А.В. ОСТРОУХ

ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

МОНОГРАФИЯ

А.В. Остроух

ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Монография

Красноярск – 2020

УДК 004.382.7:681.3
ББК 32.973.26:32.844
О79

Рецензенты:

Попов Д.И., д.т.н., профессор кафедры «Информационных технологий», "Сочинский государственный университет" (СГУ);

Варламов О.О., д.т.н., директор НИИ «МИВАР», профессор кафедры «Системы обработки информации и управления» МГТУ им Н. Э. Баумана»

Остроух, А.В.
О79 Введение в искусственный интеллект : монография / А.В. Остроух. – Красноярск: Научно-инновационный центр, 2020. – 250 с.

ISBN 978-5-907208-26-1
DOI: 10.12731/978-5-907208-26-1

В монографии изложены концептуальные основы и методы представления знаний в системах искусственного интеллекта. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, кадров высшей квалификации и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки 09.00.00 – «Информатика и вычислительная техника».

УДК 004.382.7:681.3
ББК 32.973.26:32.844

ISBN 978-5-907208-26-1

© А.В. Остроух, 2020
© МАДИ, 2020
© Научно-инновационный центр, 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 7

1. СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ИНЖЕНЕРИИ ЗНАНИЙ 9

1.1. Роль интеллектуальных систем и технологий в процессе решения трудноформализуемых задач 9

1.1.1. Понятие системы искусственного интеллекта 9

1.1.2. Направления исследований в области систем искусственного интеллекта 10

1.1.3. Классификация систем искусственного интеллекта 11

1.1.4. Значение систем искусственного интеллекта 21

1.2. Теоретические аспекты инженерии знаний и архитектура интеллектуальных информационных систем 23

1.2.1. Данные и знания 23

1.2.2. Свойства знаний 24

1.2.3. Классификация знаний 25

1.2.4. Базы знаний 30

1.2.5. Архитектура систем искусственного интеллекта 34

2. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ И РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ 36

2.1. Модели и методы представления знаний 36

2.1.1. Классификация моделей представления знаний 36

2.1.2. Неформальные (семантические) модели 38

2.1.3. Формальные модели представления знаний 42

2.2. Исчисление высказываний 44

2.3. Исчисление предикатов 52

2.3.1. Исчисление предикатов первого порядка 52

2.3.2. Интерпретация формул в логике предикатов первого порядка 56
2.3.3. Системы аксиом логики предикатов 58
2.3.4. Правила вывода в исчислении предикатов 58
2.4. Нормальные формы исчисления предикатов 60
2.4.1. Предваренные (пренексные) нормальные формы исчисления предикатов 60
2.4.2. Автоматизация доказательства в логике предикатов 62
2.5. Методы решения задач в интеллектуальных системах 66
2.5.1. Решение задач методом поиска в пространстве состояний 68
2.5.2. Решение задач методом редукции 71
2.5.3. Решение задач дедуктивного выбора 75
2.5.4. Решение задач, использующие немонотонные логики, вероятностные логики 75
2.6. Метод резолюций 77
2.6.1. Метод резолюций в исчислении высказываний 78
2.6.2. Метод резолюций в исчислении предикатов. Правило унификации в логике предикатов 80
2.6.3. Алгоритм унификации для нахождения наиболее общего унификатора 83
2.6.4. Алгоритм метода резолюций 84
2.7. Представление неопределенности знаний и данных 85
2.7.1. Источники неопределенности 85
2.7.2. Интеллектуальные системы и теория вероятностей 88
2.7.3. Сомнительность и возможность 100
2.7.4. Теория возможности 106
2.7.5. Неопределенное состояние проблемы неопределенности 108
3. ТЕХНОЛОГИИ ЭКСПЕРТНЫХ СИСТЕМ 110

3.1. Назначение, классификация и принципы построения экспертных систем 110
3.1.1. Назначение экспертных систем 110
3.1.2. Классификация экспертных систем 115
3.1.3. Структура экспертных систем 117
3.2. Разработка экспертных систем 123
3.2.1. Методология разработки экспертных систем 123
3.2.2. Этапы разработки экспертных систем 126
3.2.3. Методы поиска решений в экспертных системах 131
3.2.4. Трудности разработки экспертных систем 133
3.3. Взаимодействие инженера по знаниям (когнитолога) с экспертом 136
3.3.1. Представление знаний в экспертных системах 136
3.3.2. Уровни представления и уровни детальности 140
3.3.3. Организация знаний в рабочей системе 142
3.3.4. Организация знаний в базе данных 143
3.3. Инструментальные средства проектирования и разработки экспертных систем 146
3.3.1. Уровень используемого языка 146
3.3.2. Парадигмы программирования и механизмы реализации 148
3.3.3. Способ представления знаний 150
3.3.4. Механизмы вывода и моделирования 150
3.3.5. Средства приобретения знаний 152
3.3.6. Экспертные системы реального времени 153
Видеоролики по главе 166
4. НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ 167
4.1. Нейробионика и нейрокомпьютеры 167
4.1.1. Основы нейробионики 167
4.1.2. Нейронные сети 169

4.1.3. Нейрокомпьютеры 171
4.2. Искусственные нейронные сети 176
4.2.1. Базовая искусственная модель 176
4.2.2. Классификация искусственных нейронных сетей 183
4.2.3. Задачи, решаемые нейронными сетями 185
4.2.4. Однослойные искусственные нейронные сети 187
4.2.5. Многослойные нейронные сети 189
4.3. Модели нейронных сетей 196
4.3.1. Вероятностная нейронная сеть 196
4.3.2. Обобщенно-регрессионная нейронная сеть 200
4.3.3. Линейная сеть 202
4.3.4. Сеть Кохонена 203
4.4. Нейронные сети глубокого обучения 207
4.4.1. Контролируемое и неконтролируемое обучение 207
4.4.2. Глубокое обучение 208
4.4.3. Обучение глубокой сети 212
4.5. Системы распознавания образов и машинного зрения 214
4.5.1. Основные принципы и целостность восприятия 214
4.5.2. Распознавание символов 218
4.5.3. Распознавание рукописных текстов 227
ЗАКЛЮЧЕНИЕ 229
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 230

ВВЕДЕНИЕ

Термин «искусственный интеллект» (artificial intelligence) был предложен в 1956 году. Слово intelligence означает «умение рассуждать разумно», а вовсе не «интеллект», для которого есть термин intellect.

Искусственный интеллект (ИИ) – это одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными, задачи, общаясь с компьютером на ограниченном подмножестве естественного языка.

ИИ занимается изучением разумного поведения (у людей, животных и машин) и пытается найти способы моделирования подобного поведения в любом типе искусственно созданного механизма. Несмотря на то что термину больше полувека, единого определения его не существует. Разные исследователи по-разному определяют эту науку, в зависимости от своего взгляда на нее, и работают над созданием систем, которые:

- думают подобно людям;
- думают рационально;
- действуют подобно людям;
- действуют рационально.

Синтезируя десятки определений искусственного интеллекта из различных источников, в качестве рабочего определения можно предложить следующее.

При воссоздании разумных рассуждений и действий возникают определенные трудности. Во-первых, в большинстве случаев, выполняя какие-то действия, человек не осознает, как это делает, не известен точный способ, метод или алгоритм понимания текста, распознавания лиц, доказательства теорем, решения задач, сочинения стихов и т.д. Во-вторых,

на современном уровне развития компьютер слишком далек от человеческого уровня компетентности и работает по другим принципам.

Искусственный интеллект всегда был междисциплинарной наукой, являясь одновременно и наукой, и искусством, и техникой, и психологией. Методы искусственного интеллекта разнообразны. Они активно заимствуются из других наук, адаптируются и изменяются под решаемую задачу. Для создания интеллектуальной системы необходимо привлекать специалистов из прикладной области, поэтому в рамках искусственного интеллекта сотрудничают лингвисты, нейрофизиологи, психологи, экономисты, информатики, программисты и т.д.

1. СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ИНЖЕНЕРИИ ЗНАНИЙ

1.1. РОЛЬ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ И ТЕХНОЛОГИЙ В ПРОЦЕССЕ РЕШЕНИЯ ТРУДНОФОРМАЛИЗУЕМЫХ ЗАДАЧ

1.1.1. Понятие системы искусственного интеллекта

Система искусственного интеллекта (СИИ) – это компьютерная модель интеллектуальных возможностей человека в целенаправленном поиске, анализе и синтезе текущей информации об окружающей действительности для получения о ней новых знаний и решения на этой основе различных жизненно важных задач.

Исследователи, работающие в этом направлении, надеются достичь такого понимания механизмов интеллекта, при котором можно будет составлять компьютерные программы с человеческим или более высоким уровнем интеллекта. Общий подход состоит в разработке методов решения задач, для которых отсутствуют формальные алгоритмы: понимание естественного языка, обучение, доказательство теорем, распознавание сложных образов и т.д. Теоретические исследования направлены на изучение интеллектуальных процессов и создание соответствующих математических моделей. Экспериментальные работы ведутся путем составления компьютерных программ и создания машин, решающих частные интеллектуальные задачи или разумно ведущих себя в заданной ситуации. Систематические исследования в области искусственного интеллекта начались лишь с появлением цифрового компьютера.

Первая научная статья по искусственному интеллекту была опубликована в 1950 А. Тьюрингом. Первые исследования в области ИИ были направлены на решение проблем и разработку систем символьных вычислений. В 60-х годах это направление привлекло интерес Министерства обороны США: американские военные начали обучать

компьютеры имитировать мыслительную деятельность человека. Например, Управление перспективных исследовательских проектов Министерства обороны США (DARPA) выполнило в 70-х годах ряд проектов по созданию виртуальных уличных карт. И специалистам DARPA удалось создать интеллектуальных личных помощников в 2003 году, задолго до того, как появились Siri, Alexa и Cortana.

Эти работы стали основой для принципов автоматизации и формальной логики рассуждений, которые используются в современных компьютерах, в частности, в системах для поддержки принятия решений и умных поисковых системах, призванных дополнять и приумножать возможности человека.

1.1.2. Направления исследований в области систем искусственного интеллекта

Хотя в научно-фантастических фильмах и романах ИИ зачастую изображают в виде человекоподобных роботов, захватывающих власть над миром, на данном этапе развития технологии ИИ совсем не такие страшные и далеко не такие умные. Напротив, развитие искусственного интеллекта позволяет этим технологиям приносить реальную пользу во всех отраслях экономики. Развитие интеллектуальных систем на современном этапе идет в соответствии с тремя направлениями исследований.

Первое направление объектом исследований рассматривает структуру и механизмы работы мозга человека, а конечной целью - раскрытие тайн мышления. Необходимыми этапами исследований в этом направлении являются построение моделей интеллектуальной деятельности на основе психофизиологических данных.

