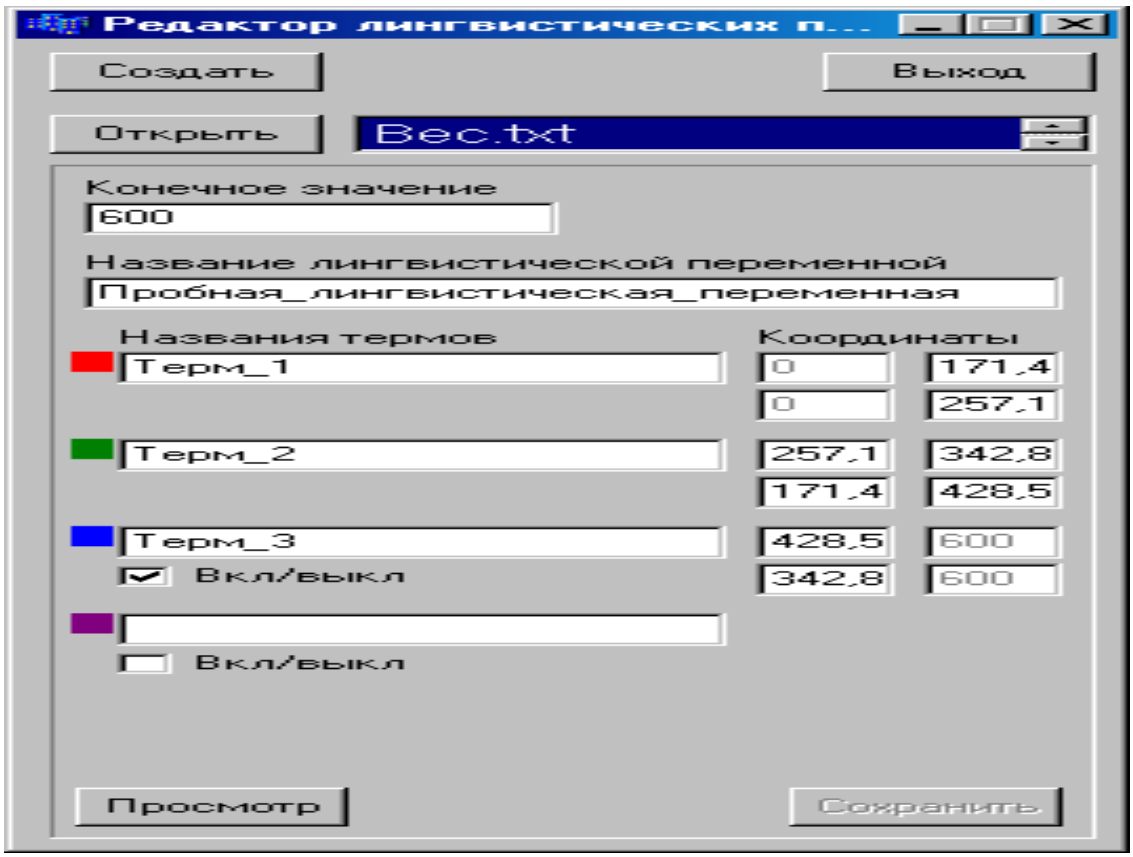


Рис. 18. Создание файла для сохранения значений для ЛП

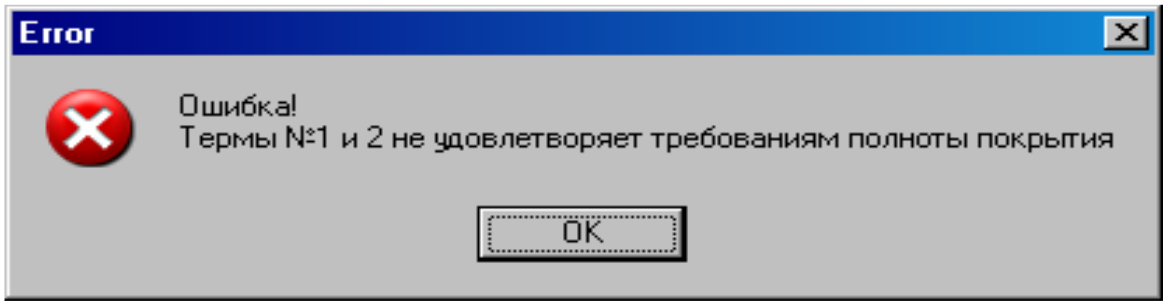


Рис. 19. Сообщение об нарушении требований к виду функций принадлежности ЛП

Глава 4

Представление знаний в виде продукционных правил в информационных- аналитических системах

Элементарные атрибуты, которые мы рассмотрели выше, для придания данным свойств знаний, описанных в гл. 1, должны быть объединены в более сложные структуры. Это могут быть фреймы, семантические сети, продукции и другие структуры.

В настоящей главе мы рассмотрим структурирование знаний в виде наиболее часто используемой в информационно- аналитических системах– продукционной модели.

4.1.Продукционные системы для представления знаний

Продукционные системы (системы с продукционными правилами) считаются частным случаем систем, основанных на правилах. Само правило состоит из двух частей. Разные авторы их определяют как: предпосылка – последствия; посылка – следствие; антецедент – консеквент; антецедента – консеквента; часть «ЕСЛИ» – часть «ТО»; часть «IF» – часть «THEN».«ЕСЛИ» и «ТО» являются служебными словами.

В общем случае продукция имеет следующий вид:

<Идентификатор продукции>

ЕСЛИ (служебное слово)<Имя информационного атрибута 1>

<Связка> (атрибута и значений–равно, больше, меньше, …)

<Значение информационного атрибута 1>

И/ИЛИ (логические операции)

<Имя информационного атрибута 2>

<Связка> (атрибута и значений–равно, больше, меньше, …)

<Значение информационного атрибута 2>

И/ИЛИ

…….

ТО<Имя описательного атрибута или атрибута управления><Связка>

<Значение описательного атрибута>.

После применения правила к имеемым данным могут быть получены новые знания (отсюда названия продукционные правила), либо будет предложено выполнить некоторое действие или получено некоторое описание.

Применение продукции может дать самостоятельный результат (одношаговый вывод) или результат может быть промежуточным и использоваться как данные для другой продукции (многошаговый вывод).

Существуют различные архитектуры продукционных систем. Наиболее известная – с рабочей памятью. В рабочей памяти размещаются факты, управляющие последовательностью применения правил. Применение правила меняет состояние рабочей области. Процесс продолжается пока не будет достигнута некоторая цель, например, будут означены некоторые терминальные атрибуты. Таким образом, продукционная система может состоять из: базы знаний – место хранения правил; рабочей области – область размещения фактов или значений информационных атрибутов; системы логического вывода – система поиска и применения подходящих правил; – системы объяснения логического вывода.

При создании экспертной системы выделяют предметную область, которая представляет собой совокупность технических, программных и прочих средств, посредством которых осуществляется функционирование в конкретной профессиональной области или даже решение отдельных её задач; методические, нормативные, технологические материалы, в которых описан и регламентируется процесс профессиональной деятельности; персонал, непосредственно принимающий участие в профессиональной деятельности, выделяются отдельные специалисты, признанные профессиональным сообществом и владеющие знаниями и навыками решения конкретных профессиональных задач, превосходящими знания и навыки основной массы специалистов предметной области. Необходимо также организационное решение и организационная воля улучшить качество решения профессиональных задач посредством распространения знаний и навыков экспертов через использование такого инстру-

мента, как экспертные системы. Следующим доводом к разработке и использованию экспертных систем является особая требовательность к качеству и надёжности выбора решения. В этом случае для исключения «человеческого фактора» используются знания и навыки эксперта для контроля за лицом, принимающим решения (ЛПР), возможно, за самим экспертом. И ещё одним доводом является агрессивность внешней среды или другая причина невозможности присутствия человека при принятии решения или нецелесообразность (малая доля компонентов, для которых необходимо участие экспертных знаний и навыков или невозможность присутствия человека из-за малой размерности носителя…). Исходя из побудительных причин создания экспертных систем, их можно разделить на автоматизированные системы (с участием человека) и автоматические. Помимо эксперта отношение к системе имеют конечные пользователи, инженеры по знаниям и программисты. Конечные пользователи хотят иметь удобный интерфейс, а также понятные и объяснимые результаты экспертизы. А вот инженеры по знаниям и программисты – те специалисты, которые должны это обеспечить.

Экспертные системы могут иметь различные внутренние структуры, однако, функционально можно выделить редактор знаний, исполняемый модуль (саму экспертную системы) и процессор объяснения результатов.

Редактор знаний позволяет последовательно следить за созданием и заполнением базы знаний экспертной системы, изменять и дополнять структуры и содержимое БЗ при обнаружении неполноты и противоречивости. Редактор знаний должен иметь возможность создания нескольких предметов экспертизы. Каждая такая задача должна иметь свой идентификатор, который указывается при создании проекта. Создание проекта представляет собой формирование файлов для размещения в них элементов представления знаний. После создания проекта все переменные и константы – атрибуты предметной –области должны быть описаны в словаре проекта. Атрибутами, используемыми при описании области

принятия решений, могут быть: параметры описания предметной области (объекта, процесса, системы); параметры описания предлагаемого решения; элементы и структуры логического вывода; промежуточные элементы и структуры, используемые для временного хранения значений, не требующих интерпретации; лингвистические структуры, используемые при объяснении предлагаемого решения; базовые множества для нечетких лингвистических переменных.

Атрибуты могут быть описаны переменными, принимающими числовые, лингвистические, логические, темпоральные (формат даты и формат времени) и строковые значения. Значениями атрибута может быть имена файлов. Это могут быть имена текстовых, графических, видео и других файлов. Лингвистические атрибуты могут представляться лингвистическими нечисловыми и лингвистическими числовыми значениями. Лингвистические нечисловые атрибуты принимают значения из некоторого терм-множества, например {«низкая», «средняя», «высокая»}. Лингвистическая числовая переменная наряду с терм- множеством описывается функциями принадлежности, предметной шкалой и процедурами, относящимися скорее к механизму вывода, чем к описанию самой переменной.

Как правило, конкретная ЭС использует один из способов представления знаний экспертов. Выбор способа представления знаний зависит от многих критериев, например: синтаксической, семантической и семиотической близости экспертным представлениям о предметной области, понятности для экспертов (после пользованием демонстрационным прототипом), необходимости объяснения предлагаемого решения, числом слоёв иерархии выявленных знаний, ресурсов вычислительного устройства (быстродействия, памяти…), наличия поддержки пакетов автоматизации проектирования. Если же есть необходимость в ЭС использовать в малых объёмах дополнительный способ представления к основному выбранному, его можно эмитировать выбранным основным способом. Например, продукцию можно представить при помощи фреймов. Приведём несколько основных систем представления знаний, это: предикатные

системы, семантические сети, продукционные системы, фреймовые сети, системы, основанные на правдоподобных рассуждениях, нечёткие системы. Все эти системы в приложении к поиску приемлемого решения предназначены для сокращения пространства перебора вариантов. Нечеткие системы в «чистом виде» могут использоваться только для описания элементарных атрибутов или оценивания их значений. Для построения советующих систем на их основе необходимо дополнительно использовать продукции, предикаты или фреймы.

4.2. Разработка и исследование редактора правил для продукционной системы

Цель работы: Разработать редактор атрибутов ЭС, редактор правил, редактор объяснения результатов вывода, приобрести навыки представления и использования знаний как компонентов информационных и управляющих системах.

Задачи: Научиться представлять атрибуты знаний; разработать алгоритмы создания и редактирования компонентов баз знаний; разрабатывать интерфейсы пользователей для редактора правил; разработать форматы представления атрибутов и правил в информационных системах.

Задание и порядок выполнения работы: Рассмотрение теоретической части задания; проектирование общей структуры редактора; разработка алгоритмов создания и редактирования атрибутов и привил; выполнение чертежей (эскизов) экранных форм; написание и отладка программы редактора; создание тестовых баз знаний; анализ удобства редактирования и корректности представления продукций в базе знаний; выполнение отчетных материалов.

Форма отчетности. По результатам выполнения лабораторной работы оформляется отчёт, состоящий из титульного листа с указанием города, года, организации и подразделения, в котором выполнена работа, названия работы, фамилия И.О. и учебной группы исполнителя, фамилии

И.О руководителя работы; описания целей и задач работы; описания методов используемых при решении задач; алгоритмов и форматов данных; указанием выбранных языков программирования; скриншотов экранных форм и рисунков форматов данных; описания экспериментов.

Защита работы заключается в предоставлении отчета и ответа на вопросы по содержанию отчёта, демонстрации работы редактора, ответа на теоретические вопросы.

Описание протоколов редактора фактов и знаний

Редактор имеет возможности для создания нескольких предметов экспертизы или для решения нескольких задач в рамках одной экспертизы. Каждая такая задача должна иметь свой идентификатор, который указывается при создании *проекта*. Создание проекта заключается в создании файлов для размещения в них элементов представления знаний.

После создания проекта все переменные и константы – атрибуты БЗ, используемые при создании БЗ, должны быть описаны в словаре проекта. Атрибуты, используемые при описании области принятия решений, могут быть: параметрами описания предметной области (объекта, процесса, системы); параметрами описания предлагаемого решения; элементами и структурами логического вывода; промежуточными элементами и структурами, используемыми для временного хранения значений, не требующими интерпретации; лингвистическими структурами, используемыми при объяснении предлагаемого решения; базовое множество для нечетких множеств.

Пункт меню редактора «Проект». При создании конкретного модуля экспертизы и выборе пункта «Проект» необходимо указать в поле «Название проекта» необходимо указать идентификатор проекта. При дальнейшем входе в БЗ (базу знаний) из редактора или из процедур получения экспертизы будут доступны только элементарные атрибуты и составные элементы знаний (например, продукции) созданные для этого проекта. Допускается существование иерархической структуры проектов

с возможностью доступа к компонентам БЗ проектом низших уровней. Проект, стоящий по иерархии ниже рассматриваемого будем называть субпроектом, выше – суперпроектом. Для создания субпроекта необходимо войти в проект с его именем и далее открыть пункт меню «субпроект», указать его имя и работать с компонентами БЗ.

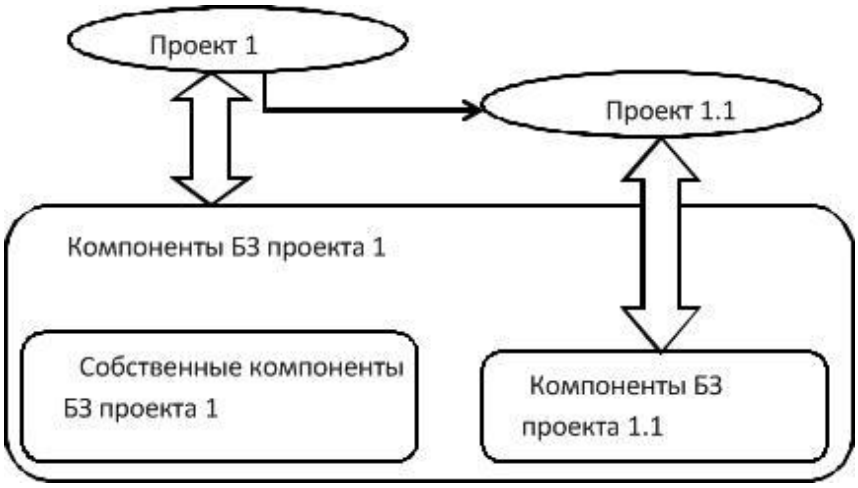


Рис. 20. Компонентный состав проекта

При работе с компонентами БЗ проекта любого уровня иерархии должна иметься (имеется) возможность «Добавить», «Удалить», «Копировать», «Вставить» из другого проекта или «Переименовать» компоненту БЗ. Помимо смыслового идентификатора (поле «Название проекта») всех групп компонентов (проект) проекту автоматически присваивается числовой идентификатор. Элементарным компонентам БЗ (атрибут, продукция) также присваивается числовой идентификатор. При дальнейшей работе редактора или процедур получения экспертизы будет использоваться только числовой идентификатор, а смысловой идентификатор, указанный пользователем, ему соответствующий будет использоваться при объяснении полученных результатов. При переименовании проекта или компоненты числовой идентификатор не меняется.

Пункт меню редактора «Атрибут». Все атрибуты, используемыми при описании области принятия решений (параметры описания предметной области, параметры предлагаемого решения) должны быть заданы в поле «Название атрибута» пользователь указывает смысловой идентификатор атрибута, имеется возможность «Добавить», «Удалить», «Копировать», «Вставить» из другого проекта или «Переименовать» атрибут БЗ. После указания имени атрибута необходимо указать тип значения выбрав из пунктов меню: «Числовое», «Символьное», «Нечёткое», «Логическое», «Дата». Для «нечёткого» значения нужно выбрать из следующих подтипов: «Нечёткое число», «Нечёткая переменная», «Лингвистическая переменная».

«Нечёткое число» представляется двумя полями – самим числом и его достоверностью, «Нечёткая переменная» потребует описания «Предметной шкалы» и «Функции принадлежности». Предметная шкала описывается указанием имени числовой переменной, размерностью и дискретностью шкалы. Для лингвистической переменной дополнительно нужно описать «Терм-множество». При использовании нечётких и лингвистических переменных нужно предоставить возможность воспользоваться встроенным редактором функций принадлежности. Редактор «Функции принадлежности» автоматически строит функции треугольные и трапециевидные функции принадлежности для каждого терма из терм- множества. Далее пользователь редактора (инженер по знаниям или эксперт) изменяет их форму и расположение на предметной шкале используя активные точки представленных функций принадлежности.

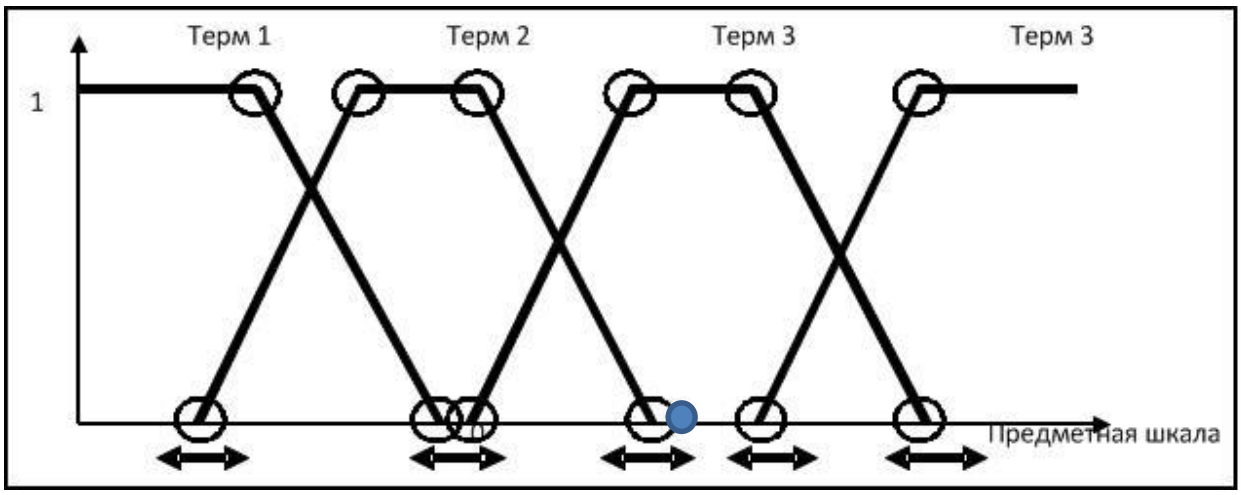


Рис. 21. Вид редактора функции принадлежности

Редактирование функций принадлежности производится после выбора терма. Выбирается активная точка функции (рис. 21) и перемещается по горизонтали до придания функции требуемого (экспертом) вида. Изменения формы контролируется редактором по следующим критериям: требование к упорядоченности элементов; наличие типового элемента (2); разграничение понятий (3) и полноте покрытия предметной шкалы (4); требования к виду крайних функций принадлежности (5); ограничение области определения. На рис. 22 представлены иллюстрации нарушений и соблюдения этих требований при редактировании функций принадлежности термов лингвистических переменных.

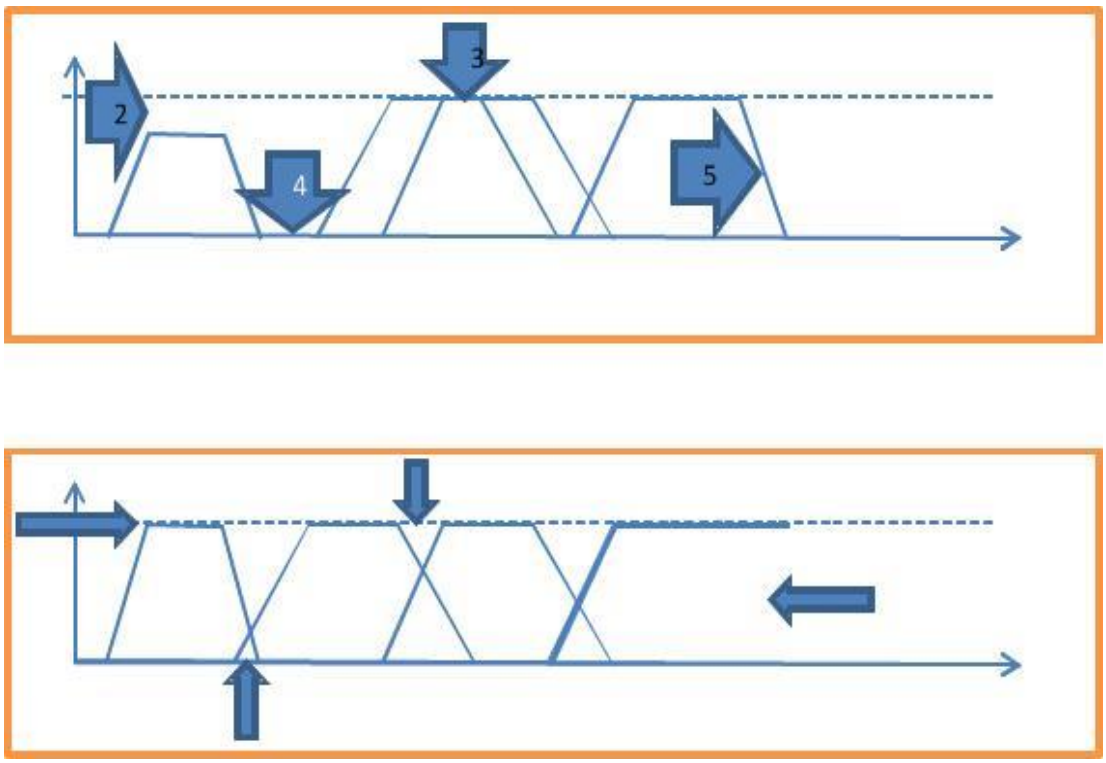


Рис. 22. Редактирование функций принадлежности

Пункт меню редактора «Продукционные правила»

Основной целью редактора является наполнение БЗ продукционными правилами, в которых представлены знания экспертов о предмете экспертизы. В нашем редакторе правила составляются из описанных атрибутов и их значений в пункте редактора «атрибут». Пример экранной формы с одним продукционным правилом приведен на рис. 23.

После служебных слов «ЕСЛИ» и «ТО», которые присутствуют только на экранных формах, следуют атрибуты антецедента и консеквента и их значения. Пользователь выбирает из предложенного ему списка атрибутов и из всего множества значений атрибута. Различные атрибуты и их значения в антецеденте разделены связкой «И». Консеквент же имеет только одно значение одного атрибута.

Рис. 23. Экранная форма с продукционным правилом

Примерные алгоритмы, описание работы редактора и форматы файлов БЗ приведены в прил. 3.