Второе направление в качестве объекта исследования рассматривает искусственную интеллектуальную систему. Здесь речь идет о

моделировании интеллектуальной деятельности с помощью вычислительных машин. Целью работ в этом направлении является создание программного обеспечения, позволяющего решать некоторые виды интеллектуальных задач так же, как их решил бы человек.

Третье направление ориентировано на создание человеко-машинных, или, как еще говорят – интерактивных, интеллектуальных систем. Важнейшими проблемами в этих исследованиях является организация **семантически безупречного диалога между человеком и такой системой**.

1.1.3. Классификация систем искусственного интеллекта

Для интеллектуальных информационных систем характерны следующие признаки:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению;
- адаптивность.

Коммуникативные способности СИИ характеризуют способ взаимодействия (интерфейса) конечного пользователя с системой, в частности возможность формулирования произвольного запроса в диалоге с ИИС на языке, максимально приближенном к естественному.

Сложные плохо формализуемые задачи – это задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Способность к самообучению – это возможность автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

Адаптивность – способность к развитию системы в соответствии с объективными изменениями модели проблемной области.

В соответствии с перечисленными признаками ИИС делятся на (данная классификация – одна из возможных) (рисунок 1.1):

– системы с коммутативными способностями (с интеллектуальным интерфейсом);
– экспертные системы (системы для решения сложных задач);
– самообучающиеся системы (системы, способные к самообучению);

адаптивные системы (адаптивные информационные системы).

Рисунок 1.1 – Классификация интеллектуальных информационных систем по типам систем

Интеллектуальные базы данных отличаются от обычных баз данных возможностью выборки по запросу необходимой информации,

которая может явно не храниться, а выводиться из имеющейся в базе данных.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое.

Естественно-языковой интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом. В более широком плане сказанное распространяется и на поиск мультимедийной информации, включающей, помимо текстовой, и цифровую информацию.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует, и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации. Например, состояние сложного управляемого объекта отображается в виде человеческого лица, на котором каждая черта отвечает за какой-либо параметр, а общее выражение лица дает интегрированную характеристику ситуации. Системы когнитивной графики широко используются также в обучающих и тренажерных системах на основе использования принципов виртуальной реальности, когда графические образы моделируют ситуации, в которых обучаемому необходимо принимать решения и выполнять определенные действия.

Экспертные системы предназначены для решения задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области.

Многоагентные системы – это динамические системы, для которых характерна интеграция в базе знаний нескольких разнородных источников

знаний, обменивающихся между собой получаемыми результатами на динамической основе.

Для многоагентных систем характерны следующие особенности:

- проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий;
- распределенное решение проблем, которые разбиваются на параллельно решаемые подпроблемы, соответствующие самостоятельным источникам знаний;
- применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы;
- обработка больших массивов данных, содержащихся в базе данных;
- использование различных математических моделей и внешних процедур, хранимых в базе моделей;
- способность прерывания решения задач в связи с необходимостью получения дополнительных данных и знаний от пользователей, моделей, параллельно решаемых подпроблем.

В основе ***самообучающихся систем*** лежат методы автоматической классификации примеров ситуаций реальной практики.

Характерными признаками самообучающихся систем являются:

- самообучающиеся системы «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);
- самообучающиеся системы «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

Индуктивные системы используют обобщение примеров по принципу от частного к общему. Процесс классификации примеров осуществляется следующим образом:

1. Выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например, в соответствии с максимальным числом получаемых подмножеств примеров).
2. По значению выбранного признака множество примеров разбивается на подмножества.
3. Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу.
4. Если какое-то подмножество примеров принадлежит одному подклассу, то есть у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается (при этом остальные признаки классификации не рассматриваются).
5. Для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается, начиная с пункта 1 (каждое подмножество примеров становится классифицируемым множеством).

Нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессоров. Каждый процессор такой сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам.

В экспертных системах, ***основанных на прецедентах*** (аналогиях), база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты.

Поиск решения проблемы в экспертных системах, основанных на прецедентах, сводится к поиску по аналогии (то есть абдуктивный вывод от частного к частному).

В отличие от интеллектуальной базы данных, ***информационное хранилище*** представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного ситуационного анализа данных (реализации OLAP-технологии).

Типичными задачами оперативного ситуационного анализа являются:

- определение профиля потребителей конкретных объектов хранения;
- предсказание изменений объектов хранения во времени;
- анализ зависимостей признаков ситуаций (корреляционный анализ).

Адаптивная информационная система – это информационная система, которая изменяет свою структуру в соответствии с изменением модели проблемной области.

При этом:

- адаптивная информационная система должна в каждый момент времени адекватно поддерживать организацию бизнес-процессов;
- адаптивная информационная система должна проводить адаптацию всякий раз, как возникает потребность в реорганизации бизнес-процессов;
- реконструкция информационной системы должна проводиться быстро и с минимальными затратами.

Ядром адаптивной информационной системы является постоянно развиваемая модель проблемной области (предприятия), поддерживаемая в специальной базе знаний – репозитории. На основе ядра осуществляется генерация или конфигурация программного обеспечения. Таким образом,

проектирование и адаптация ИС сводится, прежде всего, к построению модели проблемной области и ее своевременной корректировке.

Так как нет общепринятого определения, четкую единую классификацию интеллектуальных информационных систем дать затруднительно.

Если рассматривать интеллектуальные информационные системы с точки зрения решаемой задачи, то можно выделить системы управления и справочные системы, системы компьютерной лингвистики, системы распознавания, игровые системы и системы создания интеллектуальных информационных систем (рисунок 1.2).

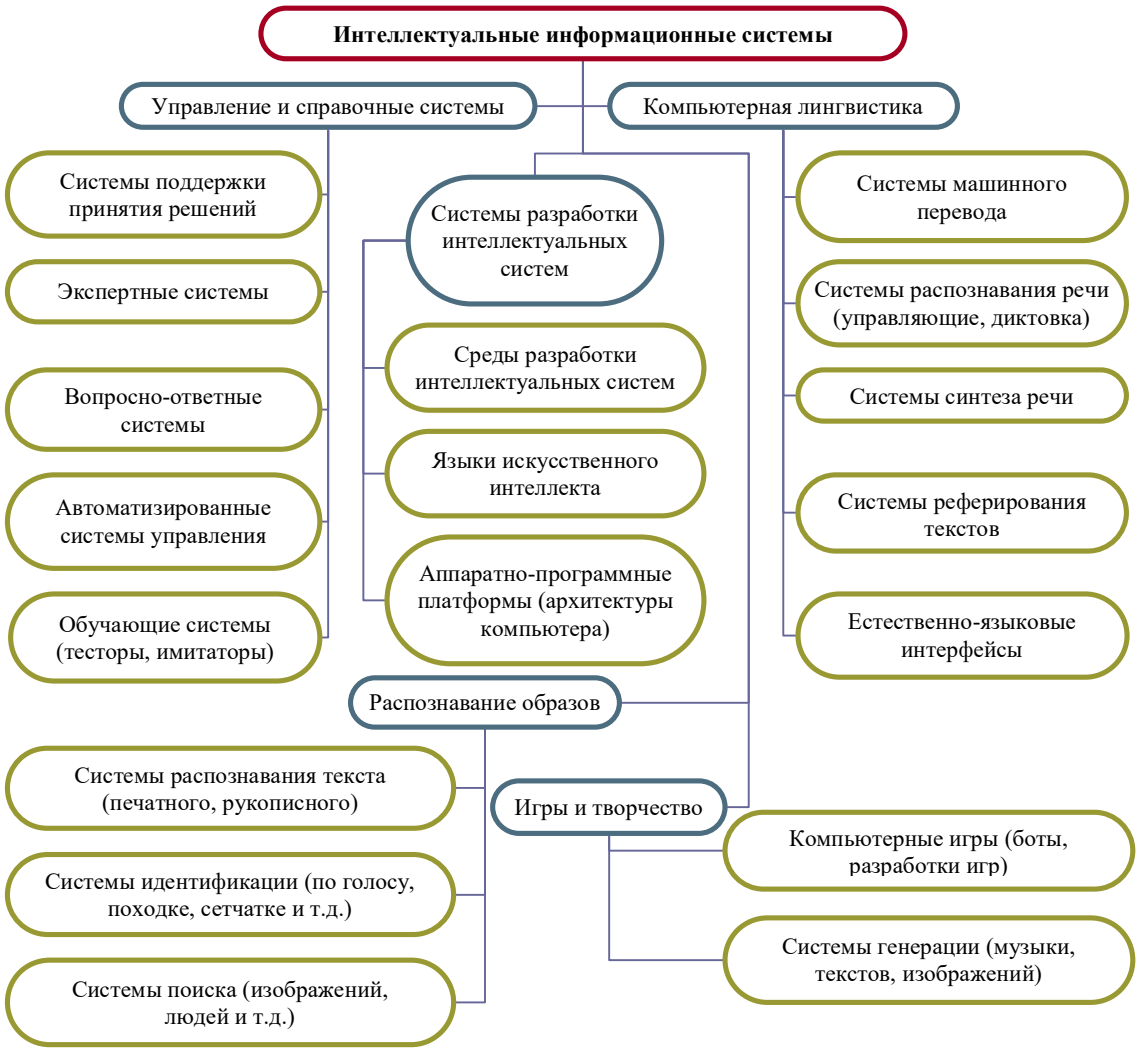

Рисунок 1.2 – Классификация интеллектуальных информационных систем по решаемым задачам

При этом системы могут решать не одну, а несколько задач или в процессе решения одной задачи решать и ряд других. Например, при обучении иностранному языку система может решать задачи распознавания речи обучаемого, тестировать, отвечать на вопросы, переводить тексты с одного языка на другой и поддерживать естественно-языковой интерфейс работы.

Если классифицировать интеллектуальные информационные системы по критерию «используемые методы», то они делятся на жесткие, мягкие и гибридные (рисунок 1.3).

Рисунок 1.3 – Классификация интеллектуальных информационных систем по методам

Мягкие вычисления (Soft Computing) – это сложная компьютерная методология, основанная на нечеткой логике, генетических вычислениях, нейрокомпьютинге и вероятностных вычислениях.

Жесткие вычисления – традиционные компьютерные вычисления (не мягкие).

Гибридные системы – системы, использующие более чем одну компьютерную технологию (в случае интеллектуальных систем – технологии искусственного интеллекта).

Возможны и другие классификации, например, выделяют системы общего назначения и специализированные системы (рисунок 1.4).

Кроме того, эта схема отражает еще один вариант классификации по методам: системы, использующие методы представления знаний, самоорганизующиеся системы и системы, созданные с помощью эвристического программирования. Также в этой классификации системы генерации музыки отнесены к системам общения.