Пример экранной формы редактора

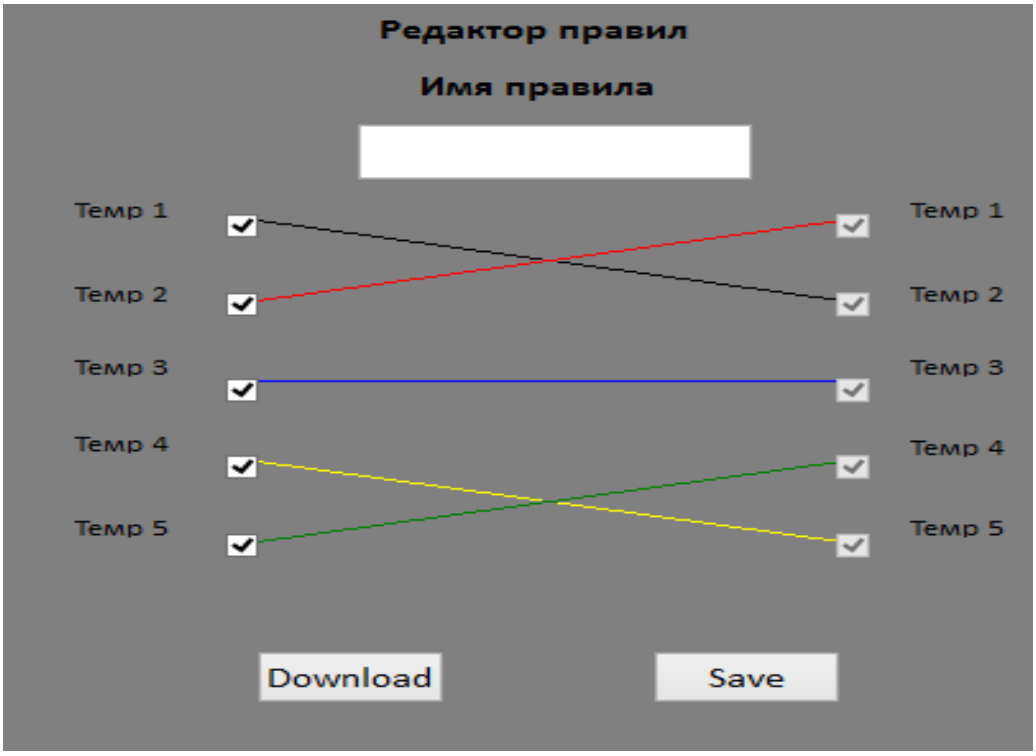


Рис. 24. Редактор правил в виде графа соответствия атрибутов

Глава 5
Построение прикладных систем с использованием лингвистических переменных

В предыдущих главах мы научились описывать значения атрибутов с помощь лингвистических переменных, научились строить продукционные правила для решения прикладных задач с использованием экспертных знаний. В этой главе мы построим нечеткий регулятор входные и выходные значения описаны лингвистическими переменными, а соответствие между ними заданы правилами.

5.1. Работа продукционной системы с лингвистическими переменными

Задача системы, которую мы будем строить, – оценки риска деятельности некоторого предприятия. Имена лингвистических переменных и имена термов будем специально выделять заглавными буквами и заключать в кавычки даже в случае их совместного использования. Такие группы имён будут напоминать конструкции естественного языка, а специальное выделение должно их отделить от поясняющего текста пособия. Используем атрибуты «надёжность системы безопасности жизнедеятельности» назовём «НАДЁЖНОСТЬ» или «ВХОДНОЕ значение 1» и «последствия от выброса вредных примесей» атрибут назовём «ВРЕД» или «ВХОДНОЕ значение 2». Для описания этих атрибутов введём одноимённые лингвистические переменные «НАДЁЖНОСТЬ» и «ВРЕД».

Атрибут «НАДЁЖНОСТЬ» определён на предметной шкале значений [0, max1=100]. Размерность переменной можно представить в процентах или баллах.

Терм-множество лингвистической переменной «НАДЁЖНОСТЬ» после экспертного опроса выглядит следующим образом T1={«НИЗКАЯ», «СРЕДНЯЯ», «ВЫСОКАЯ»}. Каждый терм (термин)

представим функцией принадлежности некоторого конкретного значения x_i предметной шкалы [0, 100] нечеткому множеству «НАДЁЖНОСТЬ» «НИЗКАЯ», «НАДЁЖНОСТЬ» «СРЕДНЯЯ», «НАДЁЖНОСТЬ» «ВЫСОКАЯ».

Оценка степени соответствия значений предметной шкалы термам производится группой экспертов.

Атрибут «ВРЕД» определим на предметной шкале значений [0,max2=100]. Размерность переменной или единицы измерения переменной определим в процентах или баллах.

Терм-множество лингвистической переменной «ВРЕД» T2={«УМЕРЕННЫЙ», «ВЫСОКИЙ», «ОЧЕНЬ ВЫСОКИЙ»}. Каждый терм (термин) представим функцией принадлежности некоторого конкретного значения y_j предметной шкалы [0, 100] нечетким множествам, которые именуются лингвистической переменной и добавлением имени одного из термов:
«ВРЕД»«УМЕРЕННЫЙ», «ВРЕД»«ВЫСОКИЙ», «ВРЕД»«ОЧЕНЬ ВЫСОКИЙ».

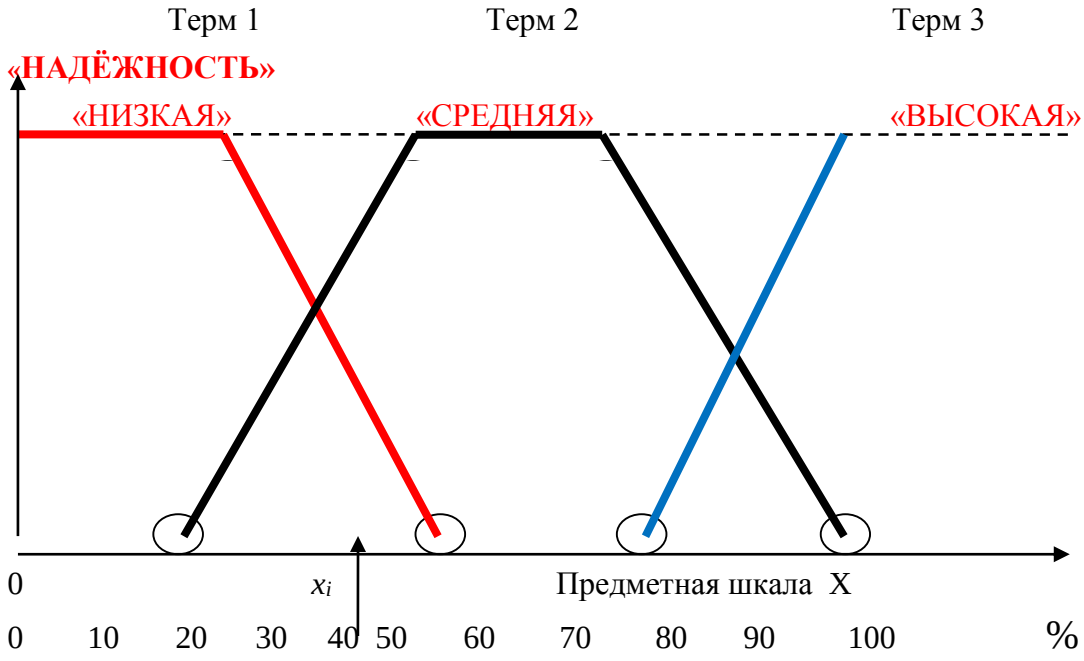


Рис. 25. Терм-множество лингвистической переменной «НАДЁЖНОСТЬ»

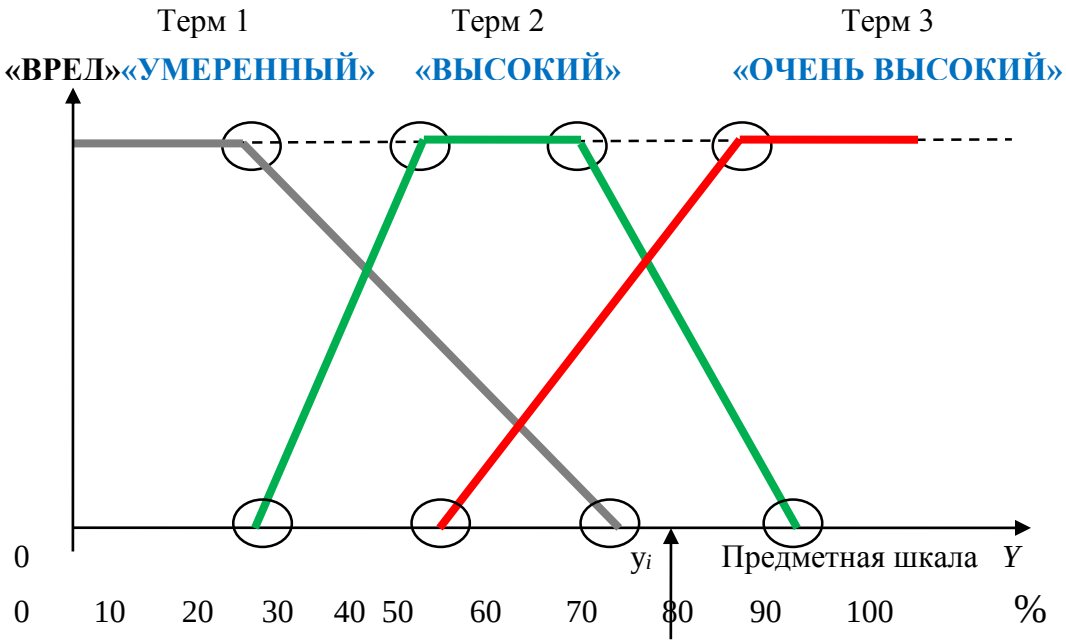


Рис. 26.Терм-множество лингвистической переменной «ВРЕД»

Умеренный риск соответствует наличию вероятности легких повреждений для здоровья. Высокий риск соответствует высокой вероятности нанесения вреда здоровью и даже некоторой (низкой) вероятности смертельных исходов. Очень высокий риск соответствует высокой вероятности смертельных исходов на предприятии и прилегающей территории.

Атрибут «РИСК» определим на предметной шкале значений [0,100]%.

Терм-множество лингвистической переменной «РИСК» Т3={«УМЕРЕННЫЙ», «ВЫСОКИЙ», «ОЧЕНЬ ВЫСОКИЙ»}. Каждый терм (термин) представим функцией принадлежности некоторого конкретного значения предметной шкалы z_k [0,100] нечетким множествам «РИСК»«УМЕРЕННЫЙ», «РИСК»«ВЫСОКИЙ», «РИСК»«ОЧЕНЬ ВЫСОКИЙ».

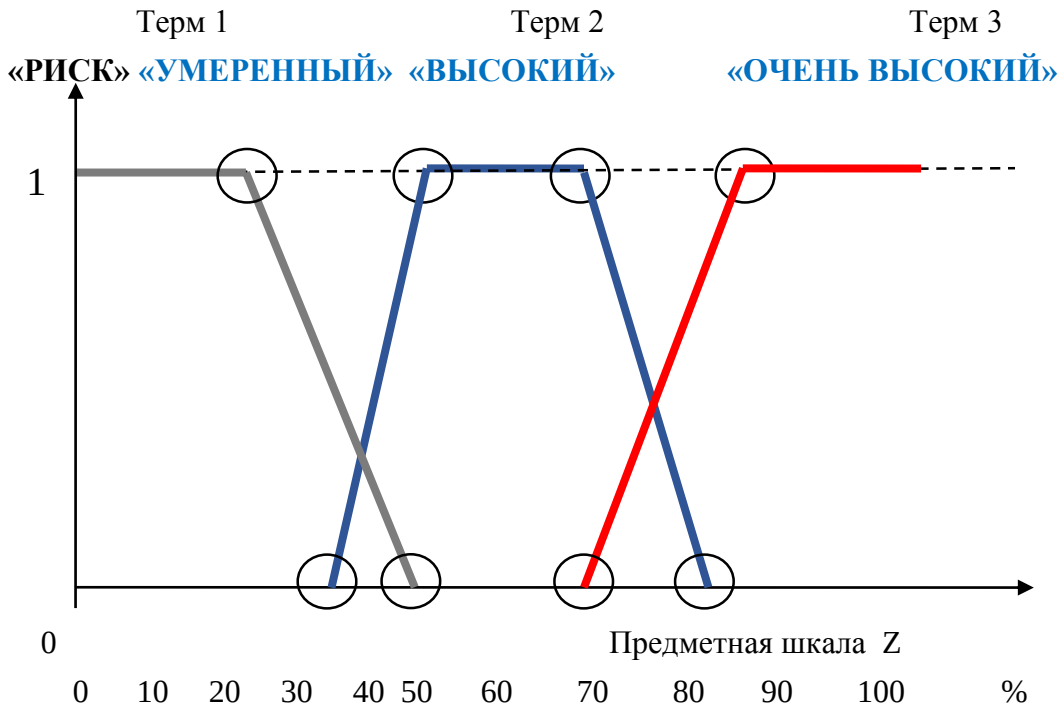


Рис. 27.Терм-множество лингвистической переменной «РИСК»

Функции принадлежности можно строить с использованием описанного выше редактора многими способами. Например, собираем несколько экспертов и, перебирая последовательно значения предметной шкалы с некоторым шагом, задаём экспертам такой вопрос: «Кто считает, что значение «m1» соответствует термину «ВХОДНОЕ значение 1» «НИЗКОЕ»? Нормируем значение числа проголосовавших «n1» (делим на общее число экспертов N) и отмечаем в прямоугольной системе координат «достоверность \ предметная шкала» точку (m1, n1: N). Повторяя эту процедуру для всех значений предметной шкалы с выбранным шагом и для всех термов мы получим все функции принадлежности лингвистической переменной «ВХОДНОЕ значение 1».

Такие же операции проведем для лингвистических переменных «ВХОДНОЕ значение 2» и «ВЫХОДНОЕ значение».

Для того чтобы описать связь входных и выходного значения, используем продукции или правила и описанный выше редактор правил.

Лингвистическая переменная «НАДЁЖНОСТЬ» определена на терм- множестве Т1={«НИЗКАЯ», «СРЕДНЯЯ», «ВЫСОКАЯ»}= t11», «t12», «t13»}.

Лингвистическая переменная «ВРЕД» определена на терм- множестве

Т2={«УМЕРЕННЫЙ», «ВЫСОКИЙ», «ОЧЕНЬ ВЫСОКИЙ»}={«t21», «t22», «t23»}.

Лингвистическая переменная «РИСК» определена на терм- множестве Т3={«УМЕРЕННЫЙ», «ВЫСОКИЙ», «ОЧЕНЬ ВЫСОКИЙ»} ={«t31», «t32», «t33»}.

Множество продукций представим в виде графа на рис. 28.

Перечислим ниже полученные продукции с их идентификаторами Р1 - Р9.

Р1

ЕСЛИ «НАДЁЖНОСТЬ»=«НИЗКАЯ» **И** «ВРЕД»=«УМЕРЕННЫЙ»,
ТО «РИСК»=«ВЫСОКИЙ».

Р2

ЕСЛИ «НАДЁЖНОСТЬ»=«СРЕДНЯЯ» **И** «ВРЕД»=«УМЕРЕННЫЙ»,
ТО «РИСК»=«УМЕРЕННЫЙ».

Р3

ЕСЛИ «НАДЁЖНОСТЬ»=«ВЫСОКАЯ» **И** «ВРЕД»=«УМЕРЕННЫЙ»,
ТО «РИСК»=«УМЕРЕННЫЙ».

Р4

ЕСЛИ «НАДЁЖНОСТЬ»=НИЗКАЯ» **И** «ВРЕД»=«ВЫСОКИЙ»,
ТО «РИСК»=«ОЧЕНЬ ВЫСОКИЙ».

Р5

ЕСЛИ «НАДЁЖНОСТЬ»=«СРЕДНЯЯ» **И** «ВРЕД»=«ВЫСОКИЙ»,
ТО «РИСК»=«ВЫСОКИЙ».

P6
ЕСЛИ «НАДЁЖНОСТЬ»=«ВЫСОКАЯ» **И** «ВРЕД»=«ВЫСОКИЙ»,
ТО «РИСК»=«ВЫСОКИЙ».

P7
ЕСЛИ «НАДЁЖНОСТЬ»=«НИЗКАЯ» **И** «ВРЕД»=«ОЧЕНЬ ВЫСОКИЙ»,
ТО «РИСК»=«ОЧЕНЬ ВЫСОКИЙ».

P8
ЕСЛИ «НАДЁЖНОСТЬ»=«СРЕДНЯЯ»**И** «ВРЕД»=«ОЧЕНЬ ВЫСОКИЙ»,
ТО «РИСК»=«ОЧЕНЬ ВЫСОКИЙ».

P9
ЕСЛИ «НАДЁЖНОСТЬ»=«ВЫСОКАЯ» **И** «ВРЕД»=«ОЧЕНЬ ВЫСОКИЙ»,
ТО «РИСК»=«ВЫСОКИЙ».

Теперь предположим, что пользователь советующей системы намеревается получить оценку степени риска воздействия операционной деятельности предприятия на жизнедеятельность сотрудников предприятия и жителей территории вблизи завода.

Предположим, что «Надёжность системы безопасности жизнедеятельности» («НАДЁЖНОСТЬ») оценивается в 55 %.

Оценка «Последствий от выброса вредных примесей» («ВРЕД») составляет 66 баллов. После процедуры фаззификации, скрытой от пользователя, получим следующие значения лингвистических переменных:

«НАДЁЖНОСТЬ»(55) ={<0/«НИЗКАЯ»>, <1/«СРЕДНЯЯ»>, <0/«ВЫСОКАЯ»>};

«ВРЕД»(66)={<0/«УМЕРЕННЫЙ»>, <1/«ВЫСОКИЙ»>, <0,7«ОЧЕНЬ ВЫСОКИЙ»>}.

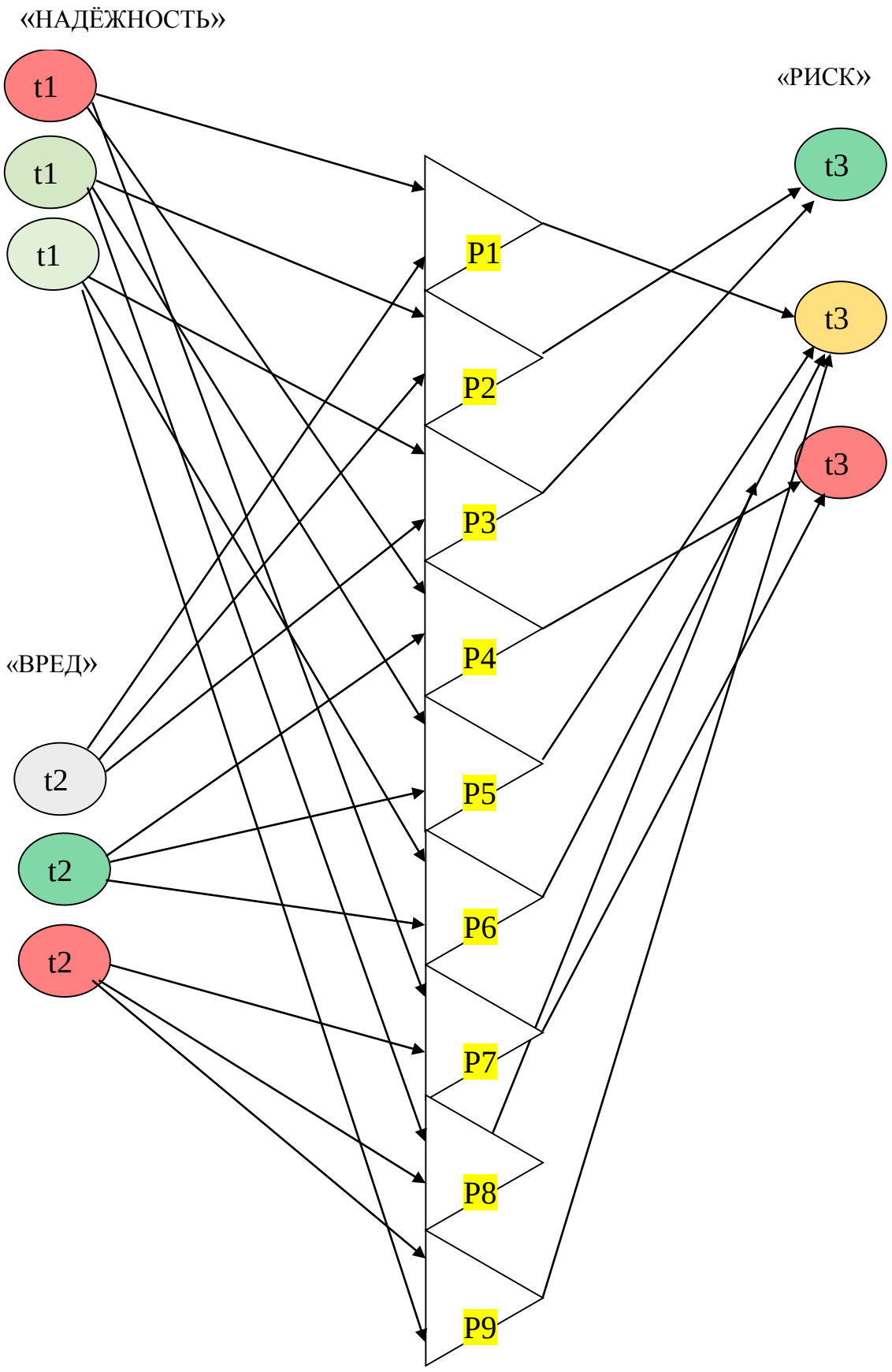


Рис. 24. Граф соответствия терм-множеств

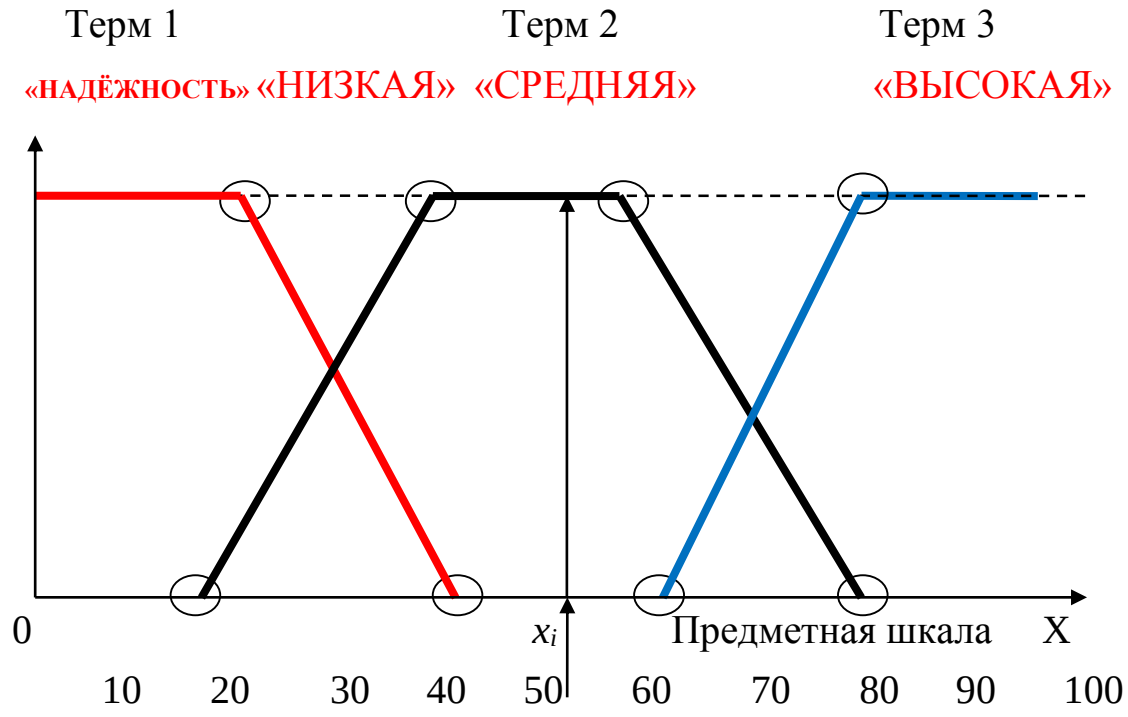


Рис. 25. Оценка значения 55 % на терм-множестве лингвистической переменной «НАДЕЖНОСТЬ»

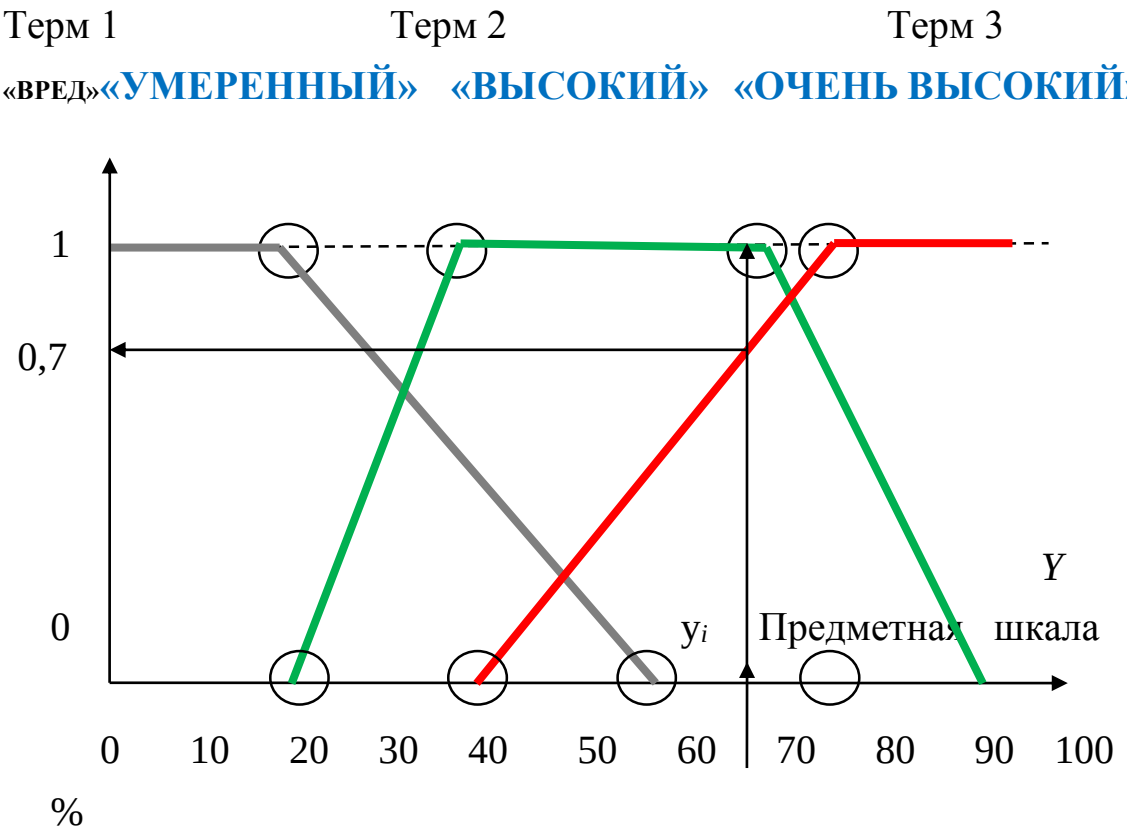


Рис. 26. Оценка значения 66 % терм-множества лингвистической переменной «ВРЕД»

После этого поисковая системы выбрала подходящие для применения правила, т.е. те, в которых встречаются термы с ненулевыми значениями функций принадлежности: Р5; Р8 (рис. 27).