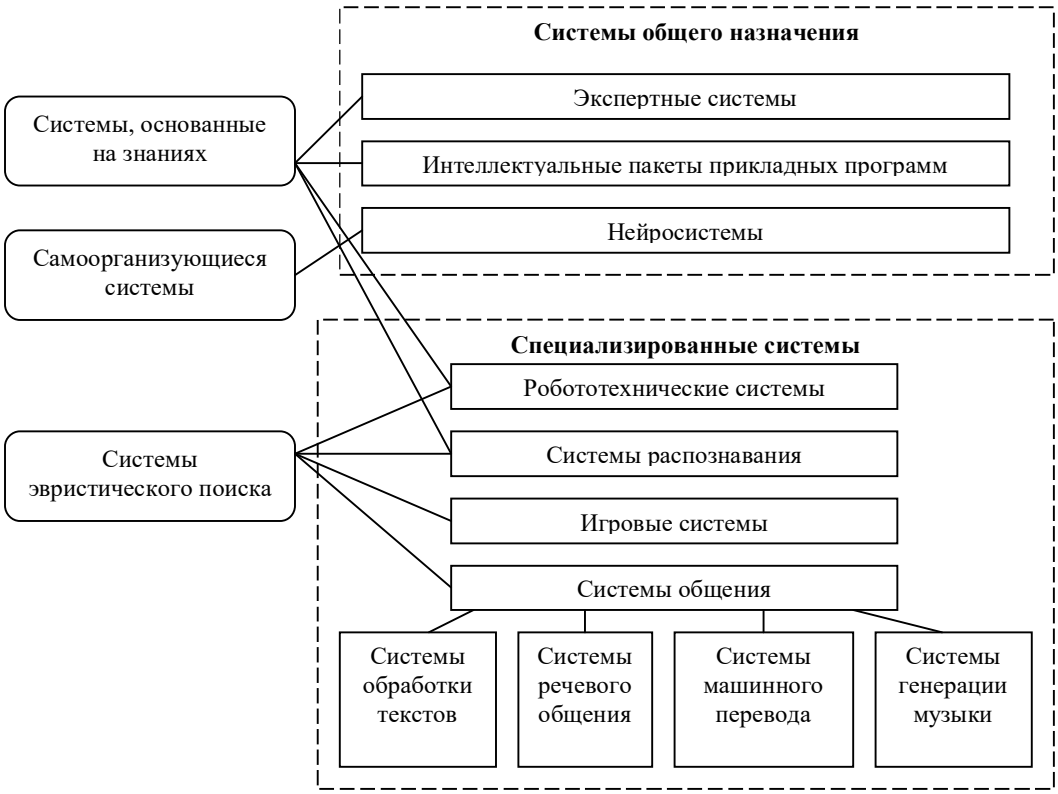

Рисунок 1.4 – Классификация интеллектуальных систем по назначению

К интеллектуальным системам общего назначения относятся системы, которые не только исполняют заданные процедуры, но на основе

метапроцедур поиска генерируют и исполняют процедуры решения новых конкретных задач.

Специализированные интеллектуальные системы выполняют решение фиксированного набора задач, предопределенного при проектировании системы.

Отсутствие четкой классификации также объясняется многообразием интеллектуальных задач и интеллектуальных методов, кроме того, искусственный интеллект – активно развивающаяся наука, в которой новые прикладные области осваиваются ежедневно.

1.1.4. Значение систем искусственного интеллекта

ИИ позволяет автоматизировать повторяющиеся процессы обучения и поиска за счет использования данных. Однако ИИ отличается от роботизации, в основе которой лежит применение аппаратных средств. Цель ИИ — не автоматизация ручного труда, а надежное и непрерывное выполнение многочисленных крупномасштабных компьютеризированных задач. Такая автоматизация требует участия человека для первоначальной настройки системы и правильной постановки вопросов.

ИИ делает существующие продукты интеллектуальными. Как правило, технология ИИ не реализуется как отдельное приложение. Функционал ИИ интегрируется в имеющиеся продукты, позволяя усовершенствовать их, точно так же, как технология Siri была добавлена в устройства Apple нового поколения. Автоматизация, платформы для общения, боты и «умные» компьютеры в сочетании с большими объемами данных могут улучшить различные технологии, которые используются дома и в офисах: от систем анализа данных о безопасности до инструментов инвестиционного анализа.

ИИ адаптируется благодаря алгоритмам прогрессивного обучения, чтобы дальнейшее программирование осуществлялось на основе данных. ИИ обнаруживает в данных структуры и закономерности, которые позволяют алгоритму освоить определенный навык: алгоритм становится классификатором или предикатором. Таким образом, по тому же принципу, по которому алгоритм осваивает игру в шахматы, он может научиться предлагать подходящие продукты онлайн. При этом модели адаптируются по мере поступления новых данных. Обратное распространение — это метод, который обеспечивает корректировку модели посредством обучения на базе новых данных, если первоначальный ответ оказывается неверным.

ИИ осуществляет более глубокий анализ больших объемов данных с помощью нейросетей со множеством скрытых уровней. Несколько лет назад создание системы обнаружения мошенничества с пятью скрытыми уровнями было практически невозможным. Все изменилось с колоссальным ростом вычислительных мощностей и появлением «больших данных». Для моделей глубокого обучения необходимо огромное количество данных, так как именно на их основе они и обучаются. Поэтому чем больше данных, тем точнее модели.

Глубинные нейросети позволяют ИИ достичь беспрецедентного уровня точности. К примеру, работа с Alexa, поисковой системой Google Search и сервисом Google Photos осуществляется на базе глубокого обучения, и чем чаще мы используем эти инструменты, тем эффективнее они становятся. В области здравоохранения диагностика раковых опухолей на снимках МРТ с помощью технологий ИИ (глубокое обучение, классификация изображений, распознавание объектов) по точности не уступает заключениям высококвалифицированных рентгенологов.

ИИ позволяет извлечь максимальную пользу из данных. С появлением самообучающихся алгоритмов сами данные становятся

объектом интеллектуальной собственности. Данные содержат в себе нужные ответы — нужно лишь найти их при помощи технологий ИИ. Поскольку сейчас данные играют гораздо более важную роль, чем когда-либо ранее, они могут обеспечить конкурентное преимущество. При использовании одинаковых технологий в конкурентной среде выиграет тот, у кого наиболее точные данные.

1.2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ИНЖЕНЕРИИ ЗНАНИЙ И АРХИТЕКТУРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

1.2.1. Данные и знания

Данные – это информация, полученная в результате наблюдений или измерений отдельных свойств (атрибутов), характеризующих объекты, процессы и явления предметной области.

Знание – форма существования и систематизации результатов познавательной деятельности человека. Знание помогает людям рационально организовывать свою деятельность и решать различные проблемы, возникающие в ее процессе; субъективный образ объективной реальности, то есть адекватное отражение внешнего и внутреннего мира в сознании человека в форме представлений, понятий, суждений, теорий.

Знание *(в широком смысле)* – совокупность понятий, теоретических построений и представлений.

Знание *(в узком смысле)* – признак определенного объема информации, определяющий ее статус и отделяющий от всей прочей информации по критерию способности к решению поставленной задачи.

Знание *(с точки зрения представления знаний в интеллектуальных системах)* – это связи и закономерности предметной области (принципы, модели, законы), полученные в результате

практической деятельности и профессионального опыта, позволяющего специалистам ставить и решать задачи в данной области.

1.2.2. Свойства знаний

Знания от данных отличаются рядом **свойств:**

- внутренняя интерпретируемость;
- структурированность;
- связность;
- семантическая метрика;
- активность.

Внутренняя интерпретируемость. Данные, хранящиеся в памяти или на внешних носителях, лишены имен, таким образом, отсутствует возможность их однозначной идентификации системой. Данные может идентифицировать лишь программа, извлекающая их по определенному алгоритму. При переходе к знаниям в память вводится дополнительная информация (атрибуты: фамилия, год рождения, специальность, стаж). Атрибуты могут играть роль имен. По ним можно осуществлять поиск нужной информации.

Структурированность. Информационные единицы должны обладать гибкой структурой. Иначе говоря, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа «часть – целое», «род – вид» или «элемент – класс».

Связность. Между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Семантика отношений может носить декларативный или процедурный характер. Например, две и более информационные единицы могут быть связаны отношением «одновременно», две информационные единицы – отношением «причина – следствие» или «быть рядом».

Семантическая метрика. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее их ситуационную близость, то есть силу ассоциативной связи. Его можно было бы назвать отношением релевантности для информационных единиц. Оно дает возможность выделять в информационной базе некоторые типовые ситуации (например, «покупка», «регулирование движения на перекрестке»). Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Активность. Все вычислительные процессы инициируются командами, а данные используются этими командами лишь в случае необходимости. Иначе говоря, данные пассивны, а команды активны.

Знания позволяют адаптироваться и действовать в реальной действительности. Существует огромное множество различных знаний, начиная от рецепта приготовления омлета до квантовой физики.

1.2.3. Классификация знаний

Знания можно классифицировать по нескольким критериям (рисунок 1.5).

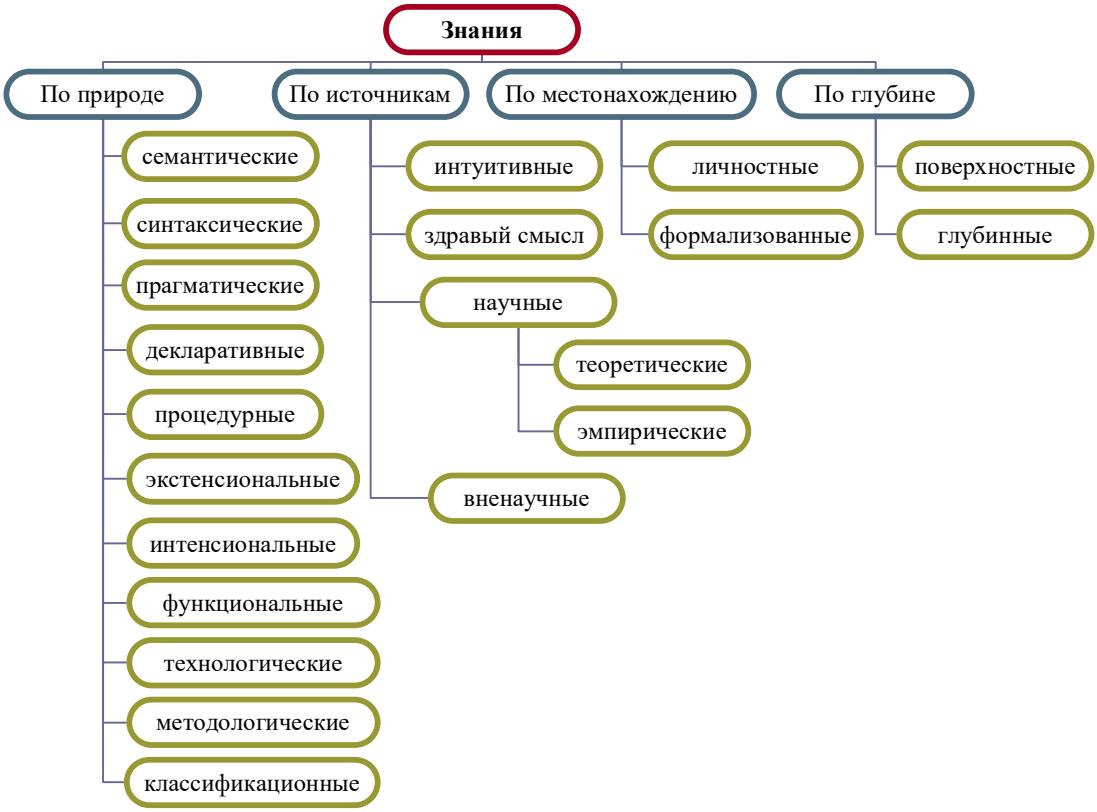

Рисунок 1.5 – Классификация знаний

Знание *синтаксического* типа характеризует синтаксическую структуру потока информации, которая не зависит от смысла и содержания используемых при этом понятий, то есть интеллектуальную систему не образует.