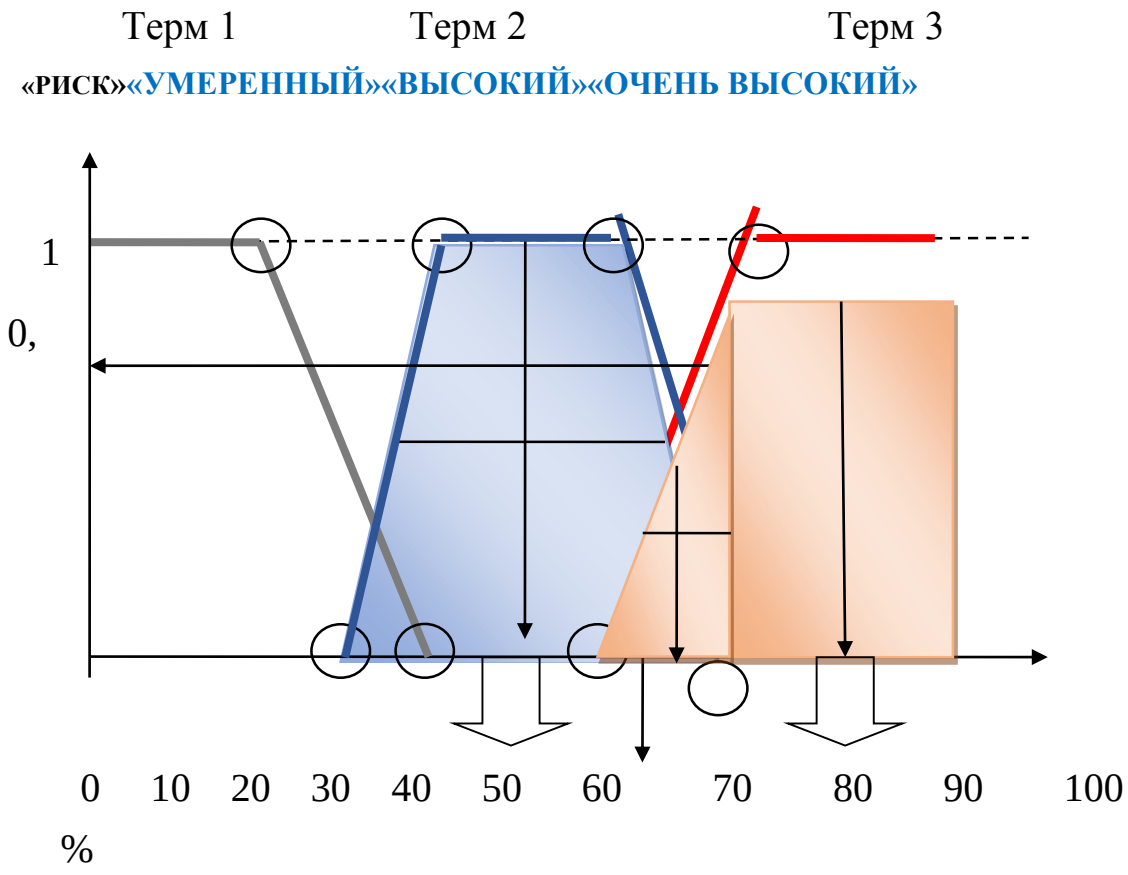


Рис. 27. Прогноз степени «РИСКа» 68% на терм-множестве лингвистической переменной «РИСК»

Таким образом, можно использовать следующие правила с полученными значениями достоверности

Р5

ЕСЛИ «НАДЁЖНОСТЬ»=<1/«СРЕДНЯЯ»>**И** «ВРЕД»=<1/«ВЫСОКИЙ»>,

ТО «РИСК»=«ВЫСОКИЙ».

Р8

ЕСЛИ «НАДЁЖНОСТЬ»=<1/«СРЕДНЯЯ»>**И** «ВРЕД»=<0,7/«ОЧЕНЬ ВЫСОКИЙ»>,

ТО «РИСК»=«ОЧЕНЬ ВЫСОКИЙ».

Далее, применив правила Р5; Р8 и произведя процедуру дефаззификации получим прогноз степени «РИСКа» 68 %.

Подробнее варианты алгоритмов фаззификации, дефаззификации, поиска подходящих правил, формат файлов БД и промежуточных файлов и взаимодействия с пользователем описаны в приложении.

5.2. Разработка и исследование нечеткого регулятора

Цель работы: Объединить в одном проекте редактор правил, редактор лингвистических переменных и нечеткий регулятор.

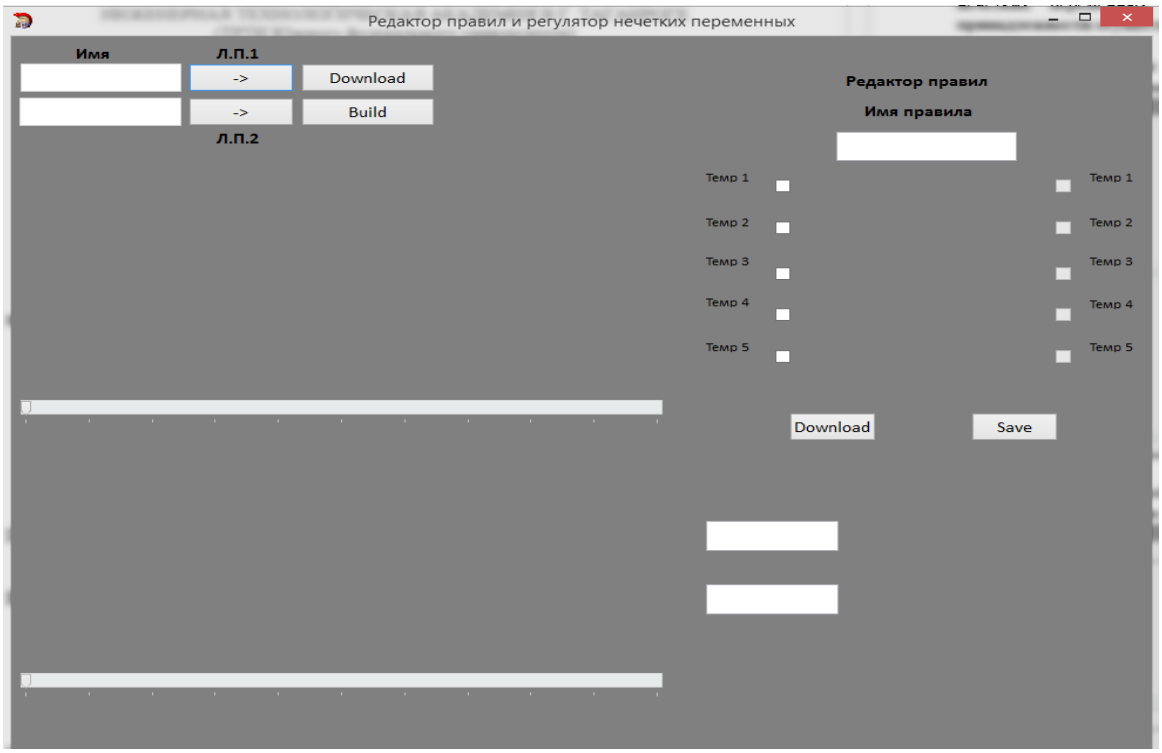
Задачи: Научиться представлять определенные атрибуты знаний; разработать алгоритмы создания и редактирования нечётких регуляторов; разрабатывать интерфейсы пользователей нечеткого регулятора, разработать форматы представления нечетких атрибутов и правил в информационных системах.

Задание и порядок выполнения работы: Рассмотрение теоретической части задания; проектирование общей структуры нечеткого регулятора; написание алгоритмов создания и редактирования нечетких атрибутов и привил; выполнение чертежей (эскизов) экранных форм; написание и отладка программы нечеткого регулятора; создание и исследование тестовых примеров для нечеткого регулятора; анализ результатов экспериментов; выполнение отчетных материалов.

Форма отчетности. По результатам выполнения лабораторной работы оформляется отчёт, состоящий из титульного листа с указанием города, года, организации и подразделения, в котором выполнена работа, названия работы, фамилии И.О. и учебной группы исполнителя, фамилии И.О. руководителя работы; описания целей и задач работы; описания методов используемых при решении задач; алгоритмов и форматов

данных; указанием выбранных языков программирования; скриншотов экранных форм и рисунков форматов данных; описания экспериментов.

Пример реализации экранных форм нечеткого регулятора



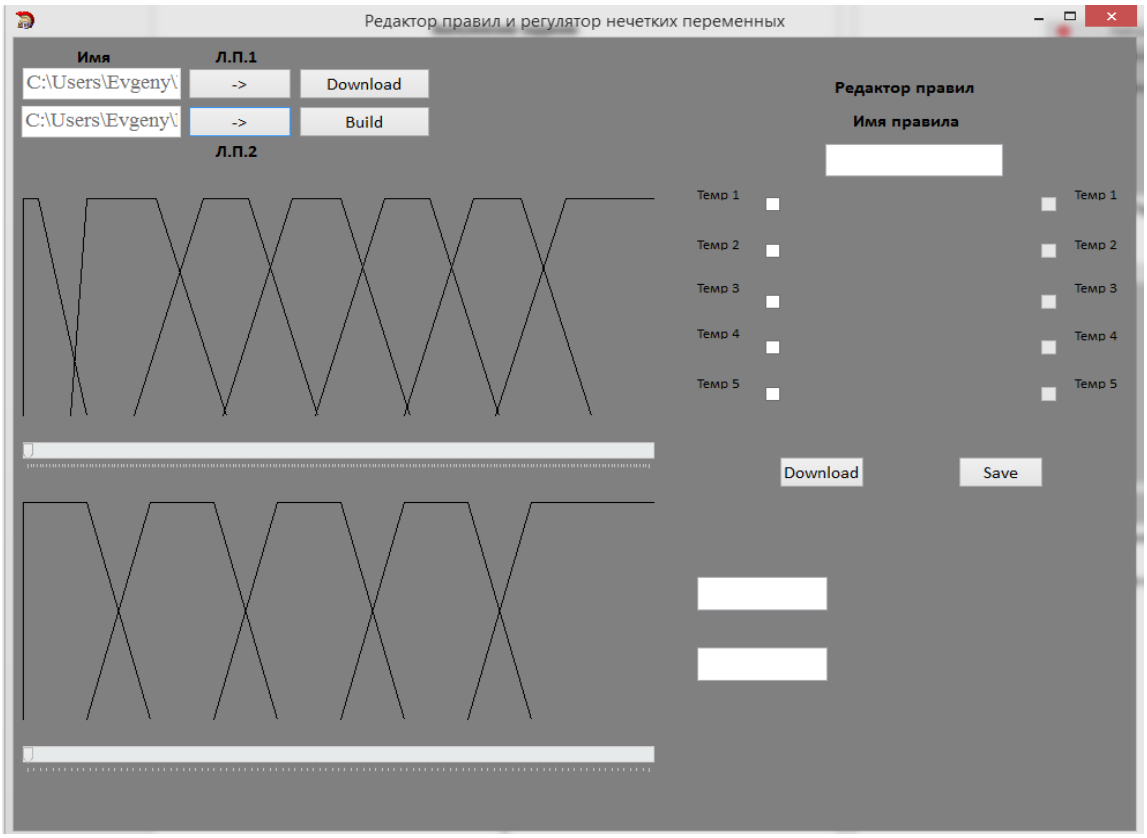
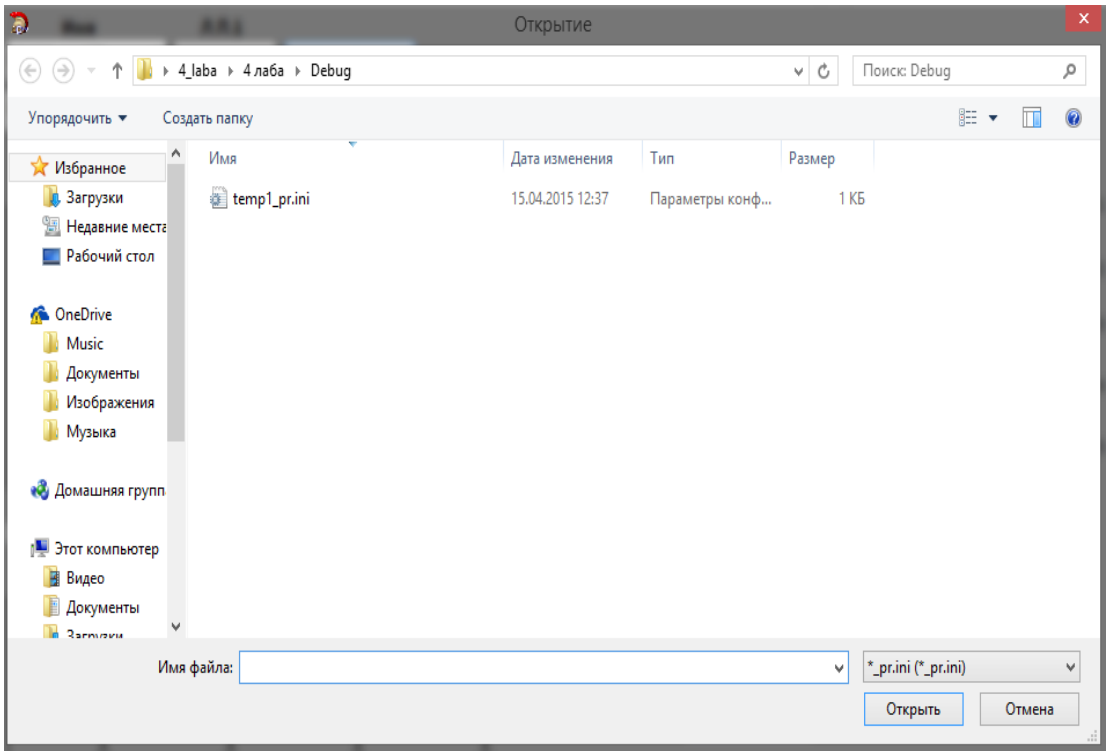


Рис. 28. Примеры экранных форм



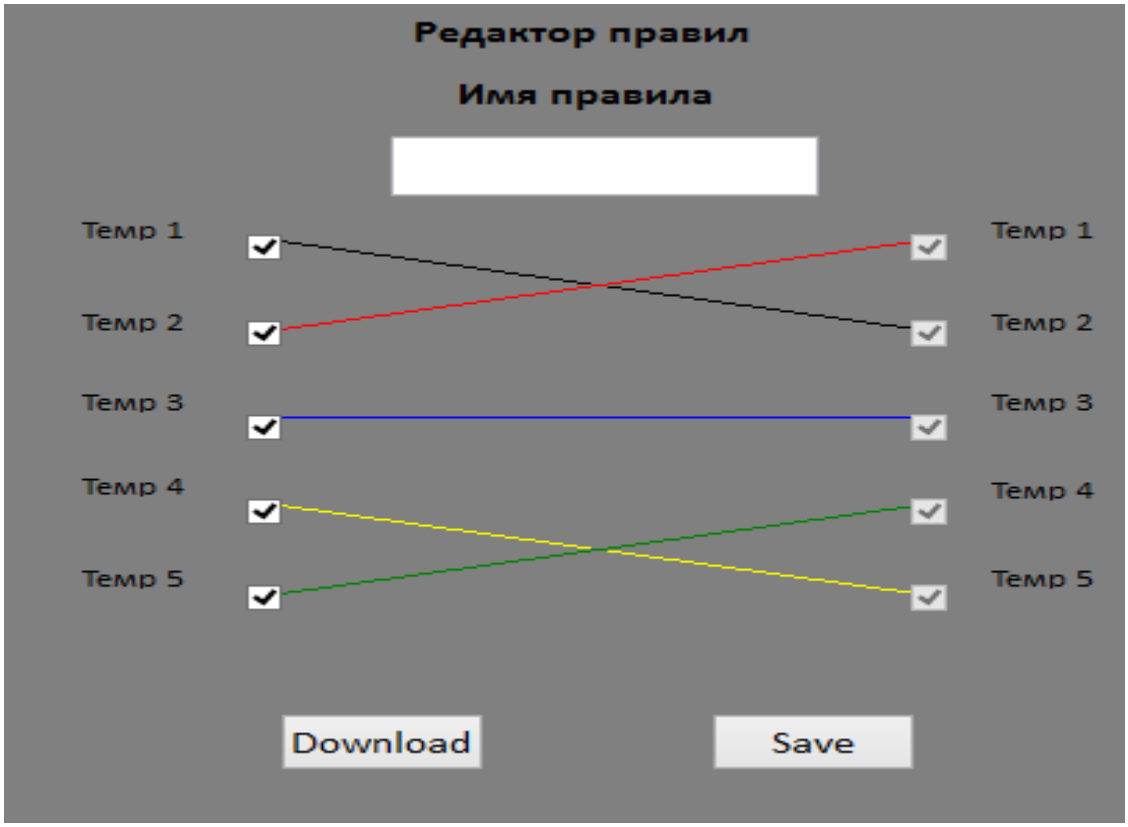
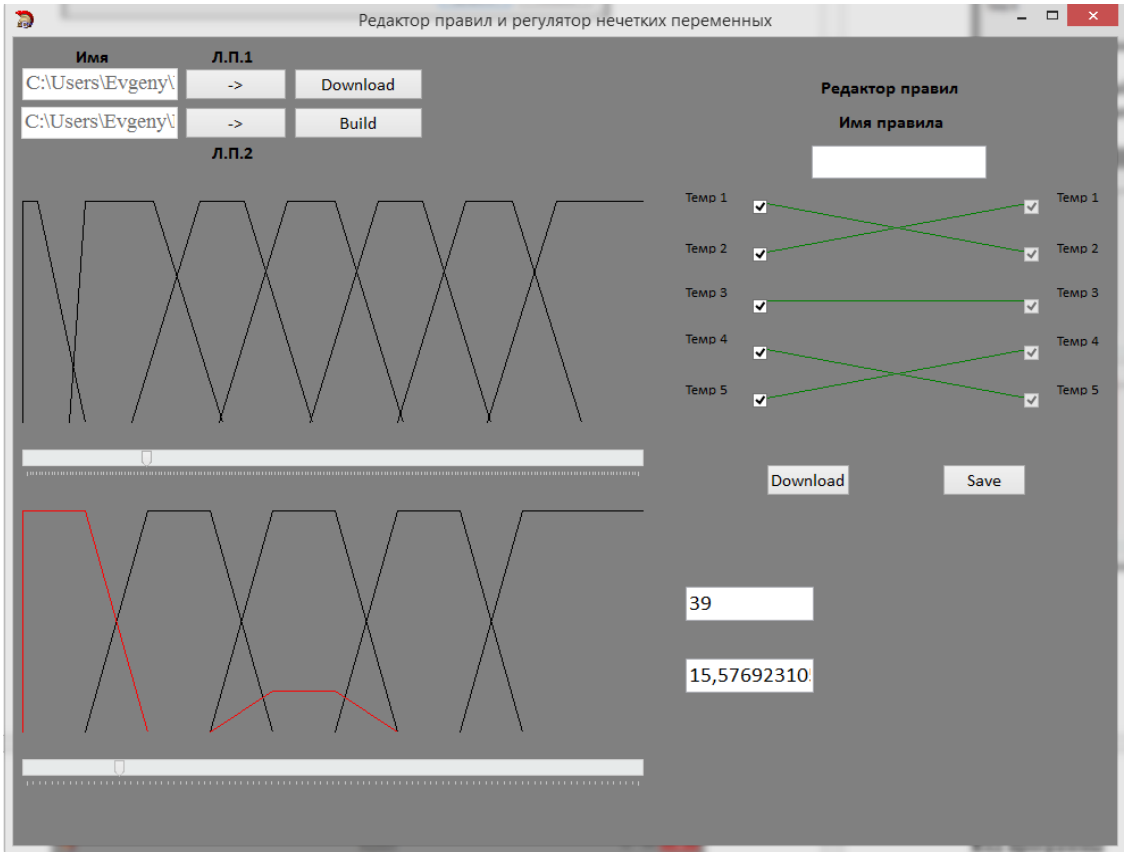


Рис. 28. Продолжение.



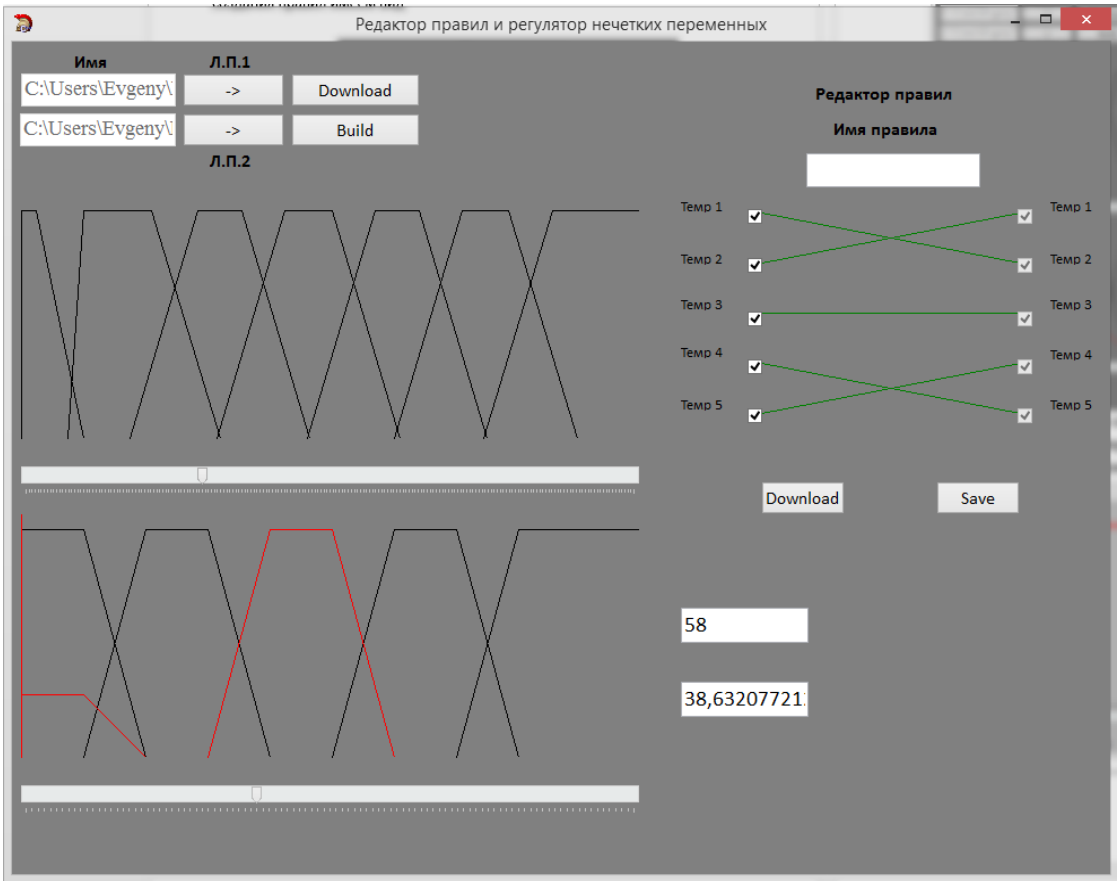


Рис. 89. Окончание.