Семантическое знание рассматривается как структура, образующая текущий контекст. Оно содержит информацию, непосредственно связанную с текущими значениями и смыслом описываемых понятий, и предопределяет состояние связей данных в информационной базе.

Прагматическое знание предопределяет наиболее вероятные связи, описывающие данные с точки зрения решаемой задачи (обобщенный или «объективный» контекст), например, с учетом действующих в данной задаче специфических критериев и соглашений.

Декларативные знания содержат в себе представление о структуре понятий. Эти знания приближены к данным, фактам. Например, высшее

учебное заведение есть совокупность факультетов, а каждый факультет в свою очередь есть совокупность кафедр.

Процедурные знания имеют активную природу. Они определяют представления о средствах и путях получения новых знаний, проверке знаний. Это алгоритмы разного рода. С развитием информатики все большая часть знаний сосредотачивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных.

Существенными для понимания природы знаний являются способы определения понятий. Один из широко применяемых способов основан на идее интенсионала и экстенсионала.

Интенсионал понятия – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств.

Экстенсионал понятия – это определение понятия через перечисление его конкретных примеров, то есть понятий более низкого уровня абстракции. Интенсионалы формируют знания об объектах, в то время как экстенсионалы объединяют данные.

Отсюда *интенсиональные* знания – это знания о предметной области, которые отражают факты, закономерности, свойства и характеристики, справедливые для любых ситуаций, которые могут возникнуть в этой предметной области.

Экстенсиональные знания – это знания о предметной области, отражающие факты, закономерности, свойства и характеристики, типичные для конкретных ситуаций или классов однотипных ситуаций, которые могут возникнуть в этой области.

Функциональные знания – это знания о выполняемых функциях отдельных предметов и о применении их в реальной действительности.

Технологические знания – специализированные знания, обеспечивающие поддержание технологических параметров производства; производственный опыт и навыки, используемые при решении повседневных производственных вопросов. Это может быть знание последовательности операций или знание технологической цепочки, позволяющие достигать поставленные цели в соответствии с принятой технологией.

Методологические знания – знания о методах преобразования действительности, научные знания о построении эффективной деятельности. К методологическим знаниям относят знание целей, форм и направлений развития теории, методов и способов эффективного преобразования практики.

Классификационные знания применяются главным образом в науке, являются обобщенными, системными знаниями. Пример – система элементов Д. И. Менделеева.

Интуиция – это вид знания, специфика которого обусловлена способом его приобретения. Это знание, не нуждающееся в доказательстве и воспринимаемое как достоверное. По способу получения интуиция – это прямое усмотрение объективной связи вещей, не опирающееся на доказательство (интуиция есть усмотрение внутренним зрением; от лат. intueri – созерцать).

Под *здравым смыслом* понимают знания, позволяющие принимать правильные решения и делать правильные предположения, основываясь на логическом мышлении и накопленном опыте. В этом значении термин зачастую акцентирует внимание на способности человеческого разума противостоять предрассудкам, заблуждениям, мистификациям.

Научные знания в любом случае должны быть основанными на эмпирической или теоретической доказательной основе.

Теоретические знания – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов. Теоретический уровень научного знания предполагает установление законов, дающих возможность идеализированного восприятия, описания и объяснения эмпирических ситуаций, то есть познания сущности явлений. Теоретические законы имеют более строгий, формальный характер по сравнению с эмпирическими. Термины описания теоретического знания относятся к идеализированным, абстрактным объектам. Подобные объекты невозможно подвергнуть непосредственной экспериментальной проверке.

Эмпирические знания получают в результате применения эмпирических методов познания: наблюдения, измерения, эксперимента. Это знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. Эмпирические знания, как правило, констатируют качественные и количественные характеристики объектов и явлений. Эмпирические законы часто носят вероятностный характер и не являются строгими.

Вненаучные знания могут быть различными. *Паранормальные* знания – знания, несовместимые с имеющимся гносеологическим стандартом. Широкий класс *паранаучного* (пара от греч. около, при) знания включает в себя учения или размышления о феноменах, объяснение которых не является убедительным с точки зрения критериев научности. *Лженаучные* знания – сознательно эксплуатирующие домыслы и предрассудки. В качестве симптомов лженауки выделяют малограмотный пафос, принципиальную нетерпимость к опровергающим доводам, а также претенциозность. Лженаучные знания сосуществуют с научными знаниями.

Личностные (неявные, скрытые) знания – это знания людей, полученные из практики и опыта.

Формализованные (явные) знания – знания, содержащиеся в документах, на компакт-дисках, в персональных компьютерах, в Интернете, в базах знаний, в экспертных системах. Формализованные знания объективизируются знаковыми средствами языка, охватывают те знания, о которых мы знаем, их можно записать, сообщить другим.

1.2.4. Базы знаний

Перечисленные ниже пять особенностей информационных единиц определяют ту грань, за которой данные превращаются в знания, а базы данных перерастают в **базы знаний.**

База знаний (БЗ) – основа любой интеллектуальной системы, где знания описаны на некотором языке представления знаний, приближенном к естественному. Сегодня знания приобрели чисто декларативную форму, то есть знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному языку и понятных неспециалистам.

Внутренняя интерпретируемость. Каждая информационная единица должна иметь уникальное имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя упомянуто. Когда данные, хранящиеся в памяти, были лишены имен, то отсутствовала возможность их идентификации системой. Данные могла идентифицировать лишь программа, извлекающая их из памяти по указанию программиста, написавшего программу. Что скрывается за тем или иным двоичным кодом машинного слова, системе было неизвестно.

Таблица 1.1 – Данные о сотрудниках предприятия

Фамилия	Год рождения	Специальность	Стаж, число лет
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

Если, например, в память компьютера нужно было записать сведения о сотрудниках учреждения, представленные в таблице 1.1, то без внутренней интерпретации в память компьютера была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные таблице 1.1 для решения возникающих у него задач.

При переходе к знаниям в память компьютера вводится информация о некоторой протоструктуре информационных единиц. В рассматриваемом примере она представляет собой специальное машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях, годах рождения, специальностях и стажах. При этом должны быть заданы специальные словари, в которых перечислены имеющиеся в памяти системы фамилии, года рождения, специальности и продолжительности стажа. Все эти атрибуты могут играть роль имен для тех машинных слов, которые соответствуют строкам таблицы. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром протоструктуры. В настоящее время СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, хранящихся в базе данных.

Структурированность. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложимость одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".

Связность. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать отношения структуризации, функциональные отношения, каузальные отношения и семантические отношения. С помощью первых задаются иерархии информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно-следственные связи, четвертые соответствуют всем остальным отношениям.

Между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц

из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют ввести общую модель представления знаний, которую можно назвать семантической сетью, представляющей собой иерархическую сеть, в вершинах которой находятся информационные единицы. Эти единицы снабжены индивидуальными именами. Дуги семантической сети соответствуют различным связям между информационными единицами.

Семантическая метрика. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами. Его можно было бы назвать отношением релевантности для информационных единиц. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации. Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Активность. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Как и у человека, в ИС актуализации тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы.

Совокупность средств, обеспечивающих работу с знаниями, образует **Систему Управления Базой Знаний (СУБЗ)**. В настоящее время не существует баз знаний, в которых в полной мере были бы реализованы

внутренняя интерпретируемость, структуризация, связность, введена семантическая мера и обеспечена активность знаний.

1.2.5. Архитектура систем искусственного интеллекта

Архитектура систем искусственного интеллекта включает три комплекса вычислительных средств (рисунок 1.6).

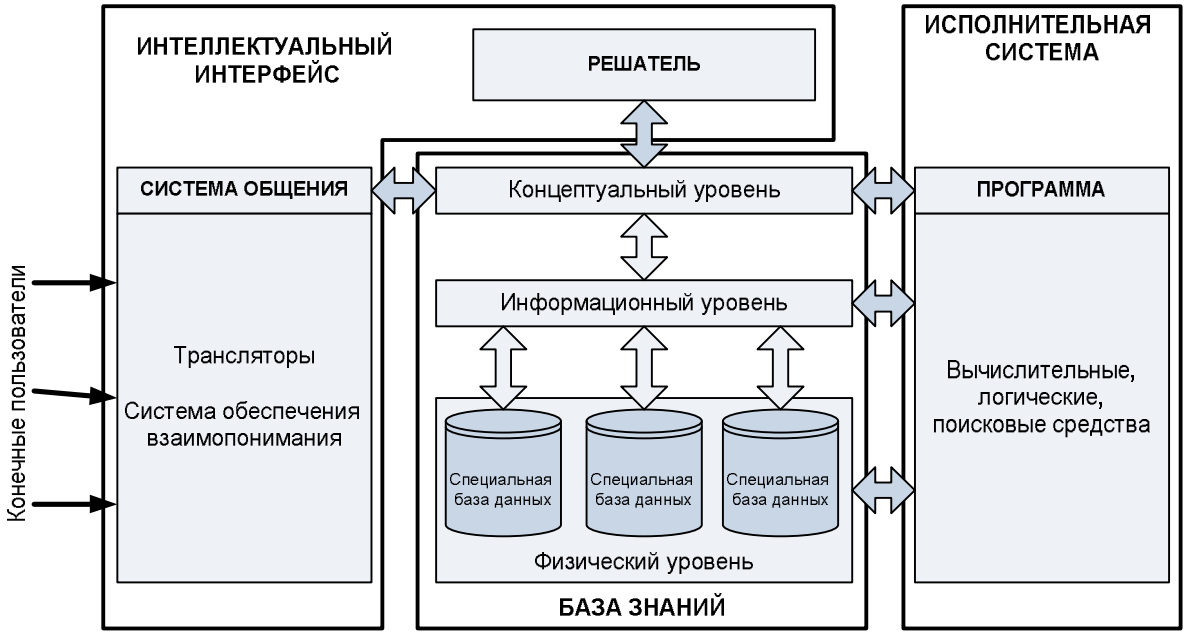

Рисунок 1.6 – Архитектура интеллектуальных систем

Первый комплекс представляет собой совокупность средств, выполняющих программы (исполнительную систему), спроектированных с позиций эффективного решения задач, имеет в ряде случаев проблемную ориентацию.