Приложения

Приложение 1

Пример структуры файлов БЗ советующей системы

Файл ATRid (идентификатор проекта) включает в себя 4 структурных элемента (рис.П1).

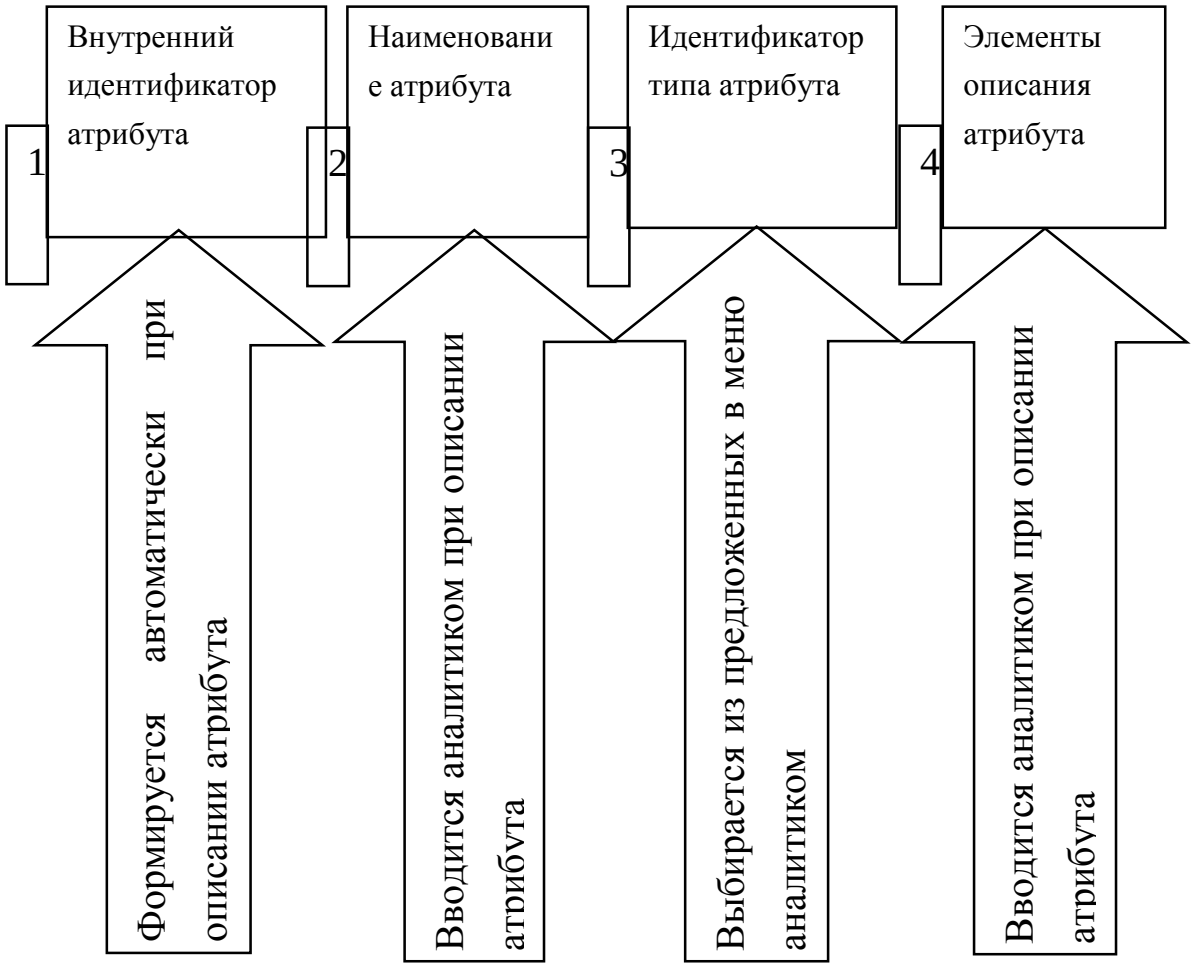


Рис. П.1. Файл ATRid (идентификатор проекта)

Внутренний идентификатор атрибута формируется автоматически при описании атрибута. Нумерация уникальна в рамках одного проекта. В разных проектах могут встречаться одинаковые идентификаторы. Существует возможность создания общей базы атрибутов для нескольких

или для всех проектов (экспертиз). В этом случае добавляется поле «доступность», со значениями «для всех проектом» (ALL) или ссылкой на файл доступности (AccessFile).

Идентификатор типа атрибута выбирается из меню, которое открывается при переходе в соответствующее поле. Возможен выбор из следующих значений: «Числовое»; «Символьное»; «Нечёткое»; «Логическое»; «Дата». Для рассматриваемой предметной области предлагается ограничить число типов значений.

Файл ATRset (идентификатор проекта) формируется следующим образом (рис. П2).

Файлы ATRset… могут создаваться для каждого типа атрибутов. Мы будем использовать единый файл для всех атрибутов, учитывая возможности его модификации при введении ограничений на число типов атрибутов.

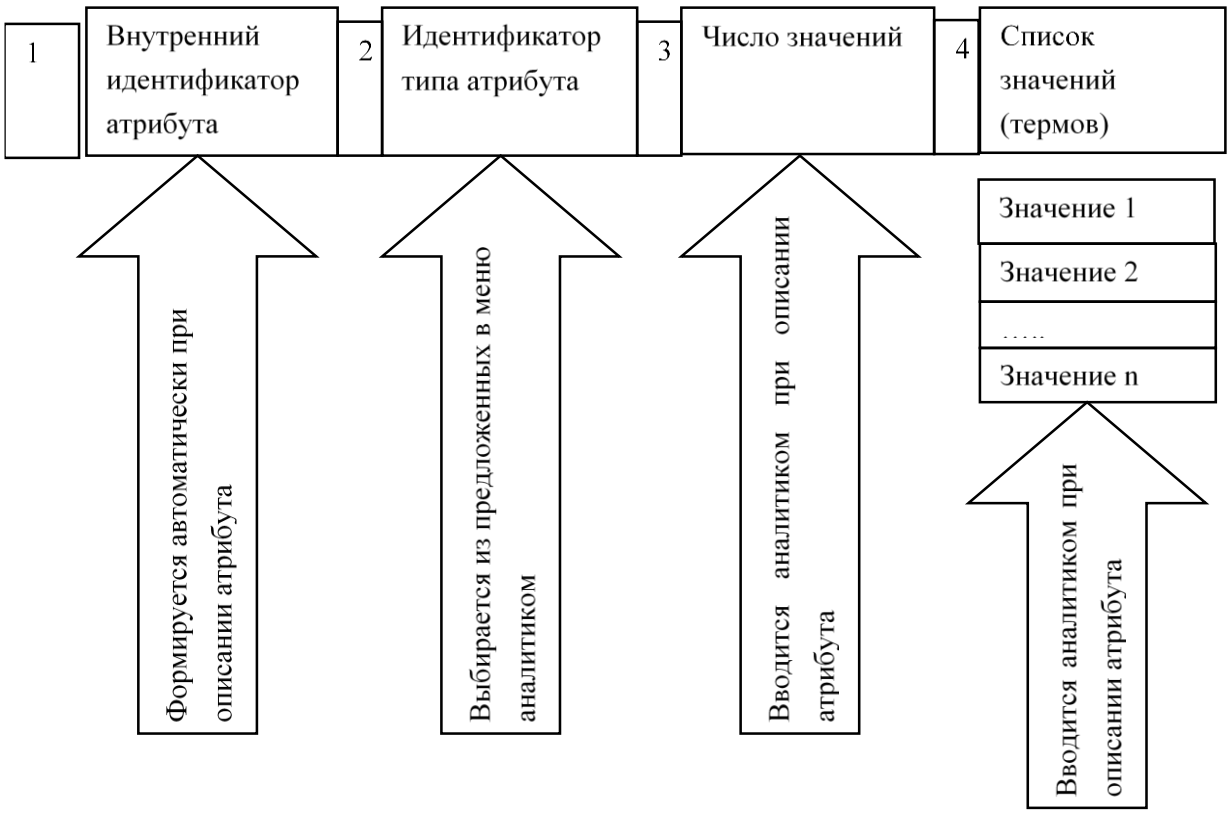


Рис. П2. Файл ATRset (идентификатор проекта)

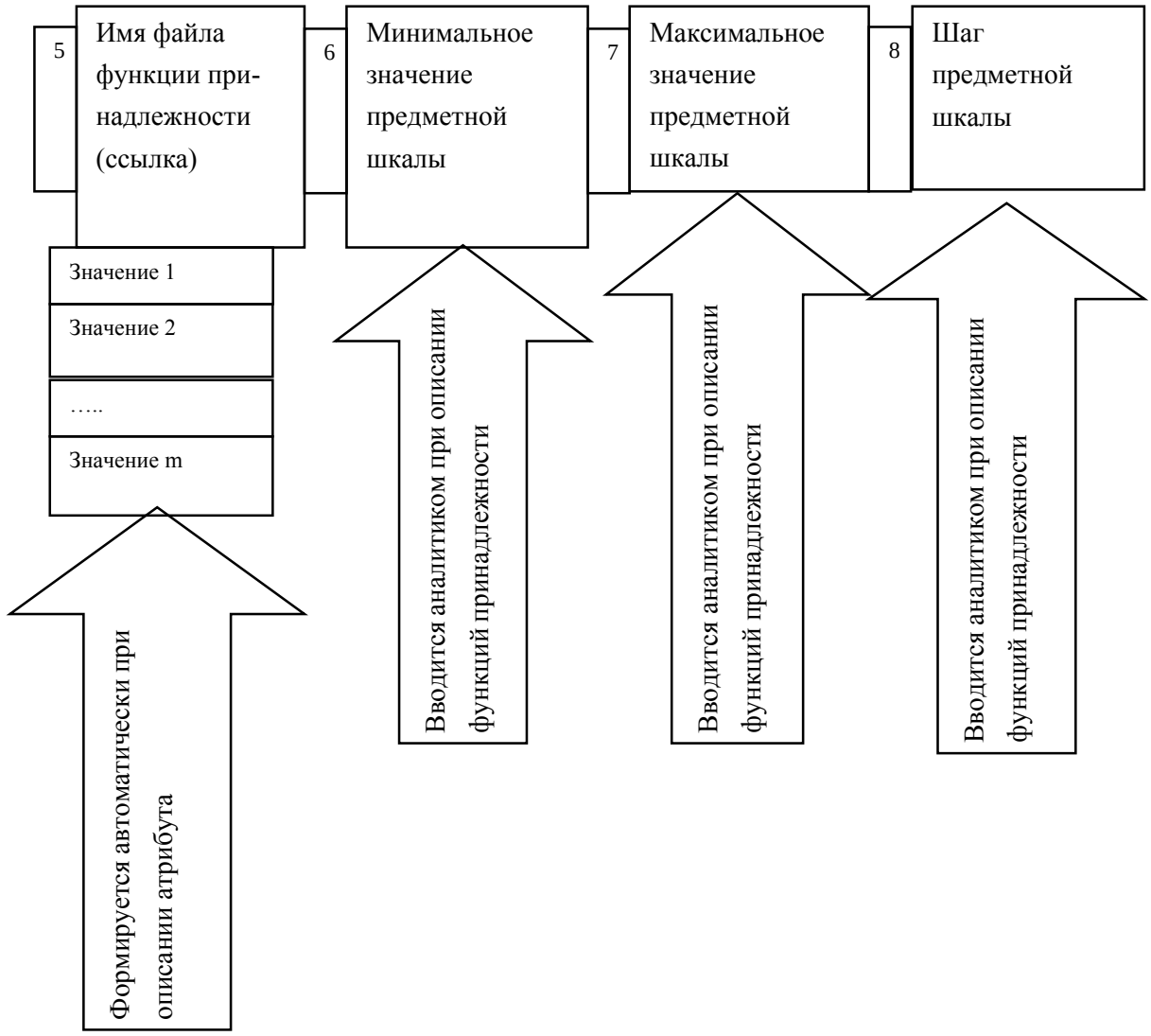


Рис. П2. Окончание.

Поля «Список значений» и «Имя файла функции принадлежности» представляют повторяющиеся поля, т.е. поля, которые могут иметь несколько значений. Значение поля «Имя файла функции принадлежности» формируется автоматически добавлением к идентификатору атрибута номера значения атрибута. Одновременно создаётся файл с таким именем. Файл заполняется редактором «Функции принадлежности». В файлах «Функции принадлежности» могут храниться функции в виде диапазонов и коэффициентов кусочно- линейных функций принадлежности в виде, представленном на рис. П3.

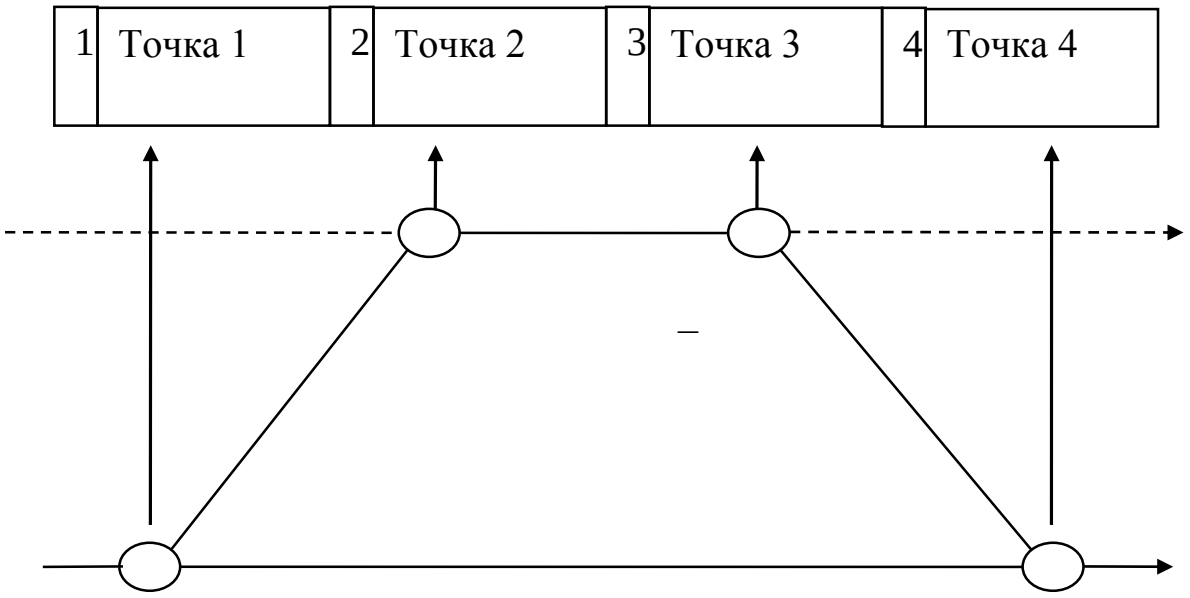


Рис. П3. Файл FAN (идентификатор проекта) (идентификатор атрибута) (номер терма)

Должна существовать возможность ввода вида функций принадлежности в виде формул, на диапазонах трёх диапазонах значений, а также множества точек (рис.П4).

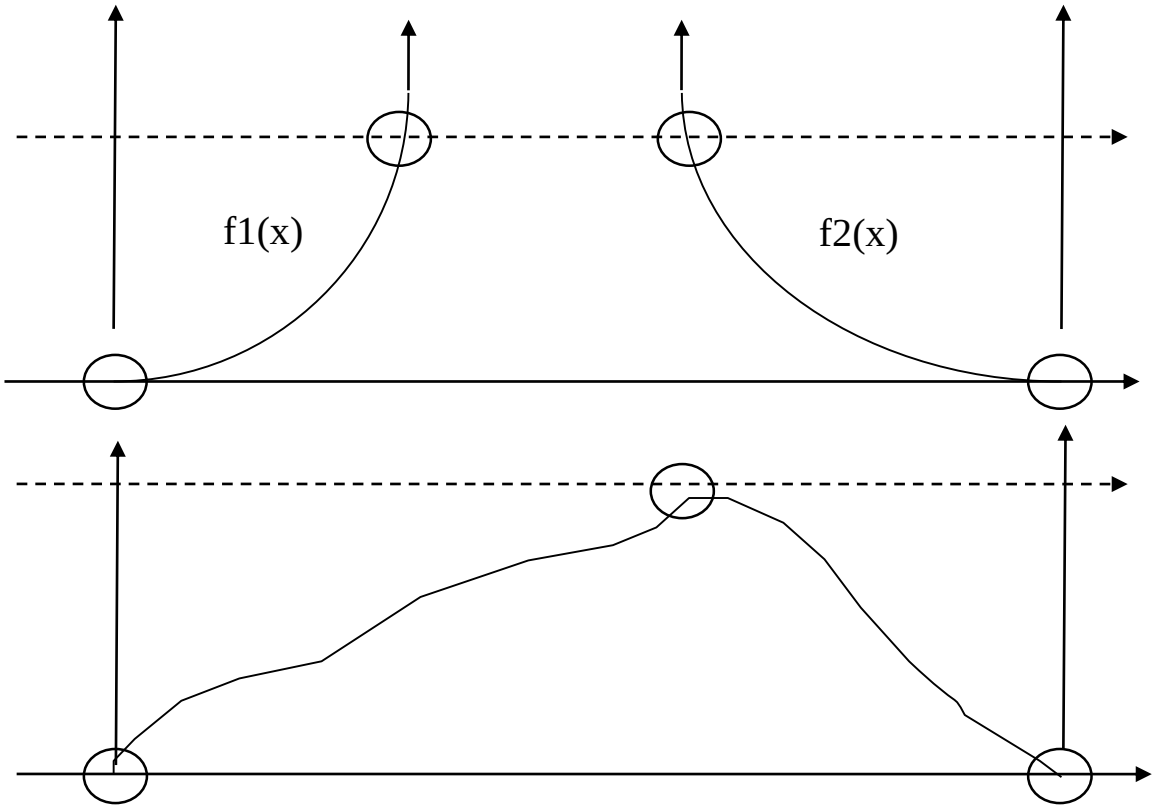


Рис. П4. Ввод вида функций принадлежности

Общий вид файла RULTid (идентификатор проекта) приведен на рис. П5.

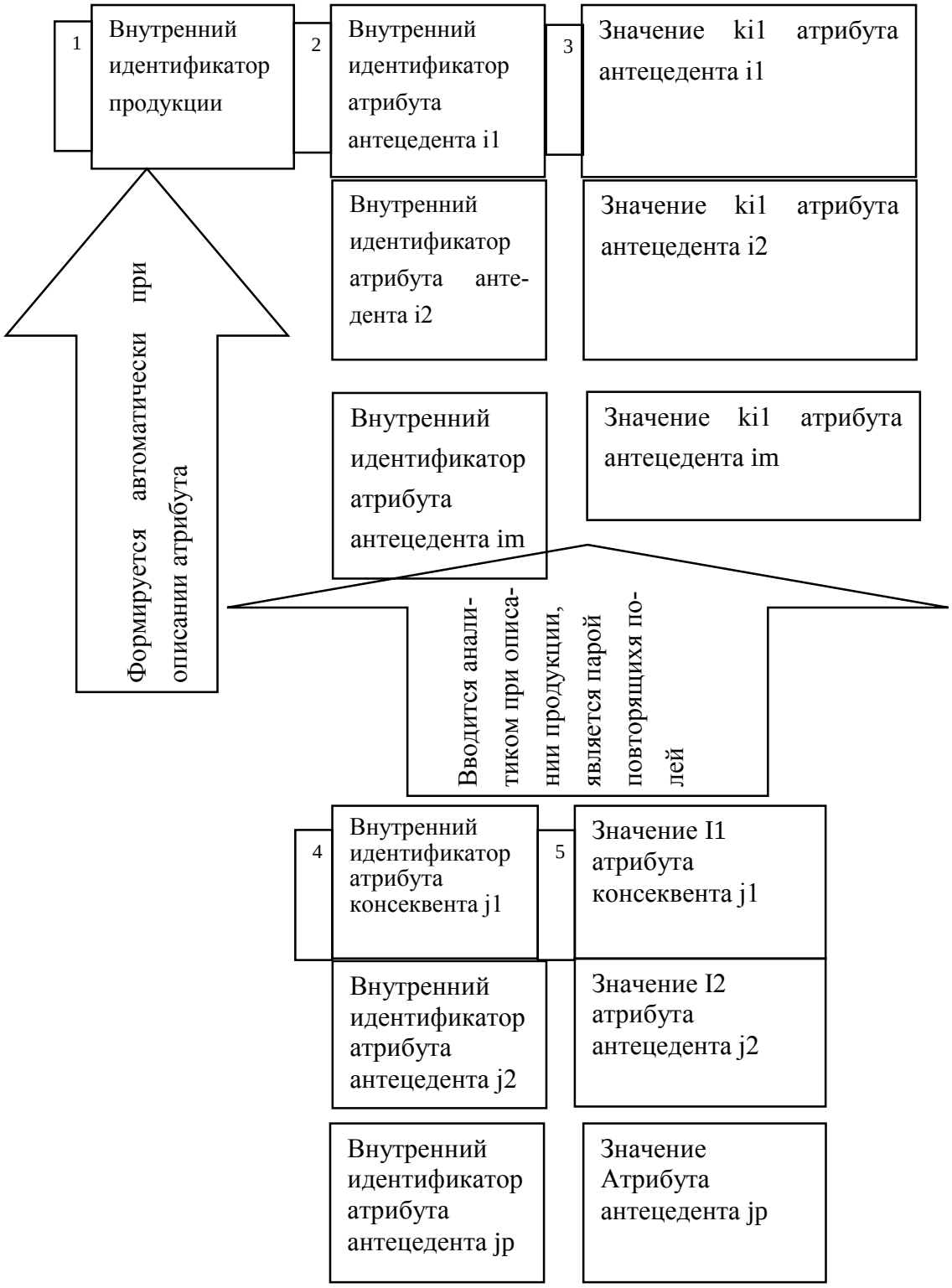


Рис. П5. Файл RULTid (идентификатор проекта)

Общая схема взаимосвязи между файлами БЗ советующей системы принимает вид, изображенный на рис. П6.