Второй комплекс представляет собой совокупность средств интеллектуального интерфейса, имеющих гибкую структуру, которая обеспечивает возможность адаптации в широком спектре интересов конечных пользователей.

Третьим комплексом средств, с помощью которых организуется взаимодействие первых двух, является база знаний, обеспечивающая использование вычислительными средствами первых двух комплексов

целостной и независимой от обрабатывающих программ системы знаний о проблемной среде. Исполнительная система объединяет всю совокупность средств, обеспечивающих выполнение сформированной программы.

Интеллектуальный интерфейс – система программных и аппаратных средств, обеспечивающих для конечного пользователя использование компьютера для решения задач, которые возникают в среде его профессиональной деятельности либо без посредников, либо с незначительной их помощью.

БЗ занимает центральное положение по отношению к остальным компонентам вычислительной системы. В целом, через БЗ осуществляется интеграция средств вычислительной системы, участвующих в решении задач.

2. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ И РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

2.1. МОДЕЛИ И МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

2.1.1. Классификация моделей представления знаний

Совокупность знаний, нужных для принятия решений, принято называть предметной областью или **знаниями о предметной области**. В любой предметной области есть свои понятия и связи между ними, своя терминология, свои законы, связывающие между собой объекты данных предметной области, свои процессы и события. Кроме того, каждая предметная область имеет свои методы решения задач. Решая задачи такого вида на компьютере, используют информационные системы, ядром которых является база знаний, содержащая основные характеристики предметных областей.

Базы знаний базируются на **моделях представления знаний**, подобно базам данных, которые основаны на моделях представления данных (иерархической, сетевой, реляционной, постреляционной и т.д.).

Существуют два типа моделей представления знаний (ПЗ):

– формальные модели ПЗ;

– неформальные (семантические, реляционные) модели ПЗ.

Очевидно, все методы представления знаний, которые рассмотрены выше, включая продукции (это система правил, на которых основана продукционная модель представления знаний), относятся к неформальным моделям. В отличие от формальных моделей, в основе которых лежит строгая математическая теория, неформальные модели такой теории не придерживаются. Каждая неформальная модель годится только для конкретной предметной области и поэтому не обладает универсальностью, которая присуща моделям формальным. Логический вывод - основная

операция в СИИ - в формальных системах строг и корректен, поскольку подчинен жестким аксиоматическим правилам. Вывод в неформальных системах во многом определяется самим исследователем, который и отвечает за его корректность.

При представлении знаний в памяти интеллектуальной системы традиционные языки, основанные на численном представлении данных, являются неэффективными. Для этого используются специальные языки представления знаний, основанные на символьном представлении данных. Они делятся на типы по формальным моделям представления знаний. Наиболее часто используется на практике классификация моделей представления знаний, приведенная на рисунке 2.1, где модели представления знании делятся на детерминированные (жесткие) и мягкие.

Рисунок 2.1 – Модели представления знаний

Детерминированные модели включают в себя фреймы, логико-алгебраические модели, семантические сети и продукционные модели. Мягкие модели включают в себя нечеткие системы, нейронные сети, эволюционные модели, гибридные системы.

С моделированием знаний непосредственно связана проблема выбора языка представления. В целях классификации моделей представления знаний выделяется девять ключевых требований к моделям знаний:

– общность (универсальность);
– наглядность представления знаний;
– однородность;
– реализация в модели свойства активности знаний;
– открытость;
– возможность отражения структурных отношений объектов предметной области;
– наличие механизма «проецирования» знаний на систему семантических шкал;
– возможность оперирования нечеткими знаниями;
– использование многоуровневых представлений (данные, модели, метамодели, метаметамодели и т.д.).

Модели представления знаний не удовлетворяют полностью этим требованиям, чем и объясняется их многообразие и активное развитие данного направления.

2.1.2. Неформальные (семантические) модели

Каждому из методов ПЗ соответствует свой способ описания знаний.

Логические модели. В основе моделей такого типа лежит *формальная система*, задаваемая четверкой вида: $M = \langle T, P, A, B \rangle$. Множество T есть *множество базовых элементов* различной природы, например, слов из некоторого ограниченного словаря, деталей детского конструктора, входящих в состав некоторого набора и т.п. Важно, что для множества T существует некоторый способ определения принадлежности или непринадлежности произвольного элемента к этому множеству. Процедура такой проверки может быть любой, но за конечное число шагов она должна давать положительный или отрицательный ответ на вопрос, является ли x элементом множества T. Обозначим эту процедуру П(T).

Множество *P* есть множество *синтаксических правил*. С их помощью из элементов *T* образуют *синтаксически правильные совокупности*. Например, из слов ограниченного словаря строятся синтаксически правильные фразы, из деталей детского конструктора с помощью гаек и болтов собираются новые конструкции. Декларируется существование процедуры П(*P*), с помощью которой за конечное число шагов можно получить ответ на вопрос, является ли совокупность *X* синтаксически правильной.

В множестве синтаксически правильных совокупностей выделяется некоторое подмножество *A*. Элементы *A* называются *аксиомами*. Как и для других составляющих формальной системы, должна существовать процедура П(*A*), с помощью которой для любой синтаксически правильной совокупности можно получить ответ на вопрос о принадлежности ее к множеству *A*.

Множество *B* есть множество *правил вывода*. Применяя их к элементам *A*, можно получать новые синтаксически правильные совокупности, к которым снова можно применять правила из *B*. Так формируется *множество выводимых* в данной формальной системе *совокупностей*. Если имеется процедура П(*B*), с помощью которой можно определить для любой синтаксически правильной совокупности, является ли она выводимой, то соответствующая формальная система называется *разрешимой*. Это показывает, что именно правило вывода является наиболее сложной составляющей формальной системы.

Для знаний, входящих в базу знаний, можно считать, что множество *A* образуют все информационные единицы, которые введены в базу знаний извне, а с помощью правил вывода из них выводятся новые *производные знания*. Другими словами, формальная система представляет собой генератор порождения новых знаний, образующих множество *выводимых* в данной системе *знаний*. Это свойство логических моделей делает их

притягательными для использования в базах знаний. Оно позволяет хранить в базе лишь те знания, которые образуют множество *A*, а все остальные знания получать из них по правилам вывода.

2. *Сетевые модели*. В основе моделей этого типа лежит конструкция, названная ранее семантической сетью. Сетевые модели формально можно задать в виде $H = <I, C_1, C_2, ..., C_n, \Gamma>$. Здесь *I* есть множество информационных единиц; $C_1, C_2, ..., C_n$ - множество типов связей между информационными единицами. Отображение Г задает между информационными единицами, входящими в *I*, связи из заданного набора типов связей.

В зависимости от типов связей, используемых в модели, различают *классифицирующие сети, функциональные сети* и *сценарии*. В классифицирующих сетях используются отношения структуризации. Такие сети позволяют в базах знаний вводить разные иерархические отношения между информационными единицами. Функциональные сети характеризуются наличием функциональных отношений. Их часто называют *вычислительными моделями*, т.к. они позволяют описывать процедуры "вычислений" одних информационных единиц через другие. В сценариях используются каузальные отношения, а также отношения типов "средство - результат", "орудие - действие" и т.п. Если в сетевой модели допускаются связи различного типа, то ее обычно называют семантической сетью.

3. *Продукционные модели*. В моделях этого типа используются некоторые элементы логических и сетевых моделей. Из логических моделей заимствована идея правил вывода, которые здесь называются *продукциями*, а из сетевых моделей - описание знаний в виде семантической сети. В результате применения правил вывода к фрагментам сетевого описания происходит трансформация семантической сети за счет смены ее фрагментов, наращивания сети и исключения из нее

ненужных фрагментов. Таким образом, в продукционных моделях процедурная информация явно выделена и описывается иными средствами, чем декларативная информация. Вместо логического вывода, характерного для логических моделей, в продукционных моделях появляется *вывод на знаниях*.

4. *Фреймовые модели*. В отличие от моделей других типов во фреймовых моделях фиксируется жесткая структура информационных единиц, которая называется *протофреймом*. В общем виде она выглядит следующим образом:

(Имя фрейма:

Имя слота 1(значение слота 1)

Имя слота 2(значение слота 2)

. . . . . .

Имя слота К (значение слота К)).

Значением *слота* может быть практически что угодно (числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов). В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет во фреймовых представлениях реализовать "принцип матрешки".

При конкретизации фрейма ему и слотам присваиваются конкретные имена и происходит заполнение слотов. Таким образом, из протофреймов получаются *фреймы - экземпляры.* Переход от исходного протофрейма к фрейму - экземпляру может быть многошаговым, за счет постепенного уточнения значений слотов.

Например, структура таблицы 1.1, записанная в виде протофрейма, имеет вид

(Список работников:

Фамилия (значение слота 1);

Год рождения (значение слота 2);

Специальность (значение слота 3);

Стаж (значение слота 4)).

Если в качестве значений слотов использовать данные таблицы 1.1, то получится фрейм - экземпляр

(Список работников:

Фамилия (Попов - Сидоров - Иванов - Петров);

Год рождения (1965 - 1946 - 1925 - 1937);

Специальность (слесарь - токарь - токарь - сантехник);

Стаж (5 - 20 - 30 - 25)).

Связи между фреймами задаются значениями специального слота с именем "Связь". Часть специалистов по ИС считает, что нет необходимости специально выделять фреймовые модели в представлении знаний, т.к. в них объединены все основные особенности моделей остальных типов.

2.1.3. Формальные модели представления знаний

Интеллектуальная система в определенном смысле моделирует интеллектуальную деятельность человека и, в частности, - логику его рассуждений. В грубо упрощенной форме наши логические построения при этом сводятся к следующей схеме: из одной или нескольких посылок (которые считаются истинными) следует сделать "логически верное" заключение (вывод, следствие). Очевидно, для этого необходимо, чтобы и посылки, и заключение были представлены на понятном языке, адекватно отражающем предметную область, в которой проводится вывод. В обычной жизни это наш естественный язык общения, в математике, например, это язык определенных формул и т.п. Наличие же языка предполагает, во-первых, наличие алфавита (словаря), отображающего в символьной форме весь набор базовых понятий (элементов), с которыми

придется иметь дело и, во - вторых, набор синтаксических правил, на основе которых, пользуясь алфавитом, можно построить определенные выражения.

Логические выражения, построенные в данном языке, могут быть истинными или ложными. Некоторые из этих выражений, являющиеся всегда истинными. Объявляются *аксиомами* (или *постулатами*). Они составляют ту базовую систему посылок, исходя из которой и пользуясь определенными правилами вывода, можно получить заключения в виде новых выражений, также являющихся истинными.

Если перечисленные условия выполняются, то говорят, что система удовлетворяет требованиям *формальной теории.* Ее так и называют *формальной системой* (ФС). Система, построенная на основе формальной теории, называется также *аксиоматической системой.*

Формальная теория должна, таким образом, удовлетворять следующему определению:

всякая формальная теория $F = (A, V, W, R)$, определяющая некоторую аксиоматическую систему, характеризуется:

- наличием алфавита (словаря), A;
- множеством синтаксических правил, V;
- множеством аксиом, лежащих в основе теории, W;
- множеством правил вывода, R.

Исчисление высказываний (ИВ) и **исчисление предикатов (ИП)** являются классическими примерами аксиоматических систем. Эти ФС хорошо исследованы и имеют прекрасно разработанные модели логического вывода - главной метапроцедуры в интеллектуальных системах. Поэтому все, что может и гарантирует каждая из этих систем, гарантируется и для прикладных ФС как моделей конкретных предметных областей. В частности, это гарантии непротиворечивости вывода,

алгоритмической разрешимости (для исчисления высказываний) и полуразрешимости (для исчислений предикатов первого порядка).

ФС имеют и недостатки, которые заставляют искать иные формы представления. Главный недостаток - это "закрытость" ФС, их негибкость. Модификация и расширение здесь всегда связаны с перестройкой всей ФС, что для практических систем сложно и трудоемко. В них очень сложно учитывать происходящие изменения. Поэтому ФС как модели представления знаний используются в тех предметных областях, которые хорошо локализуются и мало зависят от внешних факторов.

2.2. ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ

Высказывание есть утвердительное предложение, которое либо истинно, либо ложно, но не то и другое вместе.

Примеры высказываний: «Снег белый», «Николаев – декан».

«Истина» или «ложь», приписанная некоторому высказыванию, называется истинностным значением этого высказывания.

Рассмотрим три истинных предложения:

- За день до своей смерти он был еще жив;
- Если верно, что когда идет дождь, то дорога мокрая, то справедливо также и следующее утверждение: если дорога сухая, то дождя нет;
- Земля вертится.

Чтобы убедиться в правильности первого предложения, достаточно понимать смысл слов: это предложение является *истиной языка*. Чтобы принять второе утверждение, достаточно понимать смысл некоторых слов (если…то, нет), а также знать, что части фразы «идет дождь» и «дорога мокрая» являются *высказываниями*, которые могут быть истинными или ложными, однако все предложение останется истинным, если заменить эти два высказывания другими. Такие истины называются *логическими*

истинами. Третье предложение выражает некоторый факт из физики и астрономии и является *фактической истиной*.

Исчисление высказываний – формальная логическая система. Множество ее базовых элементов составляют логический словарь (алфавит) T из бесконечного счетного множества высказываний, обозначаемых строчными латинскими буквами (иногда с индексами) и называемых *атомами* и пяти элементарных *логических функций* (*связок*):

- «отрицание» -$\neg$, $\sim$, -, not, не;
- «конъюнкция» - $\wedge$, &, and, и;
- «дизъюнкция» - $\vee$, $|$, or, или;
- «импликация» - $\rightarrow$, $\supset$, $\Rightarrow$;
- «эквивалентность» - $\leftrightarrow$, $\equiv$, $\Leftrightarrow$.

Словарь исчисления высказываний дает возможность строить *составные высказывания* из простых, соединяя их логическими связками. *Правила построения S* описывают выражения, являющиеся объектами языка. Такие высказывания называются *формулами*.

Совокупность правил построения формул выглядит так:

– Всякий атом (высказывание) является формулой;
– Если X и Y - формулы, то $\neg X$, $(X \wedge Y)$, $(X \vee Y)$, $(X \rightarrow Y)$ и $(X \leftrightarrow Y)$ – формулы;
– Никаких формул, кроме порожденных применением указанных выше правил, нет.

Круглые скобки позволяют указать порядок, в котором применялись правила. Если в примере 2 утверждений, приведенных выше, обозначим высказывание «идет дождь» буквой P, а высказывание «дорога мокрая» буквой Q, то используя правила построения все утверждение записывается следующим образом:

$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$.

Объектами изучения естественных и формальных языков являются, в частности, синтаксис, который позволяет распознавать фразы среди наборов слов, и семантика, которая придает определенное значение фразам. Это относится и к исчислению высказываний. Любое высказывание может быть либо *истинно*, либо *ложно*. Введем семантическую область {И, Л}. *Интерпретировать формулу* это значит приписать ей одно из двух значений истинности: И или Л. Значение истинности формулы зависит только от структуры этой формулы и от значений истинности составляющих ее высказываний. Таблица истинности логических связок исчисления высказываний приведена ниже.

P	**Q**	**$\neg$ P**	**P $\wedge$ Q**	**P $\vee$ Q**	**P $\rightarrow$ Q**	**P $\leftrightarrow$ Q**
И	И	Л	И	И	И	И
И	Л	Л	Л	И	Л	Л
Л	И	И	Л	И	И	Л
Л	Л	И	Л	Л	И	И

Если формула состоит из нескольких атомов, то истинность формулы определяется при всех возможных комбинациях истинностных значениях атомов, встречающихся в формуле.

Рассмотрим формулу: $(P \wedge Q) \rightarrow (\neg R)$. Таблица истинности для нее будет выглядеть следующим образом:

P	**Q**	**R**	**$\neg$R**	**P $\wedge$ Q**	**(P $\wedge$ Q) $\rightarrow$ ($\neg$R)**
И	И	И	Л	И	Л
И	И	Л	И	И	И
И	Л	И	Л	Л	И
И	Л	Л	И	Л	И
Л	И	И	Л	Л	И
Л	И	Л	И	Л	И

Л	Л	И	Л	Л	И
Л	Л	Л	И	Л	И

Определение 1: *интерпретацией* формулы исчисления высказываний называется такое приписывание истинностных значений атомам формулы, при котором каждому из атомов приписано либо И, либо Л.

Определение 2: Формула *истинна при некоторой интерпретации* тогда и только тогда, когда она получает значение И в этой интерпретации, в противном случае формула ложна.

Определение 3: Формула является *общезначимой (тавтологией)* тогда и только тогда, когда она истинна при всех возможных интерпретациях. Формула является *необщезначимой* тогда и только тогда, когда она не является общезначимой.

Определение 4: Формула является *противоречивой (невыполнимой)* тогда и только тогда, когда она ложна при всех возможных интерпретациях. Формула является *непротиворечивой (выполнимой)* тогда и только тогда, когда она не является противоречивой.

Если Q - тавтология, то ее обозначают как $\models Q$. Если E – множество формул, то запись $E \models Q$ означает, что при всех интерпретациях, при которых истинны все формулы из E, истинна также формула Q. Формула Q называется *логическим следствием* из E. Таким образом, тавтология – логическое следствие из пустого множества. Если E содержит единственный элемент P, то $P \models Q$. Тогда Q является логическим следствием P тогда и только тогда, когда импликация $P \to Q$ есть тавтология, или $P \models Q \leftrightarrow \models (P \to Q)$. В более общем виде можно написать: $\{ F_1, F_2, \ldots, F_n \} \models Q \leftrightarrow \models (F_1 \wedge F_2 \wedge \ldots F_n) \models Q$.

Определение 5: Пусть даны формулы $F_1, F_2, \ldots, F_n$ и формула Q. Говорят, что Q есть *логическое следствие* формул $F_1, F_2, \ldots, F_n$ тогда и

только тогда, когда для всякой интерпретации I, в которой $F_1 \wedge F_2 \wedge ... F_n$ истинна, Q также истинна. $F_1, F_2, ..., F_n$ называются аксиомами (или постулатами, или посылками, или гипотезами).

Если формулы P и Q – логические следствия друг друга, то они называются *логически эквивалентными*. Такая ситуация имеет место тогда и только тогда, когда формула $(P \leftrightarrow Q)$ является тавтологией. Понятие *тавтологии* совпадает с понятием *теоремы* в *аксиоматической системе.* Аксиоматическая система обладает свойством адекватности, то есть она состоит из множества аксиом, считающихся общезначимыми. Кроме аксиом в аксиоматическую систему входит множество *правил вывода*, позволяющих строить новые общезначимые выражения из аксиом и уже полученных общезначимых выражений. Выводимая формула обозначается $\vdash P$.

Древнейшая из аксиоматических теорий – это Евклидова геометрия.

Исчисление высказываний тоже является аксиоматической системой. Любая аксиоматическая система должна удовлетворять следующим требованиям:

- **Непротиворечивость:** невозможность вывода отрицания уже доказанного выражения (которое считается общезначимым);
- **Независимость (минимальность):** система не должна содержать бесполезных аксиом и правил вывода. Некоторое выражение *независимо* от аксиоматической системы, если его нельзя вывести с помощью этой системы. В минимальной системе каждая аксиома независима от остальной системы, то есть не выводима из других аксиом.
- **Полнота (взаимность адекватности):** любая тавтология выводима из системы аксиом. В адекватной системе аксиом любая выводимая формула есть тавтология, то есть верно, что $\vdash$

$\vdash P \rightarrow \models P$. Соответственно в *полной систем верно:* $\models P \rightarrow \vdash P$.

Некоторое множество тавтологий составляет систему аксиом А. Приведем две наиболее известные системы аксиом, обладающие всеми вышеперечисленными свойствами.

Классическая система аксиом:

$P \rightarrow (Q \rightarrow P)$;

$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$;

$(\neg P \rightarrow \neg Q) \rightarrow ((\neg P \rightarrow Q) \rightarrow P)$.

Система аксиом Новикова:

$P \rightarrow (Q \rightarrow P)$;

$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$;

$P \wedge Q \rightarrow P$;

$P \wedge Q \rightarrow Q$;

$(P \rightarrow Q) \rightarrow ((P \rightarrow R) \rightarrow (P \rightarrow Q \wedge R))$;

$P \rightarrow P \vee Q$;

$Q \rightarrow P \vee Q$;

$(P \rightarrow R) \rightarrow ((Q \rightarrow R) \rightarrow (P \vee Q \rightarrow R))$;

$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$;

$P \rightarrow \neg\neg P$;

$\neg\neg P \rightarrow P$.

Существует три обязательных *правила вывода, входящих в B* в исчислении высказываний:

- **Все аксиомы выводимы.**
- **Правило одновременной подстановки:** если некоторая тавтология U содержит атом P, то одновременная замена всех вхождений атома P в U на любую формулу Q приводит к порождению тавтологии.

– ***Modus Ponens* (заключение):** если P – тавтология, и $P \to Q$, то Q – тавтология.

Можно доказать следующие дополнительные правила вывода:

Если P – тавтология, то $Q \to P$ – тавтология (доказательство следует из аксиомы 1.1 и правила вывода 3).

Свойство транзитивности отношения следования: если $P \to Q$, $Q \to R$ – тавтологии, то $P \to R$ – тавтология.

Обобщением правила 4 является теорема дедукции: необходимым и достаточным условием выводимости Q из гипотез R, P является выводимость $P \to Q$ *из* R. Данную теорему можно записать следующим образом: $R, P \vdash Q \leftrightarrow R \vdash (P \to Q)$.