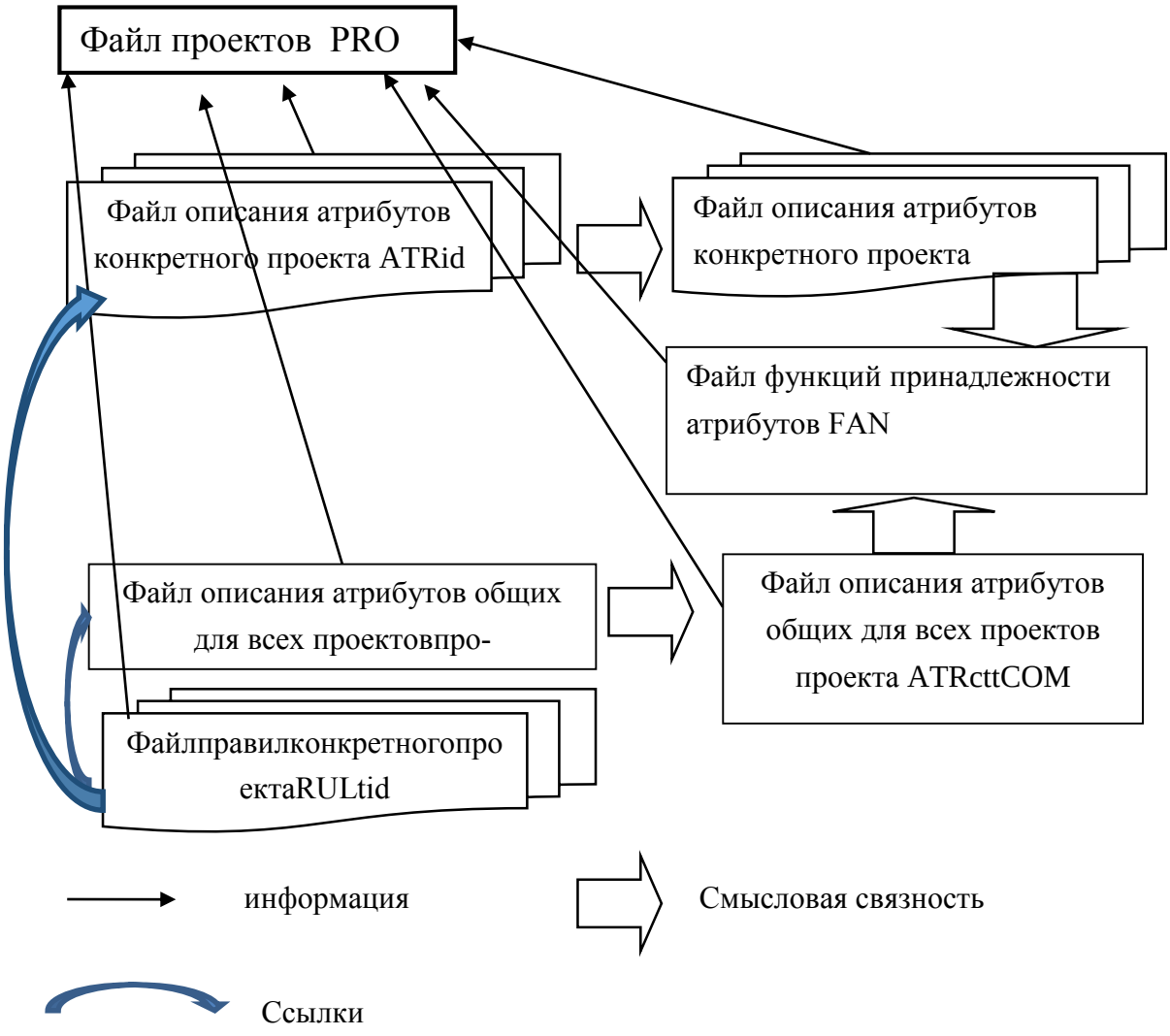


Рис. П6. Взаимосвязи между файлами БЗ советующей системы

Приложение 2

Построение и описание механизмов вывода

В качестве способа представления знаний выбраны продукционные правила или продукции. Продукционные правила хранятся в базе правил или базе знаний (БЗ). Продукционные правила создаются из терминологии словаря в редакторе БЗ. Хранятся правила в отдельных файлах. Формат и назначение файлов представлен ниже. Структура файлов выбрана из соображений наглядности и удобства доступа из редактора БЗ, при проведении конкретных экспертиз и внешними средствами.

После наполнения БЗ, проверки её полноты и не противоречивости БЗ знаний доступна для проведения экспертизы. Под экспертизой понимается выработка советов, предложение решений конкретных проблем. Собственно, сам термин экспертиза используется потому, что для наполнения БЗ используются знания квалифицированных специалистов-экспертов. Сам пользователь может наполнять базу правилами, которые он выработал в процессе своей профессиональной деятельности. Для проведения экспертизы разработан отдельный модуль, который представляет механизм логического вывода. Механизм получения решения назван логическим, поскольку само правило похоже на логическую высказывательную форму:

«Если из А следует В и А– истина, то В тоже истина».

Сам механизм вывода можно представить различными способами, например, связи между значениями атрибутов, которые описаны правилами, представить ребрами графа и тогда БЗ будет иметь жесткую структуру, а вывод будет заключаться в нахождении маршрутов от значений одних атрибутов (представленных вершинами графа) к другим. В этом случае добавления или изменения в базе правил затруднены. Например, добавление или удаление правила приводит к изменению размерности матрицы, представляющей граф. В этом случае затруднено построение процесса объяснения полученных результатов, что может уменьшить

степень доверия пользователя к предлагаемым системой решениям. В качестве механизма вывода решения предлагается использовать алгоритмы, представленные автором ниже.

Процедура вывода должна использоваться в двух режимах: в режиме логического вывода и в режиме отладки процедур вывода (получения решений).

Автором предлагается использовать две процедуры вывода: от данных к цели (DaTarget) и от цели к данным (TarData). Фактически такой выбор используется в начале задания условий экспертизы. На такой выбор могут повлиять разные последовательности фраз диалога с пользователем либо знание структуры БЗ конкретной экспертизы.

При использовании DaTarget процедура вывода «спрашивает» у пользователя значения всех атрибутов, даже если они не потребуются при выводе. Процедура TarData изначально «требует» выбора предмета экспертизы (значения какого атрибута пользователь хочет получить), а потом «спрашивает» значения только тех атрибутов, которые потребуются для вывода решения. Выбор процедуры вывода решения производится в меню экспертизы.

При использовании механизма вывода DaTarget пользователю будут выданы возможные решения по всем предметам экспертизы, а не только по тем, которые нужны пользователю. Справиться с этим можно введя ограничения на использование неактуальных атрибутов. Второй проблемой, с которой пришлось столкнуться, является неприятная возможность начала вывода с продукций, которые содержат значения одного терминального атрибута, а для получения второго атрибута необходимо использовать другую продукцию. Эта проблема решена с помощью процедуры «отката». Структуры алгоритмов приведены на рисунках ниже.

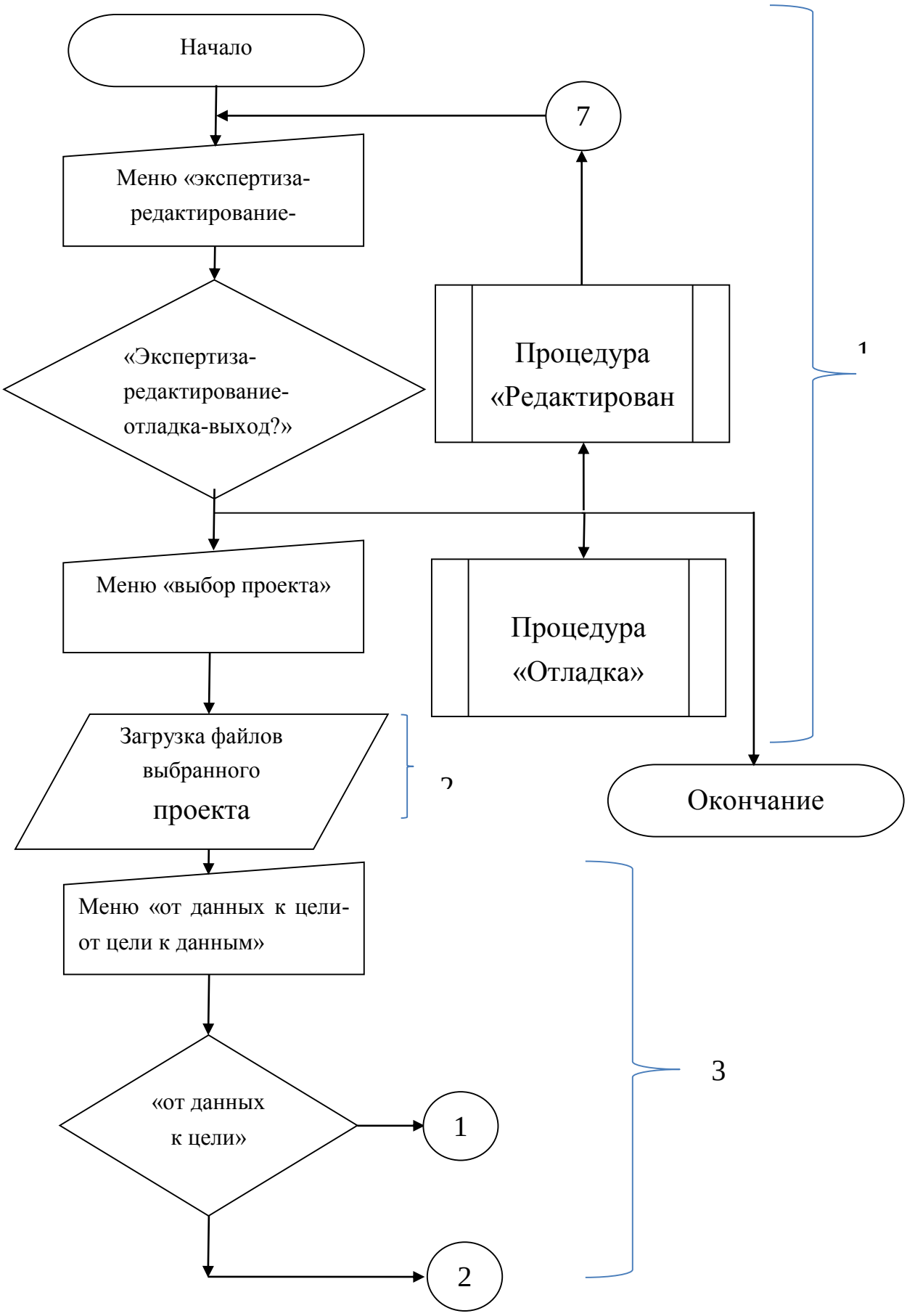


Рис. П7 Алгоритм логического вывода ЭС

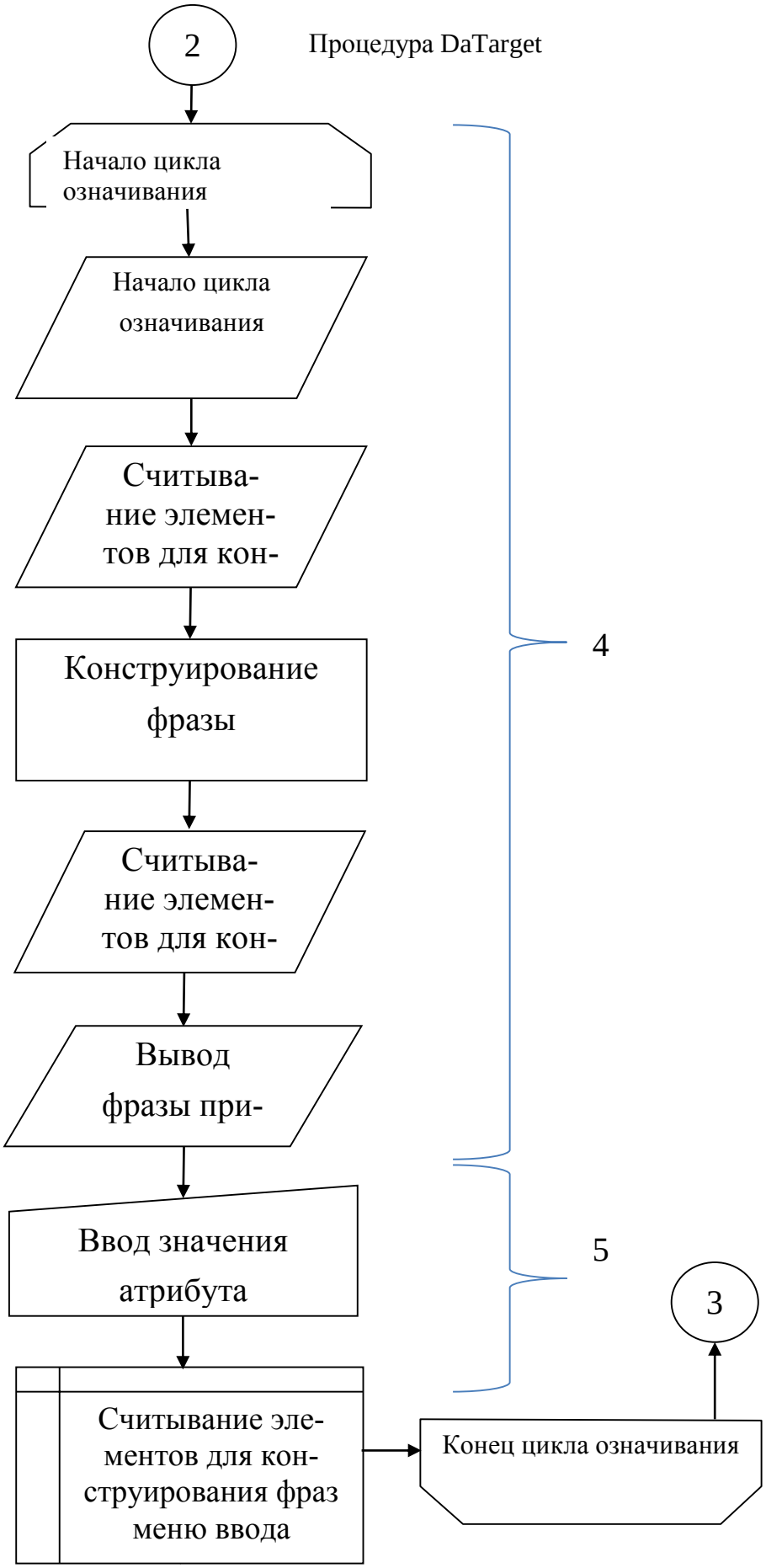

2
Процедура DaTarget
Начало цикла означивания
Начало цикла означивания
Считыва-ние элемен-тов для кон-
Конструирование фразы
Считыва-ние элемен-тов для кон-
Вывод фразы при-
Ввод значения атрибута
Считывание эле-ментов для кон-струирования фраз меню ввода
Конец цикла означивания
3
4
5

Рис. П8. Процедура DaTarget (начало)

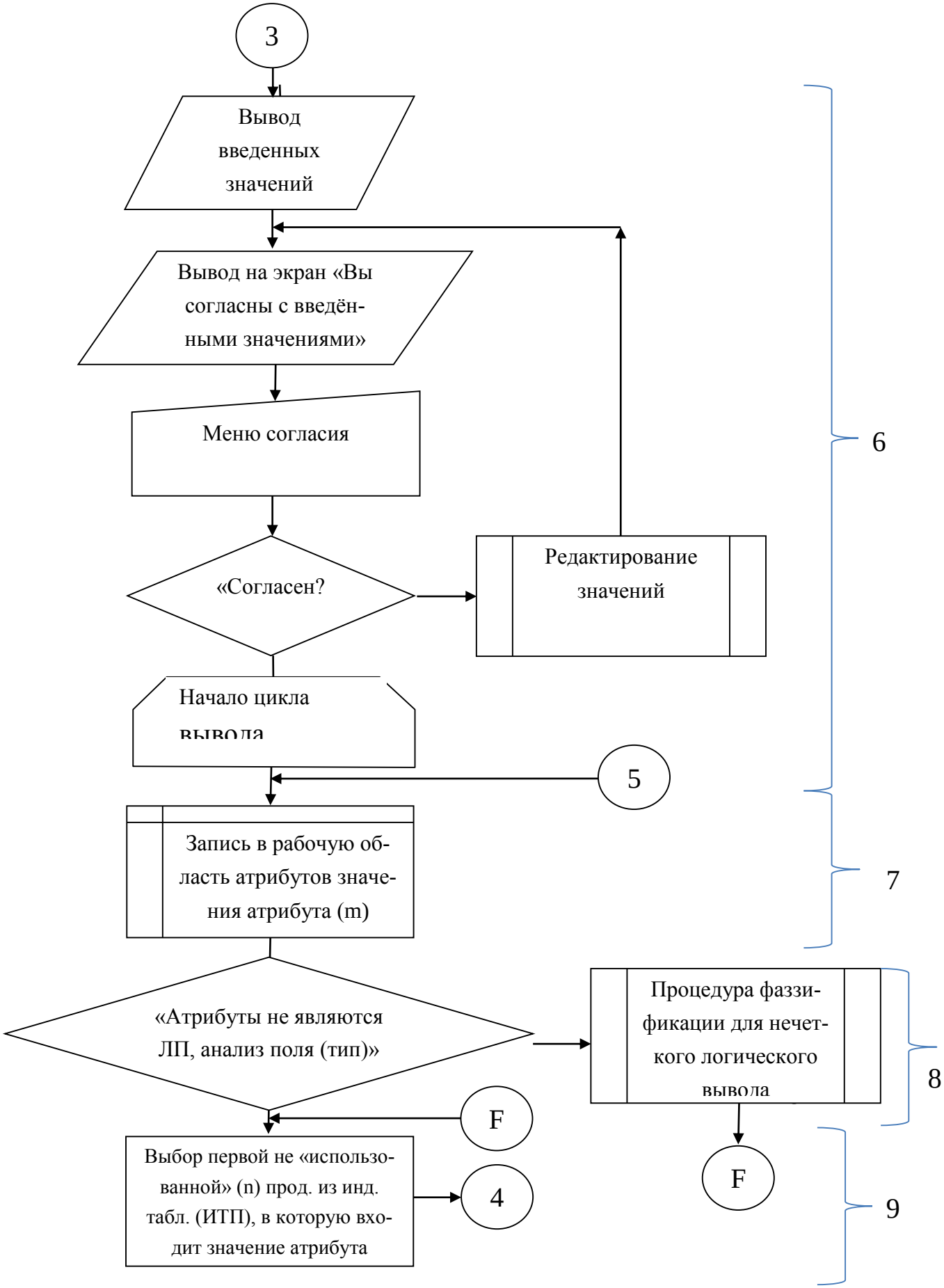


Рис. П9 Процедура DaTarget (продолжение)

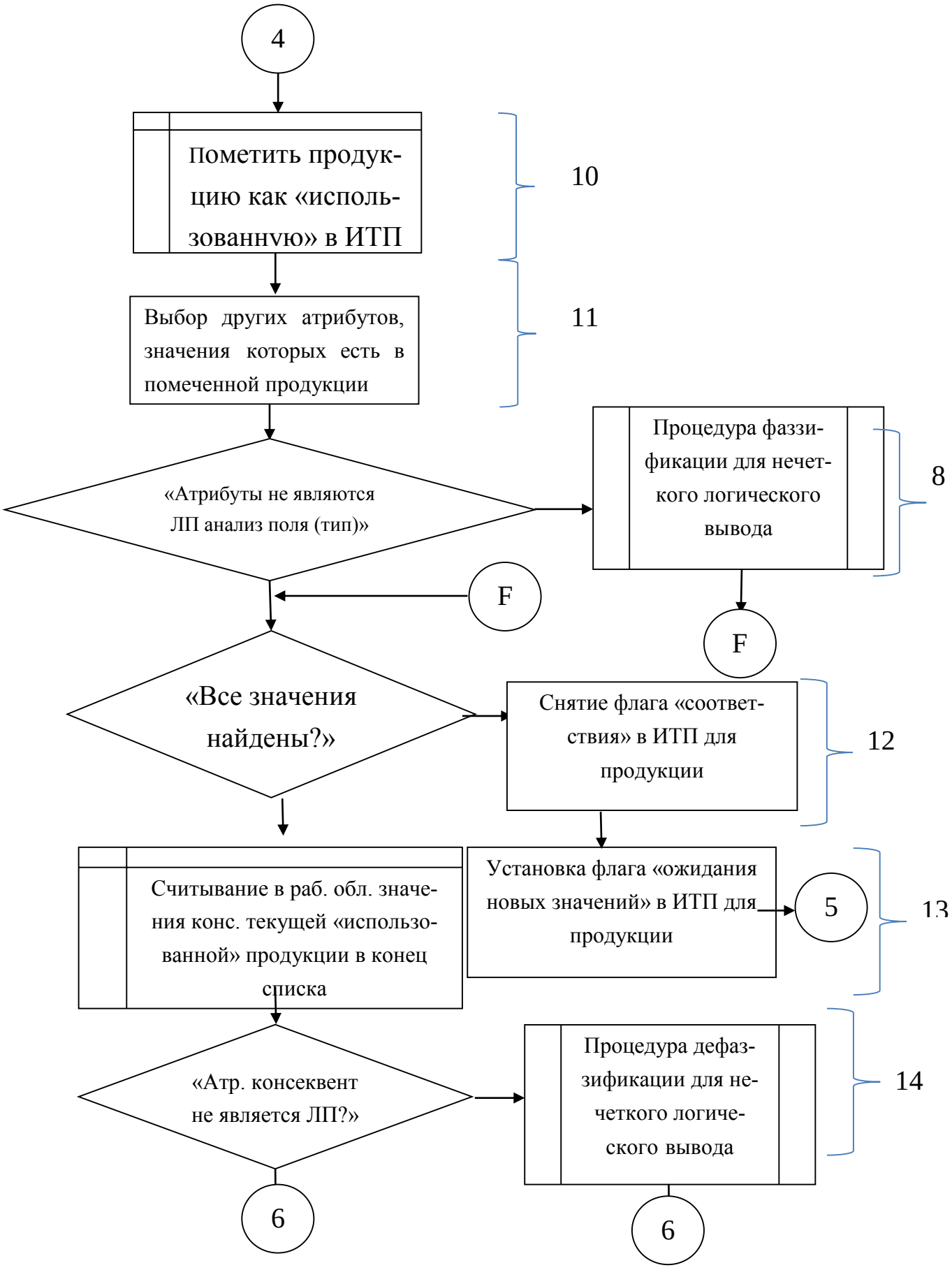


Рис. П10. Процедура DaTarget (продолжение)

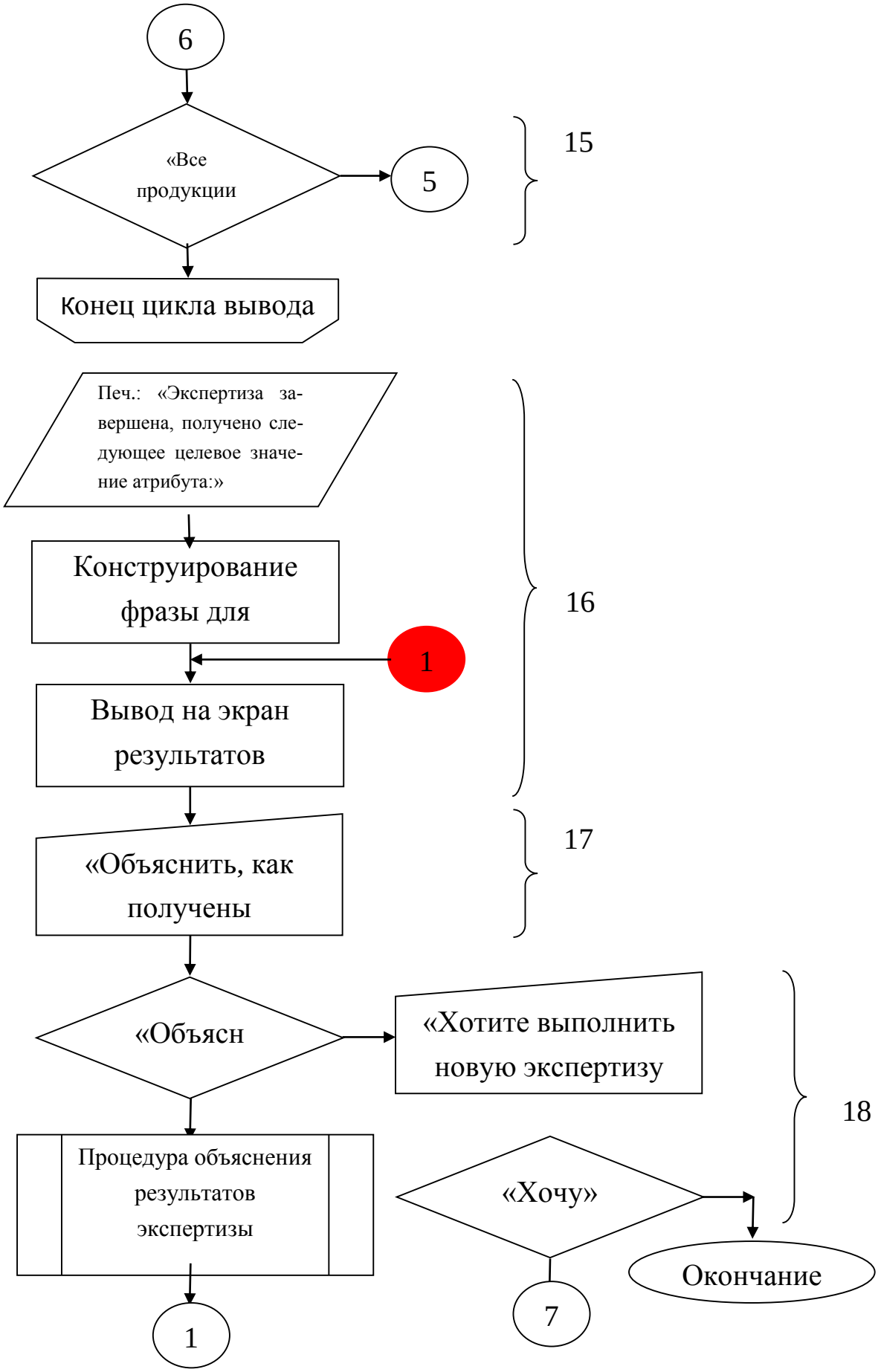


Рис. П11. Процедура DaTarget (продолжение)

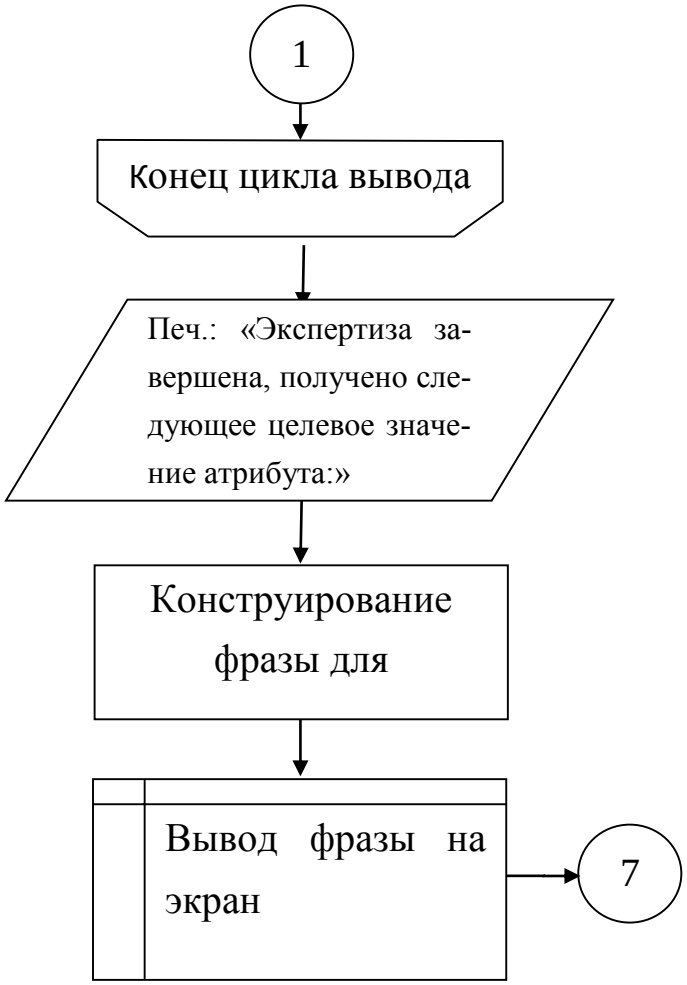


Рис. П12. Процедура DaTarget (окончание)

Описание алгоритма начала вывода и процедура DaTarget:

1. Меню «экспертиза – редактирование – отладка» является главным меню платформы, логическая последовательность начинается с пункта меню «редактирование», создаётся словарь предметной области, с использованием атрибутов словаря заполняется база продукционных правил и элементы фраз для объяснения результатов вывода.

После окончания заполнения словаря и базы правил необходимо проверить полноту и непротиворечивость полученной базы правил. Для этого выбирается пункт меню «отладка». При этом отладчик последовательно проверяет достижимость целевых значений атрибутов от значений терминальных информационных атрибутов (полнота) и отсутствие решений, предлагающих несколько разных значений одинаковых атрибутов (непротиворечивость).

2. Меню «выбор проекта» позволяет выбрать один из созданных проектов для выполнения «экспертизы».
3. Экспертиза может проводиться в двух режимах «прямого» и «обратного» вывода в зависимости от потребностей пользователя.
4. В режиме «прямого» вывод начинается с означивания атрибутов- для каждого атрибута пользователю требуется ввести конкретное значение. При этом фраза диалога с пользователем формируется из имени атрибута, всех возможных значений атрибута и служебных фраз.
5. Пользователь должен выбрать одно из предложенных значений каждого атрибута. Введённые значения записываются в предрабочую область. Предрабочая область предназначена для промежуточного хранения введённых пользователем значений атрибутов. Значения из этой области можно редактировать. Цикл повторяется для всех значений атрибутов.
6. Далее у пользователя уточняется правильность введенных значений. Имена и значения атрибутов выводятся на экран. В процедуре «редактирование значений» без повторения цикла означивания можно изменить введённые ранее значения.
7. После согласия пользователя на использования введённых значений в экспертизе атрибуты из предрабочей области последовательно переписываются в рабочую область.
8. Для атрибутов, представленных лингвистическими переменными производится процедура фаззификации, т.е. при означивании пользователь вводит числовое значение атрибута из доступного отрезка предметной шкалы, при фаззификации с использованием функций

принадлежности определяется достоверность для каждого терма из терм-множества и далее выбор продукций производится для каждого активного терма ($\mu>0$).

9. После заполнения базы правил формируется ИТП (индексная таблица правил), запись (строка) которой состоит из номера правила, номеров (индексов) использованных в ней атрибутов и поля для «флага использования» и «флага соответствия». Изначально выбирается продукция, для которой соблюдается первое условие антецедента.
10. «Флаг использования» устанавливается в «1».
11. Проверяются удовлетворение остальных условий антецедентов. Если все условия оказались истинными, то устанавливается в «1» «флаг соответствия». В случае если хотя бы одно такое соответствие отсутствует «флаг соответствия» остаётся нулевым и выбирается следующая неиспользованная продукция.
12. Проверка полноты соответствия продукции.
13. Считывание консеквента и запись в рабочую область.
14. Для атрибутов, представленных лингвистическими переменными, производится процедура дефаззификации, т.е. на основе достоверностей термов антецедента модифицируется значение функций ЛП консеквента и методом центра тяжестей вычисляется числовое значение на предметной шкале.
15. Проверка условий окончания экспертизы, если экспертиза не закончена продолжается поиск «соответствующих» продукций для других значений атрибутов, в том числе и для полученных в процессе вывода.
16. Если экспертиза закончена печатается сообщение об этом, подготавливается и выводится фраза с результатами экспертизы.
17. Выясняется желание пользователя получить объяснение каким образом получен результат вывода.
18. Производится объяснение результатов вывода, переход к новой экспертизе или редактированию данных и правил.

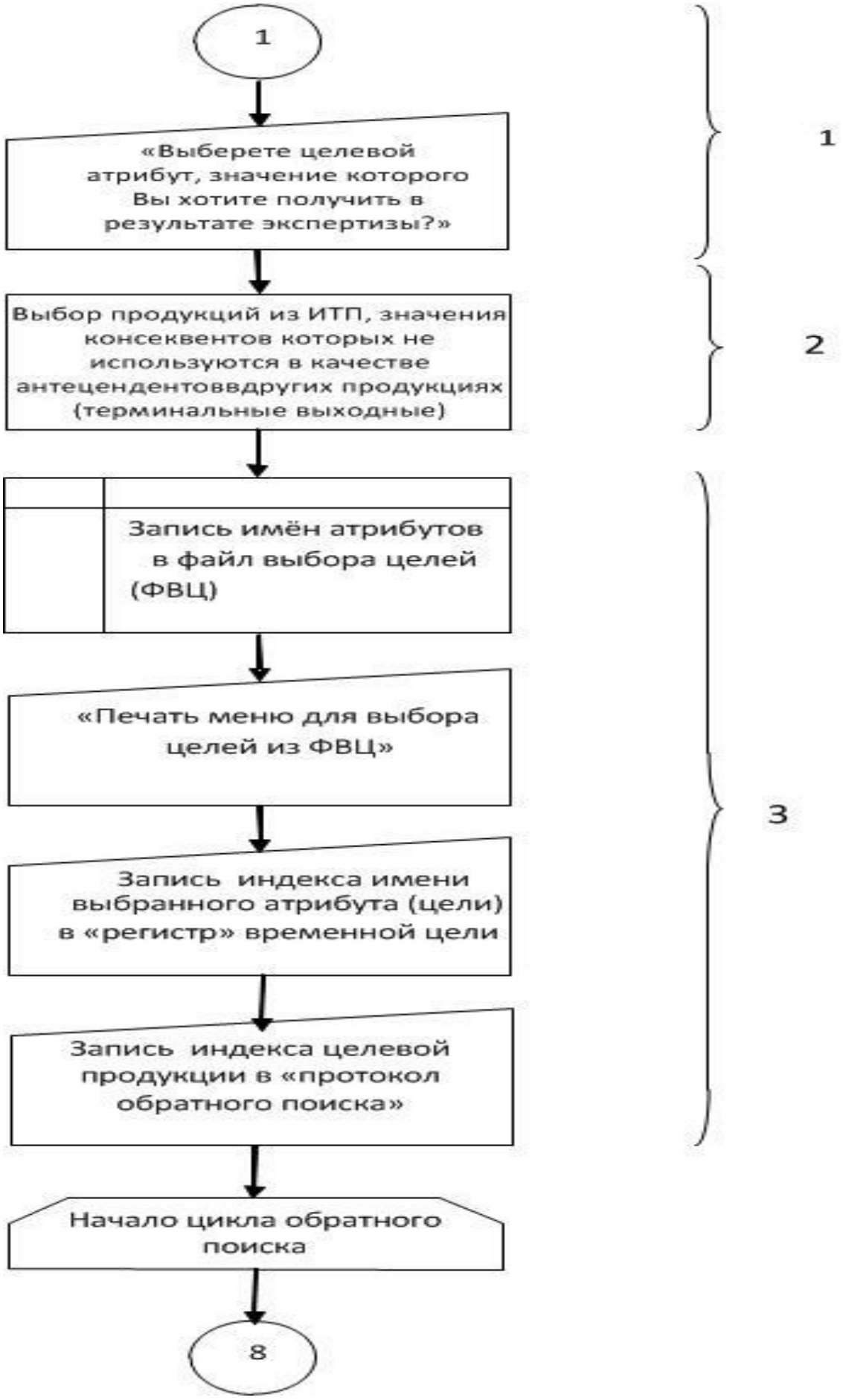


Рис. П13. Алгоритм процедуры вывода «от цели к данным» (TarData)

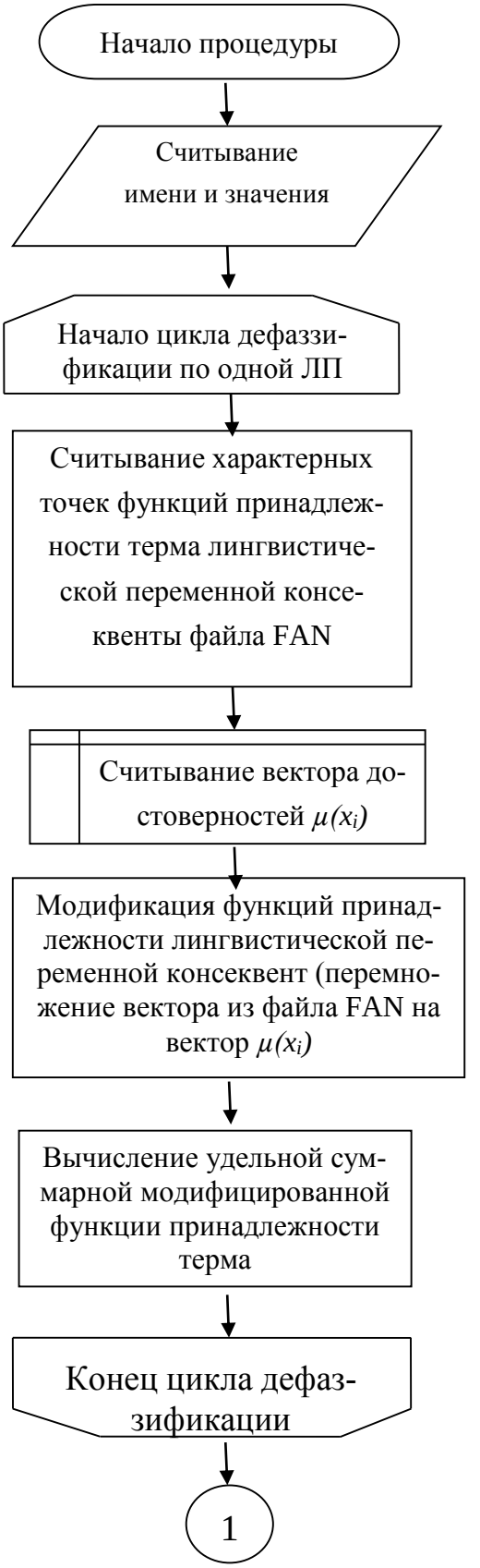
Рис. П14. Начало цикла обратного поиска

Фаззификация по одной продукции

В этом цикле вычисляется значение лингвистической переменной для конкретного значения предметной шкалы введенного пользователем или полученного иным образом. Вычисление производится последовательным перебором всех функций принадлежности ЛП, вычислением значений достоверностей и установлении «флага» активности для термов с ненулевым значением достоверности. Если для терма, значение которого использовано в продукции «флаг» активности остался нулевым, то такая продукция объявляется не подходящей и продолжается поиск подходящей.

Рис. П15. Алгоритм процедуры фаззификации для нечеткого прямого вывода

Рис. П16. Фаззификация по одной продукции



Подготовка функций принадлежности термов для использования (для случаев триангулярных функций).

Подготовка значений достоверностей активных термов использованных продукций.

Изменения вида функций принадлежности активных термов в соответствии с выбранным способом (умножения или обрезанием).

«Наложение» фигур, ограниченных измененными функциями принадлежности и предметной шкалой (осью).

Рис. П17. Алгоритм процедуры дефаззификации для нечеткого прямого вывода (начало)

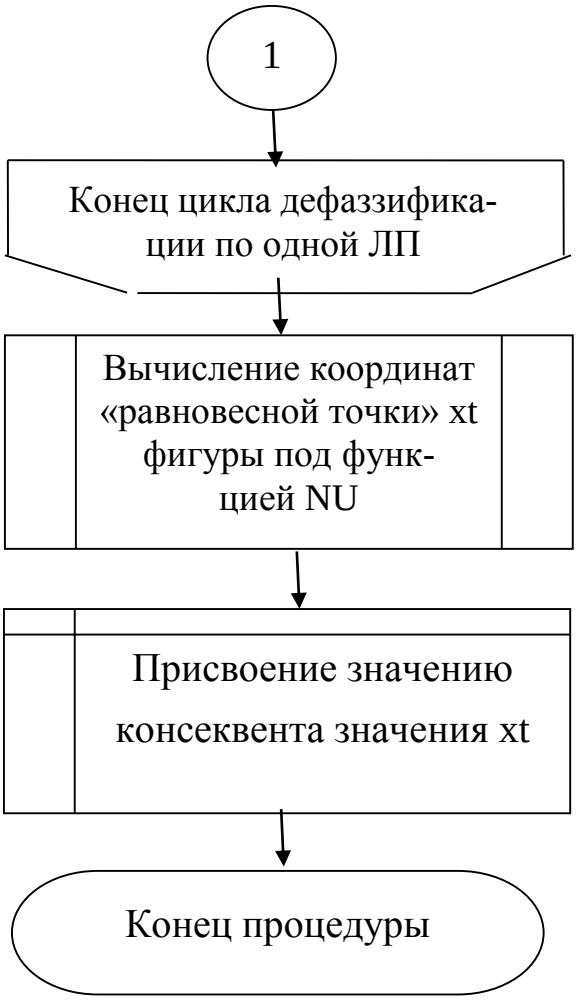


Рис.П18. Алгоритм процедуры дефаззификации для нечеткого прямого вывода (продолжение)

Через равные промежутки предметной шкалы вычисляется значение функции NU, которая далее интерпретируется как вес части фигуры под функцией. После этого для средней точки основания фигуры вычисляются веса частей фигуры справа и слева от неё. Затем текущая точка перемещается вправо или влево до достижения равной суммы весов с обеих сторон. Полученное значение и является значением консеквента.

Приложение 3.

Структура файлов БЗ

Редактор БЗ предназначен для создания и редактирования БЗ, создания условий для дальнейшего использования нескольких экспертиз по различным предметным областям. Основная задача редактора знаний (РЗ) интеллектуальной аналитической платформы (ИАС) – представление знаний, полученных от эксперта во внутреннем формате файлов БЗ. Состав и взаимосвязь файлов представлена на рисунке ниже.

Второй важной функцией РЗ является отладчика БЗ. Отладчик содержит упрощенный вариант системы логического вывода и предназначен для исследования полноты и непротиворечивости заполненной БЗ. Несоблюдение требования полноты не позволит получить результат вывода для некоторых значений исходных атрибутов. Не соблюдение требований непротиворечивости может привести к получению для одних и тех же входных данных разных и противоречивых выходных результатов.

Рекомендуется для каждого предмета экспертизы создавать отдельную БЗ. Для этого в пункте меню «проект» необходимо выбрать операцию «создать» и в предложенном поле ввести «идентификатор проекта», который записывается в файл PRO. Идентификатор проекта может включать до 8 букв латинского алфавита, цифр и символов. Предположим, мы создали проект с идентификатором «AbCdIfGh», в этом случае для размещения БЗ размещаемого проекта будут редактором созданы следующие файлы для размещения создаваемой БЗ:

- ATRidAbCdIfGh – файл атрибутов;
- ATRsetAbCdIfGh – файл значений атрибутов;
- RULTidAbCdIfGh – файл продукций;
- FANAbCdIfGh – файл фукций принадлежности для атрибутов, описанных лингвистическими переменными;
- EXPLAbCdIfGh – файл для хранения компонентов формирования текста объяснения результата вывода.

Структура файлов базы знаний советующей системы представлена ниже.

1	Внутренний идентификатор атрибута	2	Наименование атрибута	3	Идентификатор типа атрибута	4	Элементы описания атрибута

Рис. 19. Файл ATRid (идентификатор проекта)

Внутренний идентификатор атрибута формируется автоматически при описании атрибута. Нумерация уникальна в рамках одного проекта. В разных проектах могут встречаться одинаковые идентификаторы. Существует возможность создания общей базы атрибутов для нескольких или для всех проектов (экспертиз). В этом случае добавляется поле «доступность», со значениями «для всех проектом» (ALL) или ссылкой на файл доступности (AccessFile).

Идентификатор типа атрибута выбирается из меню, которое открывается при переходе в соответствующее поле. Возможен выбор из следующих значений: «Числовое»; «Символьное»; «Нечёткое»; «Логическое»; «Дата». Для рассматриваемой предметной области предлагается ограничить число типов значений.

Файлы ATRset… могут создаваться для каждого типа атрибутом. Мы будем использовать единый файл для всех атрибутов, учитывая возможности его модификации при введении ограничений на число типов атрибутов.

Поля «Список значений» и «Имя файла функции принадлежности» представляют повторяющиеся поля, т.е. поля, которые могут иметь несколько значений. Значение поля «Имя файла функции принадлежности» формируется автоматически добавлением к идентификатору атрибута

номера значения атрибута. Одновременно создаётся файл с таким именем. Файл заполняется редактором «Функции принадлежности». В файлах «Функции принадлежности» могут храниться функции в виде диапазонов и коэффициентов кусочно-линейных функций принадлежности, в представленном ниже виде.

Файл ATRset (идентификатор проекта)

1	Внутренний идентификатор атрибута	2	Идентификатор типа атрибута	3	Число значений	4	Список значений (термов)

5	Имя файла функции принадлежности (ссылка)	6	Минимальное значение предметной шкалы	7	Максимальное значение предметной шкалы	8	Шаг предметной шкалы

Рис. П20.Файл ATRset (идентификатор проекта)

Файл FAN (идентификатор проекта) (идентификатор атрибута) (номер темпора)

	Точка 1 ЛП		Точка2 ЛП		Точка3 ЛП		Точка 4

Рис. П21. Файл FAN

Должна существовать возможность ввода вида функций принадлежности в виде формул на диапазонах трёх диапазонах значений, а также посредством множества точек.

Файл RULTid(идентификатор проекта)

1	Внутренний идентификатор продукции	2	Внутренний идентификатор атрибута антецедента i1	3	Значение ki1 атрибута антецедентаi1	4
		4	Внутренний идентификатор атрибута антецедента i2	5	Значение ki2 атрибута антецедента i2	
			……		……	
			Внутренний идентификатор атрибута антецедента im		Значение kim атрибута антецедента im	
		6	Внутренний идентификатор атрибута консеквента j1	7	Значение l1 атрибута консеквента j1	
		8	Внутренний идентификатор атрибута консеквента j2	9	Значение l2 атрибута консеквента j2	
			Внутренний идентификатор атрибута консеквента jp		Значение l атрибута консеквента j1	

Рис. П22. Файл RULTid

Нижеследующие рассуждения будут проведены для атрибутов с дискретным множеством значений. Такое допущение позволяет покрыть достаточно большие области применения. Если представить конкретные атрибуты со всеми возможными значениями в виде графа, то продукция может быть представлена тоже в виде графа (рис. П27).

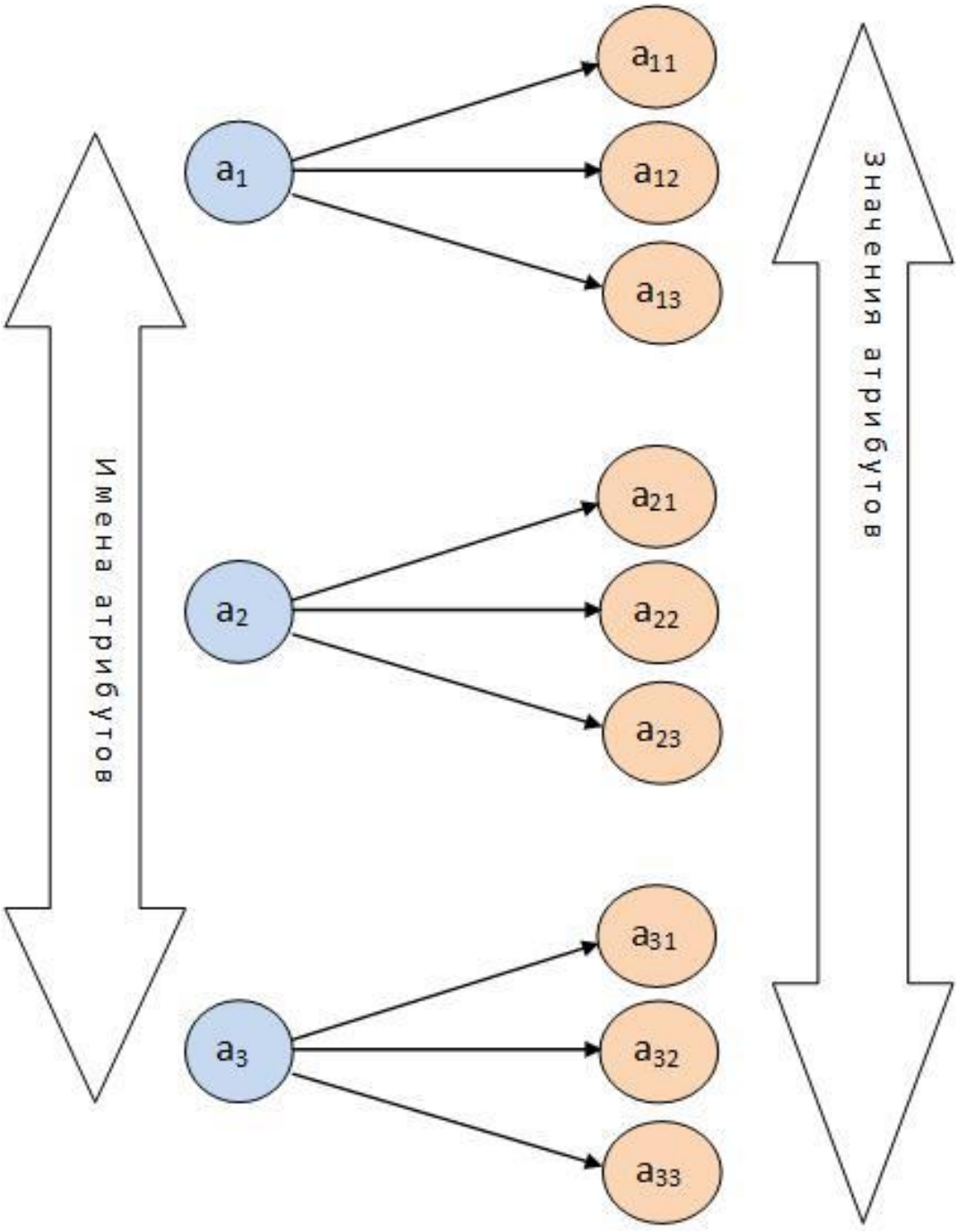


Рис. П23. Представление области определения атрибутов в виде графов

В случае многошагового вывода полученное после применения представленной на рис. 24 продукции значение a_{23} атрибута a_3 должно участвовать в качестве антецедента другой продукции. Для обозначения достаточности полученного решения некоторые атрибуты должны быть помечены как терминальные выходные при редактировании БЗ. При получении значения терминального атрибута или нескольких атрибутов логический вывод останавливается. Если при редактировании ни один атрибут не объявлен терминальным, то вывод останавливается при получении значения атрибута, указанного в условиях экспертизы.