Определение 6. Выводом формулы P из формул $U_1, U_2, \ldots, U_n$ называется последовательность формул $F_1, F_2, \ldots, F_n$ такая, что F_m есть P, а любая F_i либо аксиома, либо одна из формул U_i , либо формула, непосредственно выводимая из предшествующих ей формул.

Часто необходимо преобразовывать формулы из одной формы в другую. Поэтому, кроме аксиом и правил вывода необходимо иметь набор эквивалентных формул (законов), которые позволяют производить преобразования формул:

$P \leftrightarrow Q = P \to Q \wedge Q \to P;$

$P \to Q = \neg P \vee Q;$

Коммуникативные законы:

$P \vee Q = Q \vee P;$

$P \wedge Q = Q \wedge P;$

Ассоциативные законы:

$(P \vee Q) \vee R = Q \vee (P \vee R);$

$(P \wedge Q) \wedge R = Q \wedge (P \wedge R);$

Дистрибутивные законы:

$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R);$

$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R);$

Законы идемпотентности:

$P \vee P = P;\ P \wedge P = P;$

$P \vee Л = P;\ P \wedge И = P;$

$P \vee И = И;\ P \wedge Л = Л;$

$P \vee \neg P = И;\ P \wedge \neg P = Л;$

$\neg(\neg P) = P;$

Законы де Моргана:

$\neg(P \vee Q) = \neg Q \wedge \neg P;$

$\neg(P \wedge Q) = \neg Q \vee \neg P;$

Вследствие законов ассоциативности скобки в выражениях, связанных отношениями дизъюнкции и конъюнкции могут быть опущены, при этом выражение $F_1 \vee F_2 \vee \ldots \vee F_n$ называется дизъюнкцией формул $F_1, F_2, \ldots, F_n$, а выражение $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ называется конъюнкцией формул $F_1, F_2, \ldots, F_n$.

Определение 7. Литерал (литера) есть атом или отрицание атома.

Определение 8. Формула F находится *в конъюнктивной нормальной форме* тогда и только тогда, когда F имеет вид: $F_1 \wedge F_2 \wedge \ldots \wedge F_n$, $n >= 1$, где каждая из $F_1, F_2, \ldots, F_n$ есть *дизъюнкция литералов*.

Определение 9. Формула F находится *в дизъюнктивной нормальной форме* тогда и только тогда, когда F имеет вид: $F_1 \vee F_2 \vee \ldots \vee F_n$, $n >= 1$, где каждая из $F_1, F_2, \ldots, F_n$ есть *конъюнкция литералов*.

Теорема 1. Пусть даны формулы $F_1, F_2, \ldots, F_n$ и формула P. Тогда P есть логическое следствие $F_1, F_2, \ldots, F_n$ тогда и только тогда, когда формула $((F_1 \wedge F_2 \wedge \ldots \wedge F_n) \rightarrow P)$ общезначима.

Теорема 2. Пусть даны формулы $F_1, F_2, \ldots, F_n$ и формула P. Тогда P есть логическое следствие $F_1, F_2, \ldots, F_n$ тогда и только тогда, когда формула $(F_1 \wedge F_2 \wedge \ldots \wedge F_n \wedge \neg P)$ противоречива.

Теоремы 1 и 2 очень важны. Из них следует, что доказательство логического следствия одной формулы из конечного множества формул эквивалентно доказательству того факта, что некоторая связанная с конечным множеством формула общезначима или противоречива.

2.3. ИСЧИСЛЕНИЕ ПРЕДИКАТОВ

2.3.1. Исчисление предикатов первого порядка

В логике высказываний атом рассматривается как единое целое, его структура и состав не анализируется. Однако, есть много умозаключений, которые не могут быть представлены таким простым способом. Например, рассмотрим следующее умозаключение:

Каждый человек смертен.

Так как Конфуций человек, то он смертен.

Приведенное рассуждение интуитивно корректно, однако, если мы введем обозначения:

P: Каждый человек смертен,

Q: Конфуций – человек,

R: Конфуций смертен,

то R не есть логическое следствие P и Q в рамках логики высказываний.

В логике предикатов первого порядка по сравнению с логикой высказываний имеет еще три логических понятия: термы, предикаты и кванторы.

Множество T базовых элементов исчисления предикатов включает в себя следующие символы:

1. Константы – это обычно строчные буквы $a, b, c\ldots$ или осмысленные имена объектов;
2. Переменные – это обычно строчные буквы $x, y, z, \ldots,$ возможно с индексами;

3. Функции – это обычно строчные буквы *f, g, h,…* или осмысленные слова из строчных букв;
4. Предикаты – это обычно прописные буквы *P, Q, R …* или осмысленные слова из прописных букв;
5. Логические связки: отрицание, дизъюнкция, конъюнкция, импликация, эквивалентность;
6. Кванторы всеобщности и существования - ∀, ∃;
7. Открывающая и закрывающая скобка.

Всякая функция или предикатный символ имеет определенное число аргументов. Если функция *f* имеет *n* аргументов, то *f* называется *n*- местной функцией. Аналогично, если предикат *P* имеет *n* аргументов, то *P* называется *n*- местным предикатом.

Определение 10. Термы определяются рекурсивно следующим образом:

- *Константа есть терм;*
- *Переменная есть терм;*
- *Если f есть n- местная функция и* $t_1, t_2,\ldots,t_n$ *– термы, то f* $(t_1, t_2,\ldots,t_n)$ *– терм;*
- *Никаких термов, кроме порожденных применением указанных выше правил, нет.*

Определение 11. Предикат $P(t_1, t_2,\ldots,t_n)$ есть логическая функция, определенная на множестве термов $t_1, t_2,\ldots,t_n$, при фиксированных значениях которых она превращается в высказывания со значением истина (И) или ложь(Л).

Определение 12. Если P – n- местный предикат и $t_1, t_2,\ldots,t_n$ *– термы, то* $P(t_1, t_2,\ldots,t_n)$ *– атом.*

Для построения формул в исчислении предикатов используются пять логических связок и два квантора: ∀ - всеобщности и ∃ - существования. Если *x* – переменная, то (∀*x*) читается как «для всех *x*», «для каждого *x*»,

«для любого *x*», тогда как (∃*x*) читается как « существует *x*», «для некоторых *x*», «по крайней мере для одного *x*».

Пример 1: запишем следующие утверждения:

Каждое рациональное число есть вещественное число.

Существует число, которое является простым.

Для каждого числа x существует такое число y , что x<y.

Обозначим *«x есть простое число»* через *P(x)*, *«x есть рациональное число»* через *Q(x)*, *«x есть вещественное число»* через *R(x)* и *«x меньше y»* через *МЕНЬШЕ (x, y)*.

Тогда указанные выше утверждения могут быть записаны соответственно выражениями:

1. $(\forall x)\ (Q(x) \rightarrow R(x))$,
2. $(\exists x)\ P(x)$,
3. $(\forall x)\ (\exists y)$ *МЕНЬШЕ (x, y)*.

Каждое из выражений *1, 2, 3* называется формулой. Прежде чем дать формальное определение формулы в логике предикатов, следует установить различие между *связанными переменными* и *свободными переменными* и определить *область действия квантора*, входящего в формулу, как ту формулу, к которой этот квантор применяется. Так, область действия квантора существования в выражении *3* есть *МЕНЬШЕ (x, y)*, а область действия квантора всеобщности в выражении *3* есть *(∃y) МЕНЬШЕ (x, y)*.

Определение 13. Вхождение переменной *x* в формулу называется связанным тогда и только тогда, когда оно совпадает с вхождением в квантор (∀*x*) или (∃*y*) или (и?) находится в области действия квантора. Вхождение переменной в формулу свободно тогда и только тогда, когда оно не является связанным.

Определение 14. Переменная свободна в формуле, если хотя бы одно ее вхождение в эту формулу свободно. Отметим, что переменная в формуле может быть свободной и связанной одновременно.

В формуле $(\forall x)\ P(x,y)$переменная x связана, так как оба вхождения x связаны, однако переменная y – свободна, так единственное вхождение y свободно.

Определение 15. Правильно построенные формулы логики первого порядка рекурсивно определяются следующим образом:

Атом есть формула.

Если F и G – формулы, то $\neg(F)$, $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$, $(F \leftrightarrow G)$ – формулы.

Если F – формула, а x – свободная переменная в F, то $(\forall x)\ F$ и $(\exists x)\ F$ – формулы.

Формулы порождаются только конечным числом применений правил 1-3.

Определение 16. Терм t называется свободным для переменной x в формуле f, если ни x, ни другая произвольная переменная из t не находится в области действия никакого квантора $\forall x$ *или* $\exists x$ *в f.*

Пример2: переведем в формулу утверждение «Каждый человек смертен. Конфуций – человек, следовательно, Конфуций смертен».

Обозначим «x есть человек» через $P(x)$, а «x смертен» через $Q(x)$. Тогда утверждение «Каждый человек смертен» может быть представлено формулой $(\forall x)\ (P(x) \rightarrow Q(x))$, утверждение «Конфуций – человек» формулой *P(Конфуций)* и «Конфуций смертен» формулой *Q(Конфуций).*

Утверждение в целом может быть представлено формулой

$(\forall x)\ (P(x) \rightarrow Q(x)) \wedge P(\text{Конфуций}) \rightarrow Q(\text{Конфуций})$.

2.3.2. Интерпретация формул в логике предикатов первого порядка

Чтобы определить интерпретацию для формулы логики первого порядка, мы должны указать предметную область, значения констант, функций и предикатов, встречающихся в формуле.

Определение 17. Интерпретация формулы F логики первого порядка состоит из непустой (предметной) области D и указания значения всех констант, функций и предикатов, встречающихся в F.

1. Каждой константе мы ставим в соответствие некоторый элемент из D.
2. Каждой n- местной функции мы ставим в соответствие отображение из D^n в D (заметим, что $D^n = \{(x_1, x_2, \ldots, x_n) / x_1 \in D, x_2 \in D, \ldots, x_n \in D\}$).
3. Каждому n- местному предикату мы ставим в соответствие отображение D^n в {И, Л}.

Для каждой интерпретации формулы из области D формула может получить значение И или Л согласно следующим правилам:

1. Если заданы значения формул F и G, то истинностные значения формул $\neg(F)$, $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$, $(F \leftrightarrow G)$ получаются с помощью таблиц истинности соответствующих логических связок.
2. $(\forall x)$ F получает значение И, если F получает значение И для каждого x из D , в противном случае она получает значение Л.
3. $(\exists x)$ F получает значение И, если F получает значение И хотя бы для одного x из D , в противном случае она получает значение Л.

Отметим, что формула, содержащая свободные переменные, не может получить истинностное значение. Поэтому, в дальнейшем, будем считать, что формула либо не содержит свободных переменных, либо свободные переменные рассматриваются как константы.