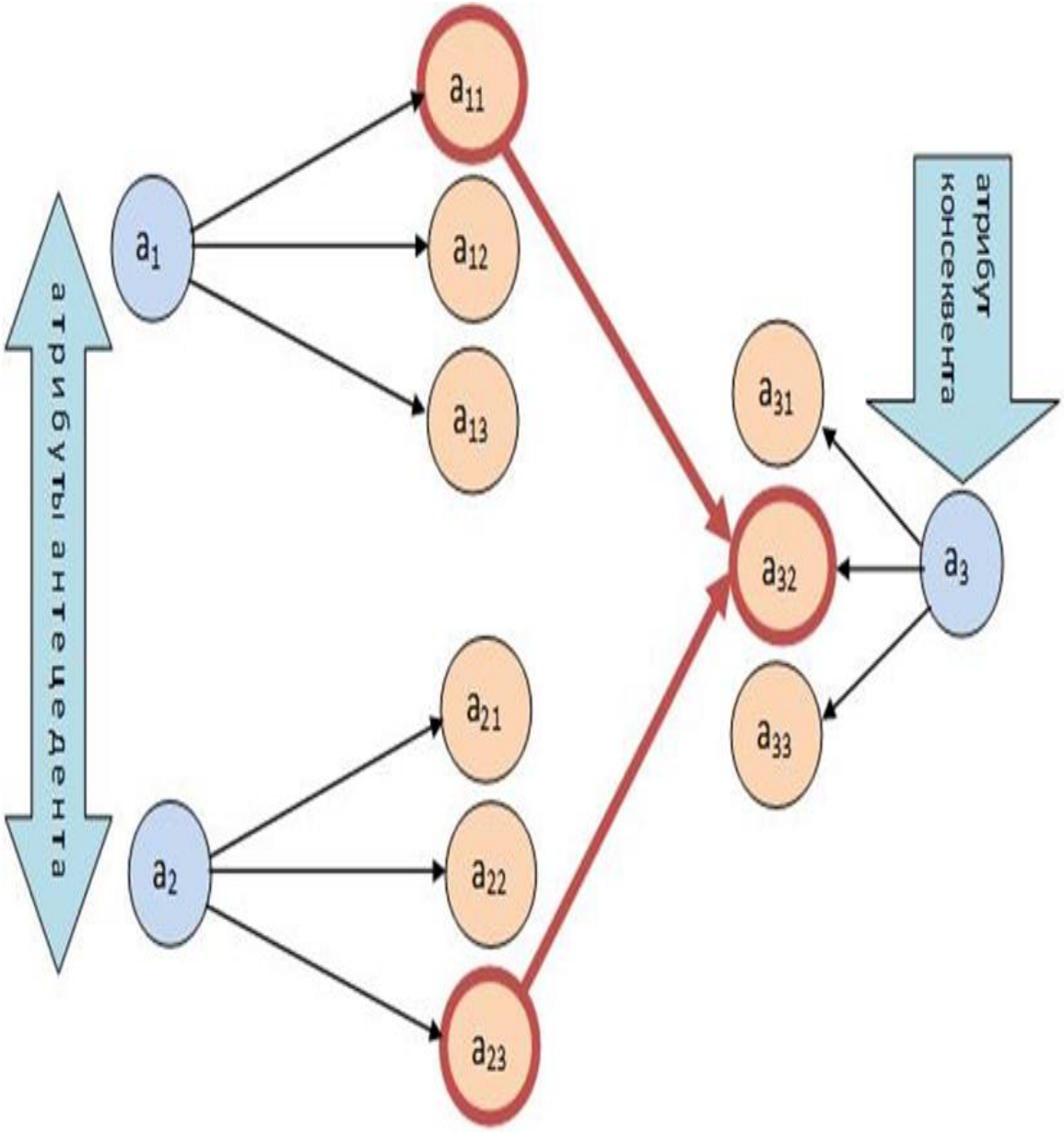


Рис. П24. Представление в виде графа продукции

Атрибуты, значения которых не могут быть получены посредством применения продукций, а могут быть получены только из источников внешних по отношению к ИАС (в том числе посредством ответа на вопросы, поставленные ИАС), назовём входными терминальными атрибутами.

«ЕСЛИ ***атрибут*** a_1 ***принимает значение*** a_{11}
И ***атрибут*** a_2 ***принимает значение*** a_{23}**,**
ТО ***значение атрибута*** a_3 ***следует считать равным*** a_{23} **».**

Работа отладчика по проверке полноты заключается в последовательном переборе (применении) всех продукций, построении графа вывода и нахождении всех маршрутов от всех значений входных терминальных атрибутов к значениям выходных терминальных атрибутов. Если хотя бы из одного значения такого маршрута не найдено, то условие полноты считается не соблюденным. В этом случае отладчику необходимо инженеру по знаниям указать точки начала и окончания маршрута, нарушающего полноту. Устранение неполноты может быть достигнуто добавлением новых продукций и/или удалением продукций и атрибутов «виновных» в неполноте.

Работа отладчика по проверке непротиворечивости заключается в проверке отсутствия маршрутов при одинаковых значениях начальных величин, оканчивающихся разными значениями одной и той же величины. Окончания вывода разными значениями разных атрибутов не говорят о наличии противоречивости.

Следует различать три типа представлений одного и того же продукционного правила в нашей аналитической платформе:

Продукционное правило, представленное аналитиком на основе экспертных знаний, используется для визуального представления в редакторе продукционных правил, при объяснении результатов логического вывода.

Продукционное правило, представленное вместе со служебной информацией (идентификаторы атрибутов, идентификаторы значений продукции, идентификаторы условий применимости продукции), используется для отладки, проверки на полноту и непротиворечивость базы правил.

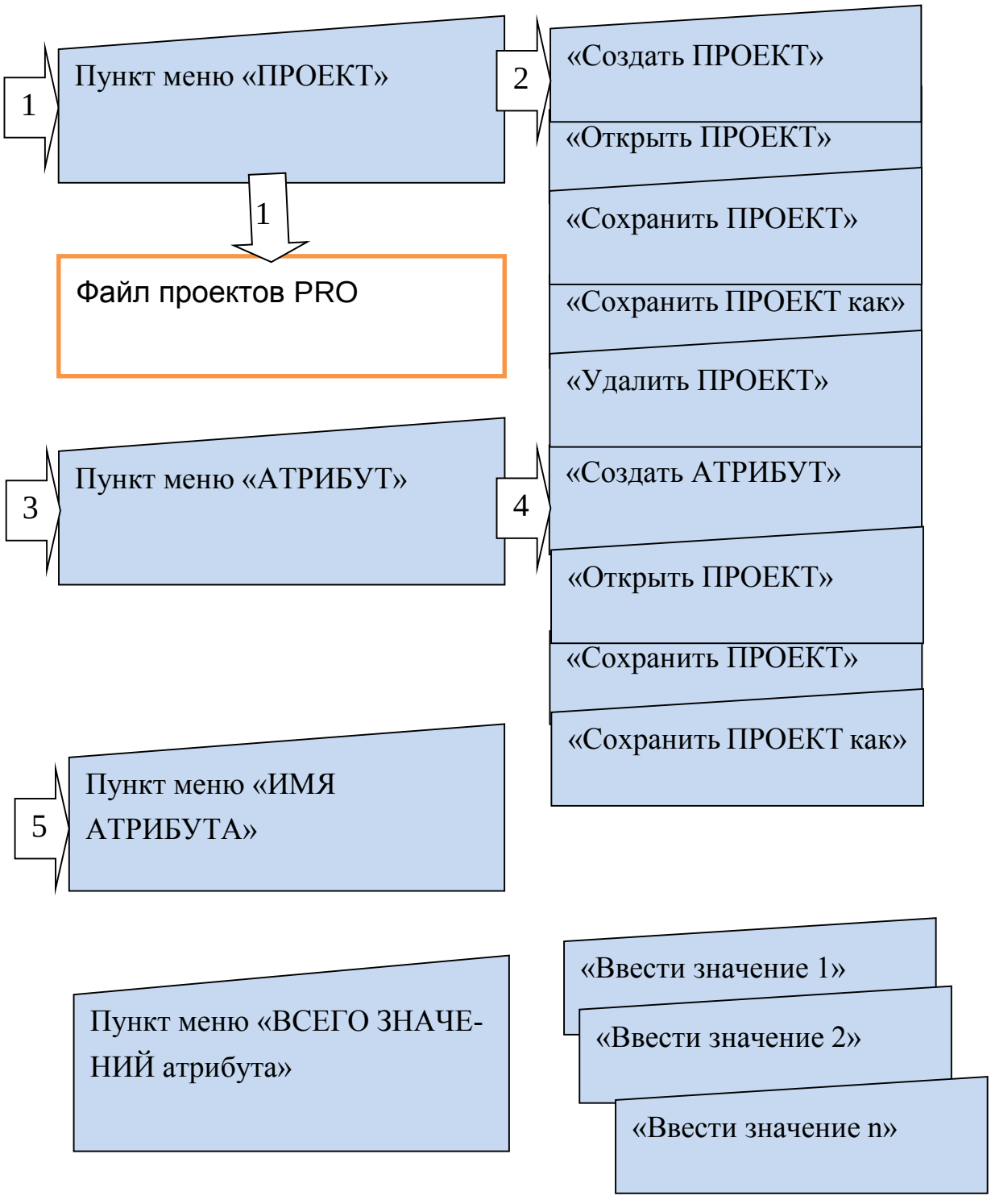


Рис. П25. Структура меню редактора

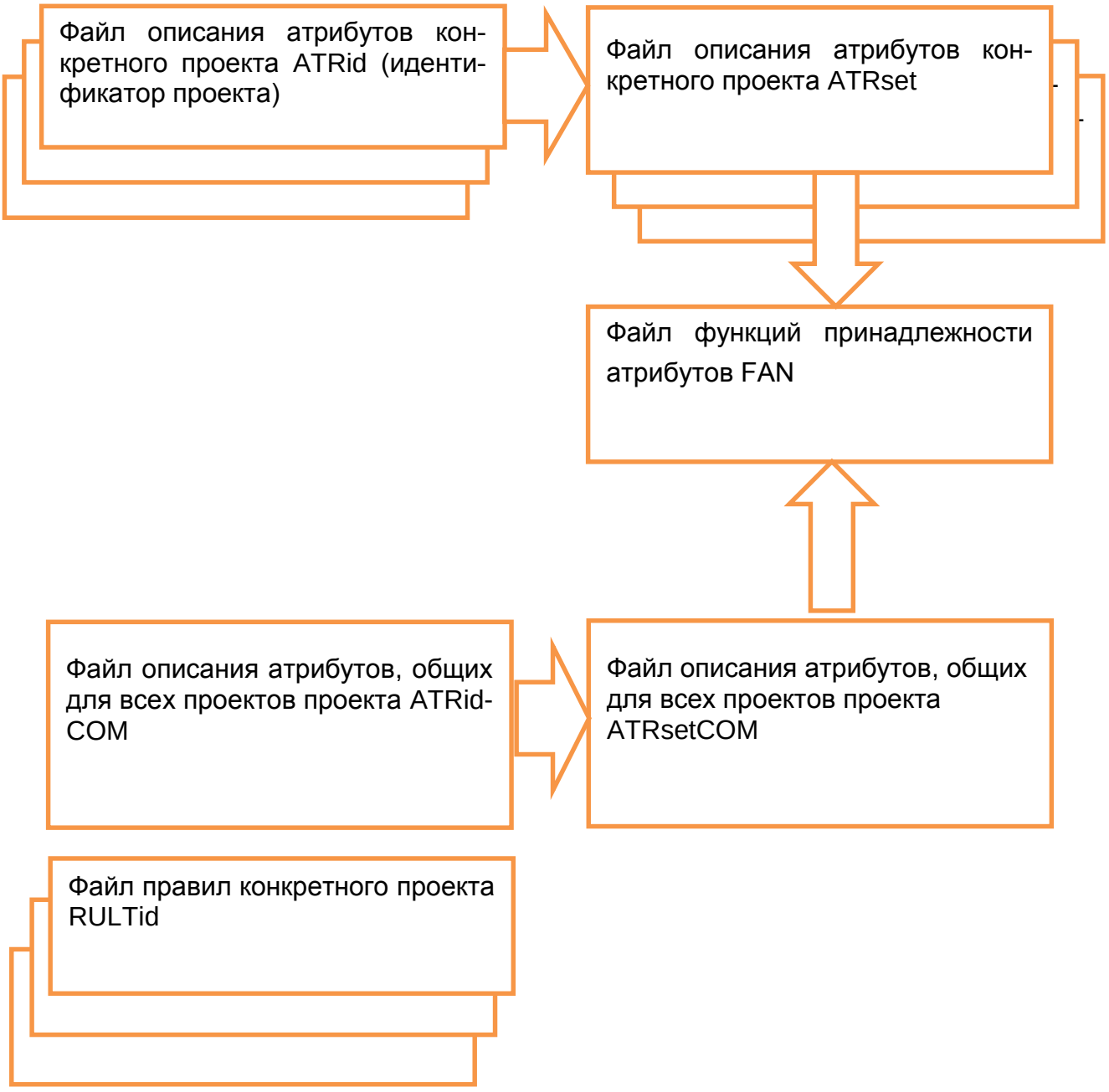


Рис. П26. Взаимосвязи пунктов меню редактора и файлов БЗ советующей системы

Приложение 4

Описание системы объяснения результатов вывода
Редактор процедур объяснения

Одним из существенных преимуществ экспертных систем (далее-систем), основанных на продукционных правилах, является возможность объяснить пользователю, почему получено и предлагается именно такое решение, а не иное. Объяснение процесса и результатов вывода полезно не только оператору-пользователю системы, но и инженеру по знаниям при отладке БЗ. Именно поэтому в таких системах появляется возможность разбить процесс разработки и внедрения системы на отдельные этапы со своими отдельными задачами. Например, демонстрационный прототип может иметь своей основной целью продемонстрировать правильность выбора способа представления, удобство интерфейсных решений и вызвать доверие эксперта и побудить его к дальнейшей работе….

Объяснение предоставляется оператору- пользователю по запросу и требует повторения процессов вывода либо использование предварительно сформированного протокола вывода. Результирующий модуль объяснения состоит из использованных для вывода значений *исходных данных* (задаются оператором или берутся из базы данных), получаемых в процессе вывода *промежуточных значений атрибутов,* полученных *результирующих значений атрибутов*, имён всех использованных атрибутов, применённых продукций с их внутренними идентификаторами и служебных фраз, имитирующих предложения ЕЯ (естественного языка). Напротив, другие способы представления и вывода, например, нейронные сети или генетические алгоритмы, не позволяют разложить процесс вывода на компоненты, которые можно представить на ЕЯ в логике рассуждений эксперта. Здесь иногда имеется возможность разработать подсистему, которая объясняла только технические аспекты получения решения и только в терминах выбранной парадигмы.

Каждому правилу, атрибуту и значению атрибута можно сопоставить текстовый файл или строку (запись или поле) файла, который (ая)

будет использоваться при объяснении результатов предложенного решения по требованию пользователя. Строки будут объединяться в текст объяснения в порядке использования правил, атрибутов и их значений. Простейшее объяснение может выглядеть следующим образом:

Вы сказали, что величина (07) атрибута 7 ***приняла*** (073) значение 3, ***а параметр*** (12) атрибута 12 *стал «более широким»* (значение 122).

«***Существует правило*** **176** «**ЕСЛИ** атрибут 7 ***равен*** значению 3 и атрибут 12 ***равен*** значению 122, ТО атрибуту 22 ***необходимо придать*** значение 221».

Служебные фразы могут быть одинаковыми для всех предложений объяснения. Но автор рекомендует для каждой величины (атрибута) написать свои служебные фразы, тогда текст объяснения будет ближе к ЕЯ и будет более понятным. Связки между атрибутами и значениями могу быть не только «РАВНО», но и «БОЛЬШЕ», «МЕНЬШЕ», «МЕНЬШЕ ЧЕМ», «БОЛЬШЕ ЧЕМ», (глаголы) «УВЕЛИЧИТЬ НА…», «УМЕНЬШИТЬ НА…»,…«ВЫПОЛНИТЬ СЛЕДУЮЩИЕ ДЕЙСТВИЯ…», (Сравнительная степень прилагательного) «КРАСНЕЕ», «СИЛЬНЕЕ»…

Если значение получено в процессе диалога с пользователем (входное терминальное значение), то в тексте объяснения можно использовать фразу: «Вы сказали, что …(придание)». Если значение получено в процессе вывода посредством применения продукции, то в тексте объяснения можно использовать фразу «После применения (использования) правила значение атрибута стало равным (придание)…». В обоих случаях целью конструируемой фразы является объяснение, каким образом атрибут получил своё конкретное значение.

Если значение атрибутов описываются существительными (одушевленными или неодушевленными), то для диалога необходимо построить вопросительное предложение, требующее указать кто? или что? является подходящим для конкретной ситуации. Например, для означивания атрибута «правообладатель» нужно построить вопросительное предложение:

«Кто является *правообладателем патента на конструкцию производимого устройства*?».

При объяснении результатов вывода:

«Вы сказали, что *правообладателем патента на конструкцию производимого устройства* является ИВАНОВ Ю. А.».

Если значение атрибутов описываются прилагательными, то для построения предложения можно использовать вопросительное слово *какой* для качественного и относительные прилагательного, *чей* – для притяжательного прилагательного.

Для числительного – *сколько, какой* или *который.*

Глаголами описываются, как правило, значения терминальных атрибутов, описывающие полученное решение.

Алгоритмы и описание процедур объяснения результатов вывода приведены на рис. П27, П28.

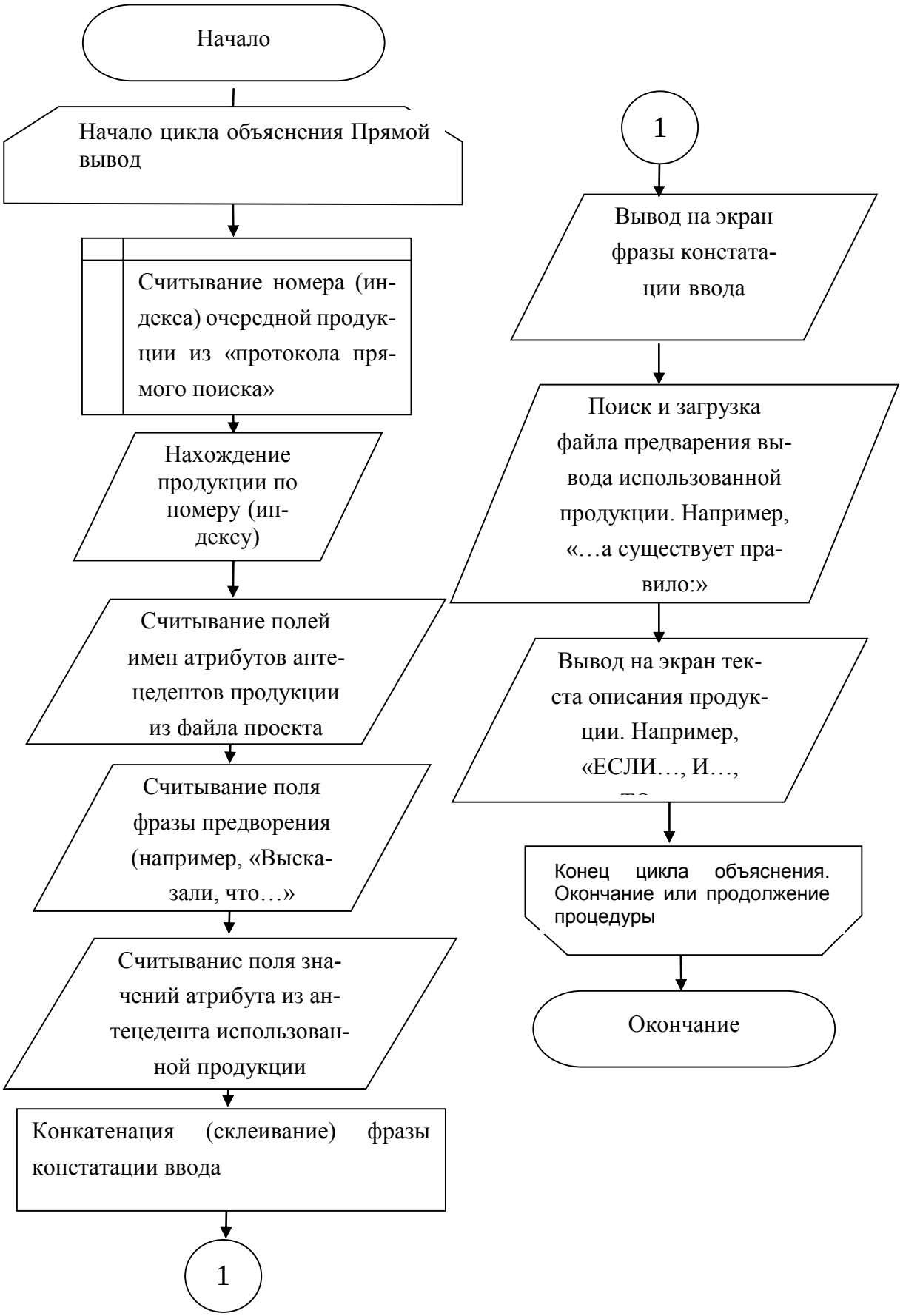


Рис. П27. Алгоритм и описание процедур объяснения результатов экспертизы для прямого вывода

1. Использование «протокола прямого поиска для» последовательного определения идентификатора примененной продукции.

2. Нахождение продукции в базе правил по идентификатору для последующего предъявления пользователю и нахождения значений атрибутов.

3. Нахождение имен атрибутов, которые использовались в этой продукции.

4. Нахождение фраз диалога, соответствующих механизму получения использованных атрибутам (ввод или результат применения продукции).

5. Нахождение значений атрибутов, использованных в выбранной продукции.

6. Формирование фразы объяснения соответствующей использованной продукции использованных атрибутов и их значениям и вывод на экран составленной фразы объяснения, отнесённой к атрибутам текущей продукции.

7. Поиск и вывод фразы, описывающей, каким образом на основе введённых или полученных при выводе значений атрибутов получено новое значение другого атрибута.

8. Печать текста использованной продукции вместе со служебной информацией (номер продукции, индексы атрибутов и их значения).

9. Объяснения до использования (объяснения применения) последней продукции из «протокола прямого поиска».

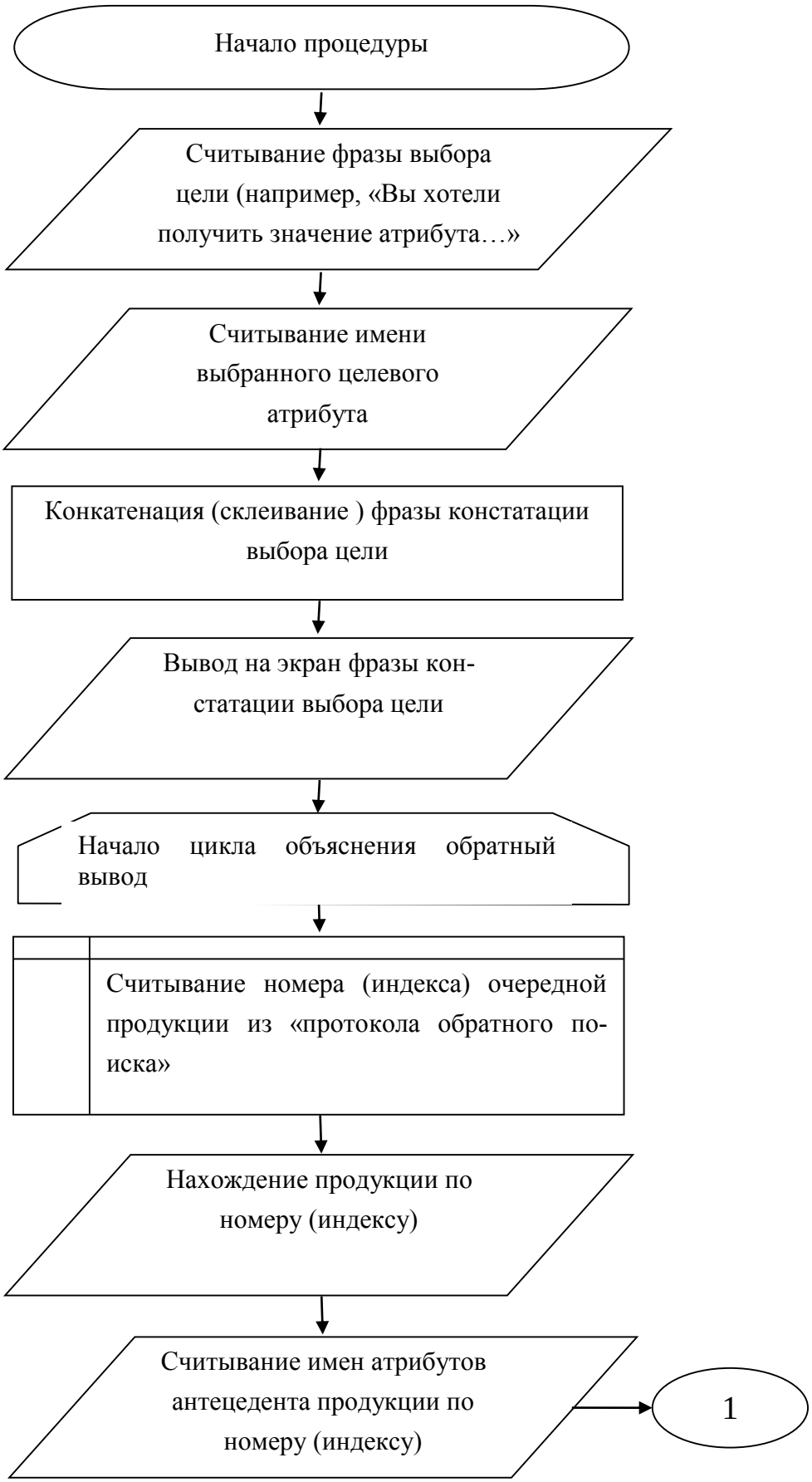


Рис. П28. Алгоритм процедуры объяснения результатов экспертизы для обратного вывода

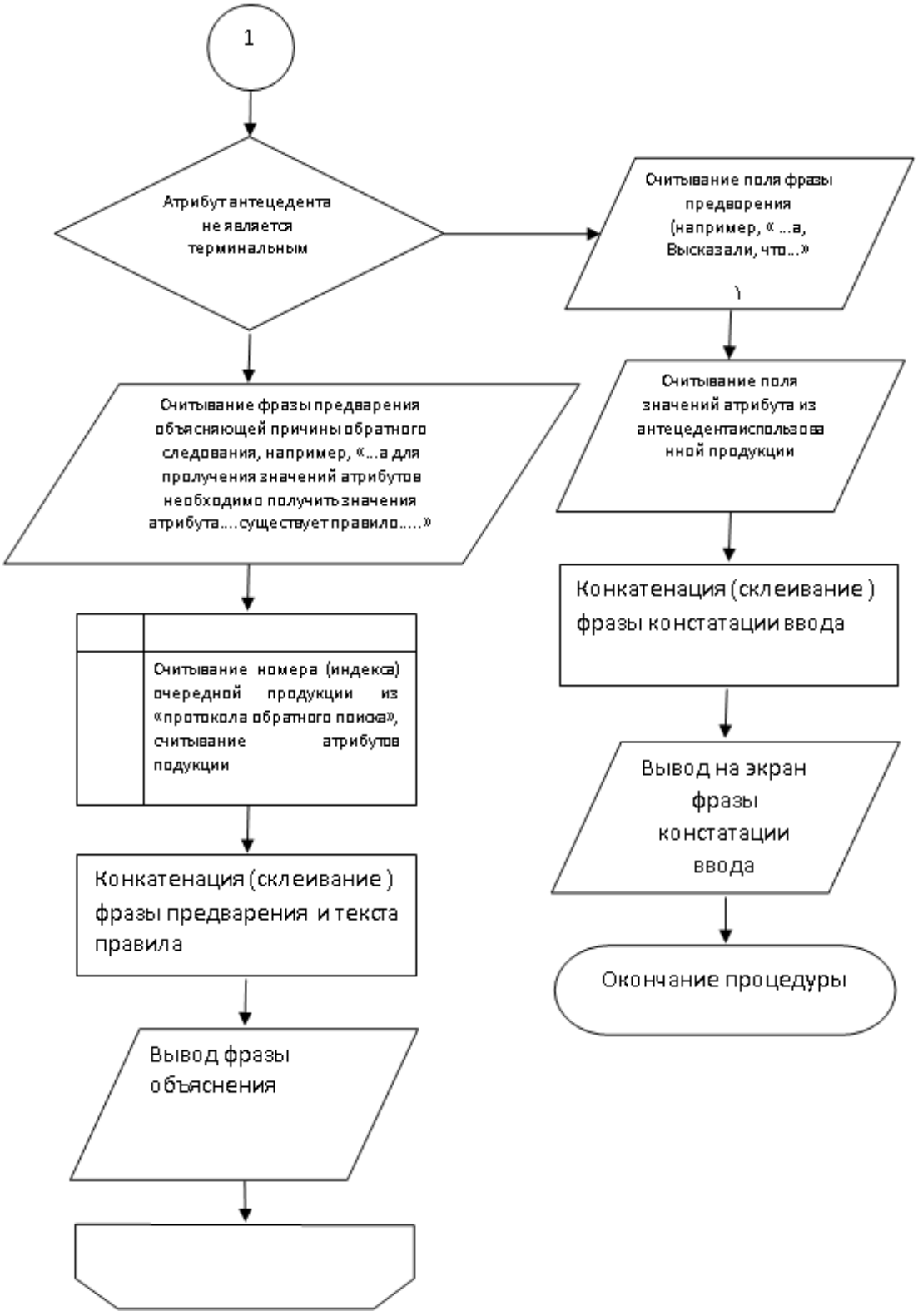


Рис. П28. Продолжение

Процедура объяснения продолжается посредством последовательного объяснения: почему использована та или иная продукция из протокола до достижения последнего использованного информационного атрибута и объяснения, откуда взято его значение. Выход из цикла – не в конце его, а в теле цикла.

Приложение 5

Описание демонстрационного прототипа ЭС

Экспертные системы на пути от проекта к коммерческому программному продукту в большинстве своем проходят стадию демонстрационного прототипа. Задача демонстрационного прототипа состоит в демонстрации правильности выбора способа представления знаний, эффективности функционирования механизма вывода и объяснения получаемых результатов, а также удобства интерфейса. Состав компонентов демонстрационного прототипа (представлен на рис. П29): редактор базы знаний, блок выполнения экспертизы, блок объяснения результатов экспертизы.

Редактор базы знаний состоит из элемента редактирования проекта, в котором инженер по знаниям для каждой экспертизы может создать отдельные компоненты для размещения файлов БЗ, редактора атрибутов, в котором описывается терминология, используемая в дальнейшем для формирования структурных компонентов знаний, редактора процедур объяснения вывода для создания компонентов, из которых впоследствия формируется текст объяснения результата по требованию пользователя.

Компоненты редактора объяснения относятся к проекту в целом (вводные фразы, ориентирующие пользователей на особенности предмета экспертизы, особенности ввода данных для экспертизы...). Например, «Производится поиск решения следующей проблемы:…». Компоненты, относящиеся к конкретным атрибутам, описывают особенности физических процессов, к которым относится атрибут. Здесь также инженеру по знаниям предоставляется возможность описать, каким образом связано конкретное значение атрибута с самим атрибутом. Самые простые связки: «равно», «больше\ меньше», «увеличить\ уменьшить на…» Компоненты, относящиеся к продукциям, позволяют изменить структуру представления вида продукции. Например, структуру "ЕСЛИ.... – ТО..." Можно заменить "В случае наличия нижеследующей ситуации... – Необходимо предпринять следующие действия...".

Все типы компонентов описания имеют значения умолчания, т.е. такие, которые будут использованы в том случае, если какие- то компоненты объяснения не будут определены разработчиком для конкретной экспертизы.

В состав редактора входит также модуль отладки БЗ, который построен на основе упрощения механизма прямого вывода. Упрощение заключается в том, что не используется интерфейсный модуль, а последовательно перебираются все атрибуты и их значения и производится поиск атрибутов и значений атрибутов, не использованных в продукциях (проверка на полноту), поиск значений информационных атрибутов, для которых могут быть получены разные значения целевых атрибутов (проверка на противоречивость). По сути дела, уже готовая к использованию БЗ представляется в виде ориентированного графа, строится матрица инцидентности и проверяется наличие маршрутов от информационных атрибутов-вершин до целевых, затем по идентификатору вершины находится наименование и значение атрибута и сообщается пользователю редактора о наличии или отсутствии коллизии.

Блок выполнения экспертизы представляет собой основную компоненту информационно-аналитической системы (ИАС) и может работать самостоятельно без других компонентов. Блок состоит из интерфейсного модуля, модулей прямого и обратного вывода. Интерфейсный модуль предназначен для создания комфортного «общения» пользователя и советующей системы, получения данных (начальных условий) для экспертизы ситуации с целью выработки решения.

Пользователю предлагается выбрать значения необходимых для экспертизы атрибутов. Предлагается составлять вопросы и описывать значения атрибутов, покрывающих большую часть предметной шкалы. Для числовых атрибутов это может быть разбиение предметной шкалы на интервалы, для лингвистических атрибутов– использование до 5-7 термов, описывающих значение на неметрической шкале расположенных в порядке

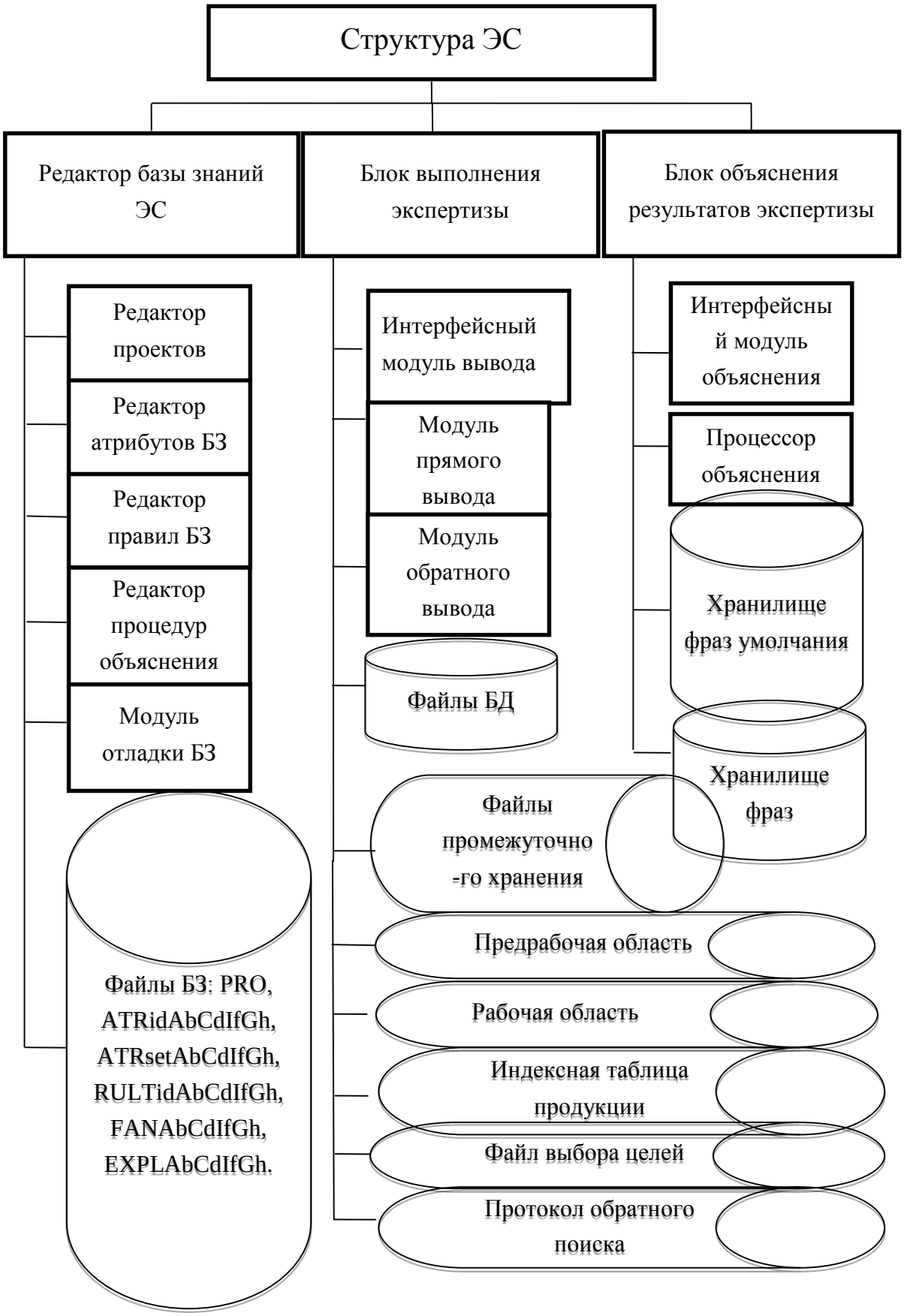


Рис. П29. Структура «Модуля экспертизы ЭС» (обработки данных)

возрастания или убывания представлений пользователя о величине значения атрибута (например, «ничтожно малое значение», «очень малое значение», «малое значение», «среднее…», «большое…», «очень большое…»).

В этом же блоке пользователь выбирает способ вывода и, следовательно, какой из блоков вывода будет использоваться (прямого или обратного вывода). При создании конкретной экспертизы инженер по знаниям и эксперт могут ограничиться возможностью использования только одного вида вывода. Прямой способ вывода рекомендую выбирать, когда, например, пользователь желает оценить сложившуюся ситуацию или определить, что он должен делать в этом случае. Обратный способ вывода удобен в том случае, если, требуется получить значения некоторых целевых атрибутов исходя из некоторых текущих данных. После получения результатов вывода интерфейсный модуль предъявляет их пользователю и предлагает объяснить как они были получены.

Блок объяснения результатов экспертизы имеет в своем составе интерфейсный модуль и процессор объяснения. Для объяснения блок может использовать записи из хранилища фраз пользователя или из хранилища фраз умолчания, если они пользователя устраивают. Используя протокол вывода и компоненты фраз из хранилища фраз умолчания и хранилище фраз пользователя, последовательно описываются проведенные операции по получению решения.

Ниже в таблице использования компонентов демонстрационного прототипа представлена связь между операциями при работе с демонстрационным прототипом и пользователями платформы. Представлены также участвующие в этих процессах компоненты и формируемые файлы постоянного и временного хранения.

Таблица 1

Использование компонентов демонстрационного прототипа ЭС

п\п	Операция, процесс	Субъект	Компонента ИАС	Результат
1	Создание проекта	Инженер по знаниям	«Редактор проектов»	Создание файлов БЗ: ATRidAbCdIfGh, ATRsetAbCdIfGh, RULTdAbCdIfG, FANAbCdIfGh, EXPLAbCdIfGh
2	Описание атрибутов	Инженер по знаниям эксперт	«Редактор атрибутов БЗ»	Заполнение файлов ATRidAbCdIfGh,ATRsetAbCdIfGh,
3	Описание продукций	Инженер по знаниям эксперт	«Редактор правил БЗ»	Заполнение файлов RULTidAbCdIfGh,FANAbCdIfGh
4	Формирование базы объяснения	Инженер по знаниям	«Редактор процедур объяснения»	Заполнение файла EXPLAbCdIfGh
5	Отладка базы знаний	Инженер по знаниям	«Модуль отладки БЗ»	Сообщения об ошибках
6	Выбор метода вывода	Конечный пользователь, инженер по знаниям (для анализа БЗ)	«Интерфейсный модуль вывода»	Переход к выбранному методу вывода

Продолжение Табл.1

п\п	Операция, процесс	Субъект	Компонента ИАС	Результат
7	Ввод значений атрибутов	Конечный пользователь, инженер по знаниям (для анализа БЗ)	«Интерфейсный модуль вывода»	Данные в предрабочей, рабочей области и индексной таблице
8	Запуск процесса экспертизы	Конечный пользователь, инженер по знаниям (для анализа БЗ)	«Интерфейсный модуль вывода»	Переход к модулю вывода
9	Получение экспертного решения	Автоматически	«Модуль прямого вывода», «Модуль обратного вывода»	Протокол вывода
10	Объяснение решения	Для конечного пользователя, для инженера по знаниям (для анализа БЗ)	«Интерфейсный модуль объяснения», «Процессор объяснения»	Вывод текста объяснения решения

Продолжение Табл.1

п\п	Операция, процесс	Субъект	Компонента ИАС	Результат
11	Анализ полученного решения (решение устраивает/ решение не устраивает)	Инженер по знаниям (при анализе БЗ)	Компоненты ИАС не участвуют	Решение устраивает/ решение не устраивает
12	Выявление противоречий, если решение не устраивает	Инженер по знаниям (при знанализе БЗ)	Компоненты ИАС не участвуют	Нахождение или отсутствие некоторой компоненты БЗ
13	Синтез решений по устранению противоречий (при наличии)	Инженер по знаниям (при анализе БЗ)	Компоненты ИАС не участвуют	Решение удалить, изменить или добавить компоненту БЗ
14	Редактирование БЗ (пп.2,3,4)	Инженер по знаниям (при анализе БЗ)	«Редактор атрибутов БЗ», «Редактор правил БЗ», «Редактор процедур объяснения»	Изменения в файлах БЗ: ATRidAbCdIfGh, ATRsetAbCdIfGh, RULTdAbCdIfGh, FANAbCdIfGh, EXPLAbCdIfGh

Окончание Табл.1

п\п	Операция, процесс	Субъект	Компонента ИАС	Результат
15	Проверка устранения противоречий (пп.6, 7, 8, 9,10)	Инженер по знаниям (при анализе БЗ)	«Интерфейсный модуль вывода» «Модуль прямого вывода», «Модуль обратного вывода», «Процессор объяснения»	Вывод текста объяснения решения

Библиографический список

1. Анисов А.М. Свойства времени // Логические исследования. Вып. 8. – М.: Наука, 2001. – 320 с.

2. Беллман Р., Заде Л. Принятие решений в расплывчатых условиях. В кн.: Вопросы анализа и процедуры принятия решений. – М.: Мир, 1976. – С. 172-215.

3. Берштейн Л.С., Мелихов А.Н., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. – М.: Наука, 1990. – 272 с.

4. Берштейн Л.С., Боженюк А.В., Малышев Н.Г. Нечеткие модели для экспертных систем в САПР. – М.: Энергоатомиздат,1991. – 136 с.

5. Берштейн Л.С., Боженюк А.В., Сергеев Н.Е. Управление на плоскости динамическим объектом на основе нечетких правил вывода// Материалы всероссийской научной конференции «Управление и информационные технологии». т. 1– СПб., 2003. – С. 262-266.

6. Берштейн Л.С., Сергеев Н.Е. Использование составных и усеченных лингвистических переменных//Труды IEEEAIS’03. – М.: Изд-во Физико-математической литературы, 2003. – С. 130-137.

7. Берштейн Л.С., Сергеев Н.Е. Использование нечетких множеств (переменных) третьего порядка для мониторинга подвижного объекта. Труды

IEEEAIS’03. – М.: Изд-во Физико-математической литературы, 2003. – С.141-I46.
8. Борисов А.Н., Алексеев А.В., Крумберг и др. Модели принятия решений на основе лингвистической переменной. – Рига: Знание, 1982. – 256 с.
9. Борисов А.М. ,Алексеев А.В., Меркурьева Г.В. и др. Обработка нечёткой информации в системах принятия решений. – М.: Радио и связь, 1989. – 304 с.
10. Варосян С.О., Поспелов Д.А. Неметрическая пространственная логика //Известия АН СССР. Техническая кибернетика. – 1982 – № 5. – С.86-89.
11. Домбровский Б. Каким временем мы пользуемся? (Этическая концепция времени) // Философско-литературный журнал «Логос». – 2007. – № 2. – С.75-97.
12. Ежкова И.В., Поспелов Д.А. Принятие решений при нечетких основаниях: Универсальная шкала //Известия АН СССР. Серия «Техническая кибернетика» – 1977. – №6.
13. Заде Л.А. Основы нового подхода к анализу сложных систем и процессов принятия решений // Математика сегодня. – М.: Знание, 1974. – С. 5-49.
14. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976 – 165 с.
15. Заде Л.А. Размытые множества и их применение в распознавании образов и кластер-анализе. В кн.: Классификация и кластер. –М.: Мир, 1980. – С. 208-247.
16. Кандрашина Е.Ю., Литвинцева Л.В., Поспелов Д.А. Представление знаний о времени и пространстве в интеллектуальных системах. – М.: Наука, 1989. – 328 с.
17. Люгер Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. – 4-е изд.;пер. с англ. М.: Издательский дом «Вильямс, 2003. –863 с.
18. Пегат А. Нечеткое моделирование и управление; пер. с англ. –М.: Бином. Лаборатория знаний, 2009. – 798 с.

19. Попов Э. Искусственный интеллект: Справочник в 3-ч кн. Кн. 1. Системы общения и экспертные системы: справочник/под ред. Э.В. Попова. – М.: Радио и связь,1990. – 464 с.
20. Поспелов Д.А. Нечеткие множества в моделях управления и искусственного интеллекта / под ред. Д.А. Поспелова. – М.: Наука, 1986.
21. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – 2-е изд.; пер. с англ. – М.: Издательский дом “Вильямс”, 2006. – 1408 с.
22. Сергеев Н.Е. Нечеткие модели инструментальных двигательных действий оператора: монография. – Ростов-на-Дону: Изд-во Ростовского университета, 2004. – 136 с.
23.Сергеев Н.Е. Моделирование инструментальных двигательных действий оператора: монография.– Ростов-на-Дону: Изд-во Ростовского университета, 2004. – 94 с.
24. Сергеев Н.Е. Представление перемещения объектов в пространстве при помощи лингвистических переменных //Известия ТРТУ. – 2004. – С. 270-273.
25. Сергеев Н.Е., Целых Ю.А. Информационная система автоматического описания сцен по видеоизображениям // Известия ЮФУ. Технические науки. – 2009.– №3.. – С. 253-259.
26. Уэно Х., Кояма Т., Окамото Т., Мацуби Б., Исидзука М. Представление и использование знаний. – М.: Мир, 1989. – 220 с.
28. Шилов А.А. О Классификации графов. Организационное управление и искусственный интеллект // Труды института системного анализа Российской академии наук. – М.: Едиториал УРСС, 2003. – С. 395-420.
29. Ягер Р. Нечеткие множества и теория возможностей. Последние достижения. – М: Радио и связь, 1986.

Сведения об авторе: Сергеев Николай Евгеньевич, доктор технических наук, профессор кафедры вычислительной техники ИКТИБ ЮФУ.

Учебное издание

Сергеев Николай Евгеньевич

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА
Часть1

Учебное пособие

Ответственный за выпуск Сергеев Н.Е.
Редактор Надточий З.И.
Корректор Чиканенко Л.В.

Подписано к печати
Заказ № Тираж 40 экз
Формат 60х841\16.
Печ.л. – 8,1. Уч.-изд.л. – 8,0.

Издательство Южного федерального университета
344091, г. Ростов-на-Дону, пр. Стачки, 200/1.
Тел. (863) 2478051.
Отпечатано в Отделе полиграфической, корпоративной и сувенирной продукции
ИПК КИБИ МЕДИА ЦЕНТРА ЮФУ.
ГСП 17А, Таганрог, 28, Энгельса, 1.
Тел. (8634) 37171117, 371655.