Пример 3. Рассмотрим формулы $(\forall x)$ $P(x)$ *и* $(\exists x)$ $\neg P(x)$.

Пусть интерпретации такова: область - $D=\{1,2\}$;

Оценка для P :P(1) =И, P(2) =Л.

В таком случае $(\forall x)$ *P(x) есть Л, а* $(\exists x)$ $\neg P(x)$ *есть И в данной интерпретации.*

Пример 4. Рассмотрим формулы $(\forall x)$ $(\exists y)$ *P(x, y).*

Пусть интерпретации такова: область - D={1,2};

Оценка для P: P(1,1) =И, P(1,2) =Л, P(2,1) =Л, P(2,2) =И.

При x=1 существует y=1, что P(x, y) =И.

При x=2 существует y=2, что P(x, y) =И.

Следовательно, в указанной интерпретации, для каждого x из D существует такой y, что P(x, y)=И, то есть $(\forall x)$ $(\exists y)$ *P(x, y) есть И в данной интерпретации.*

Определение 18: Формула является *непротиворечивой (выполнимой)* тогда и только тогда, когда существует такая интерпретация *I,* что *G* имеет значение *И* в *I*. В этом случае говорят, что *I* удовлетворяет *G*.

Определение 19: Формула является *противоречивой (невыполнимой)* тогда и только тогда, когда не существует интерпретации, которая удовлетворяет *G*.

Определение 20: Формула *G* является *общезначимой* тогда и только тогда, когда не существует никакой интерпретации, которая не удовлетворяет формуле *G*.

Определение 21: Формула *G* есть *логическое следствие* формул F_1, $F_2, \ldots, F_n$ тогда и только тогда, когда для каждой интерпретации *I*, если $F_1 \wedge F_2 \wedge \ldots F_n$ истинна в *I*, то *G* также истинна в *I*.

Теоремы 1 и 2 верны также и для логики предикатов первого порядка.

Пример 5. Рассмотрим формулы:

F1: $(\forall x)$ $(P(x) \rightarrow Q(x))$,

F2: P(a).

Докажем, что Q(a) есть логическое следствие формул F1 и F2.

Рассмотрим любую интерпретацию I, которая удовлетворяет $(\forall x)(P(x) \rightarrow Q(x)) \wedge P(a)$. *Конечно, в этой интерпретации P(a) есть И. Пусть Q(a) есть Л в данной интерпретации, тогда* $P(a) \rightarrow Q(a)$ *есть Л в данной интерпретации по определению операции импликации. Это значит, что* $(\forall x)(P(x) \rightarrow Q(x))$ *есть Л в I, что невозможно. Следовательно, Q(a) должна быть И в каждой интерпретации, которая удовлетворяет* $(\forall x)(P(x) \rightarrow Q(x)) \wedge P(a)$. *Это означает, что Q(a) есть логическое следствие из F1 и F2.*

Так как в логике первого порядка имеется бесконечное число областей, то имеется бесконечное число интерпретаций формулы. Следовательно, в отличие от логики высказываний, невозможно доказать общезначимость или противоречивость формулы оценкой формулы при всех возможных интерпретациях.

2.3.3. Системы аксиом логики предикатов

Системы аксиом исчисления высказываний остаются верными и в исчислении предикатов первого порядка, только к ним следует добавить еще две аксиомы, которые дают возможность оперировать с кванторами:

1. $(\forall x)\, P(x) \rightarrow P(y)$;
2. $P(y) \rightarrow \exists\, P(x)$.

Эти 2 аксиомы, добавленные в классическую систему аксиом или в систему аксиом Новикова, образуют системы аксиом, обладающие свойствами полноты, независимости и непротиворечивости.

2.3.4. Правила вывода в исчислении предикатов

Из правил вывода исчисления высказываний в исчислении предикатов действует только правило *Modus Ponens*. Правило одновременной подстановки модифицировано, а остальные правила вывода касаются выводимости формул, содержащих кванторы.

1. *Modus Ponens: Если выводима формула P и выводима формула P →Q, то выводима и формула Q. Часто это правило записывают следующим образом:*

   $\frac{P,\ P \to Q}{Q}$;

2. *Правило одновременной подстановки: если терм t свободен для переменной x в формуле F, то можно подставить терм t вместо переменной x во всех вхождениях x в F.*
3. *Правило обобщения: если P не содержит свободных вхождений переменной x, то*

   $\frac{P \to Q(x)}{P \to \forall x Q(x)}$;

4. *Правило конкретизации: если P не содержит свободных вхождений переменной x, то*

   $\frac{Q(x) \to P}{\exists x Q(x) \to P}$;

5. *Правило переименования. Из выводимости формулы F(x), содержащей свободное вхождение x, ни одно из которых не содержится в области действия кванторов ∀y и ∃y следует выводимость F(y).*

Пример 6. Докажем правило переименования:

1. *+F(x);*
2. *Из аксиомы 2 классической системы следует, что* $F(x) \to (G \to F(x))$, *где G – тавтология, не содержащая свободный вхождений x;*
3. *По правилу Modus Ponens следует, что* $\frac{F(x), F(x) \to (G \to F(x))}{G \to F(x)}$;
4. *Используя правило обобщения, получаем:* $G \to \forall x F(x)$;
5. *По правилу Modus Ponens следует, что* $\frac{G, G \to \forall x F(x)}{\forall x F(x)}$;

6. *Из аксиомы 2 логики предикатов и правила Modus Ponens следует, что* $\dfrac{\forall x F(x), \forall x F(x) \to F(y)}{F(y)}$.

2.4. НОРМАЛЬНЫЕ ФОРМЫ ИСЧИСЛЕНИЯ ПРЕДИКАТОВ

2.4.1. Предваренные (пренексные) нормальные формы исчисления предикатов

В логике высказываний были введены две нормальные формы – КНФ и ДНФ. В логике предикатов также существуют нормальная форма - ПНФ, которая используется для упрощения процедуры доказательства общезначимости или противоречивости формул.

Определение 22: *Формула F логики предикатов находится в предваренной нормальной форме, тогда и только тогда, когда формула F имеет вид:*

$(K_1x_1)\ldots(K_nx_n)$ *(M), где каждое* (K_ix_i), $i = 1,\ldots,n$, *есть или* $(\forall x_i)$ *или* $(\exists x_i)$, *и M есть формула, не содержащая кванторов.* $(K_1x_1)\ldots(K_nx_n)$ *называется префиксом, а M – матрицей формулы F.*

Законы эквивалентных преобразований логики высказываний используются и в логике предикатов. Кроме них, существуют другие эквивалентные формулы, содержащие кванторы.

Пусть *P* есть формула, содержащая свободную переменную *x*. Пусть *Q* есть формула, которая не содержит переменной *x*. Тогда следующие пары эквивалентных формул являются законами эквивалентных преобразований логики предикатов:

1. $(\forall x)\ P(x) \vee Q = (\forall x)\ (P(x) \vee Q)$;
2. $(\forall x)\ P(x) \wedge Q = (\forall x)\ (P(x) \wedge Q)$;
3. $(\exists x)\ P(x) \vee Q = (\exists x)\ (P(x) \vee Q)$;
4. $(\exists x)\ P(x) \wedge Q = (\exists x)\ (P(x) \wedge Q)$;
5. $\neg((\forall x)\ P(x)) = (\exists x)\ (\neg P(x))$;
6. $\neg((\exists x)\ P(x)) = (\forall x)\ (\neg P(x))$;

7. $(\forall x)\ P(x) \wedge (\forall x)\ Q(x) = (\forall x)\ (P(x) \wedge Q(x))$;

8. $(\exists x)\ P(x) \vee (\exists x)\ Q(x) = (\exists x)\ (P(x) \vee Q(x))$;

Правила 7 и 8 называются правилами выноса кванторов, которые позволяют распределять квантор всеобщности и квантор существования по операциям конъюнкции и дизъюнкции соответственно. Следует отметить, что *нельзя* распределять квантор всеобщности и квантор существования по операциям дизъюнкции и конъюнкции соответственно, то есть не эквивалентны следующие пары формул:

$(\forall x)\ P(x) \vee (\forall x)\ Q(x) <> (\forall x)\ (P(x) \vee Q(x))$;

$(\exists x)\ P(x) \wedge (\exists x)\ Q(x) <> (\exists x)\ (P(x) \wedge Q(x))$;

В подобных случаях можно заменить связанную переменную x в формуле $(\forall x)\ Q(x)$ на переменную z, которая не всречается в $P(x)$, так как связанная переменная является лишь местом для подстановки какой угодно переменной. Формула примет вид: $(\forall x)\ Q(x) = (\forall z)\ Q(z)$. Пусть z не встречается в $P(x)$. Тогда

$(\exists x)\ P(x) \wedge (\exists x)\ Q(x) = (\exists x)\ P(x) \wedge (\exists z)\ Q(z)$

$= (\exists x)\ (\exists z)(\ P(x) \wedge Q(z))$ по правилу 1.

Аналогично, можно написать

$(\forall x)\ P(x) \vee (\forall x)\ Q(x) = (\forall x)\ P(x) \vee (\forall z)\ Q(z)$

$= (\forall x)\ (\forall z)(\ P(x) \vee Q(z))$ по правилу 1.

В общем случае эти правила можно записать в следующем виде:

9. $(K_1 x)\ P(x) \vee (K_2 x)\ Q(x) = (K_1 x)\ (K_2 z)(\ P(x) \vee Q(z))$;

10. $(K_3 x)\ P(x) \wedge (K_4 x)\ Q(x) = (K_3 x)\ (K_4 z)(\ P(x) \wedge Q(z))$,

где K_1, K_2, K_3, K_4 есть кванторы $\forall$ или $\exists$, а z не входит в P(x).

Когда $K_1 = K_2 = \exists$ и $K_3 = K_4 = \forall$, то необязательно переименовывать переменную x, можно прямо использовать правила 7 и 8.

Используя законы эквивалентных преобразований логики высказываний и логики предикатов, всегда можно преобразовать любую формулу в ПНФ.

Алгоритм преобразования формул в ПНФ

Шаг 1. Используем законы 1 и 2 исчисления высказываний для того, чтобы исключить логические связки импликации и эквивалентности.

Шаг 2. Многократно используем закон двойного отрицания, законы де Моргана и законы 5 и 6 исчисления предикатов, чтобы внести знак отрицания внутрь формулы.

Шаг 3. Переименовываем связанные переменные, если это необходимо.

Шаг 4. Используем законы 1, 2, 3, 4, 7,8, 9 и 10 до тех пор, пока все кванторы не будут вынесены в самое начало формулы, чтобы получить ПНФ.

Пример 6. Приведем формулу $(\forall x)\ P(x) \to (\exists x)\ Q(x)$ *к ПНФ:*

$(\forall x)\ P(x) \to (\exists x)\ Q(x) = \neg((\forall x)\ P(x)) \vee (\exists x)\ Q(x)$ *(по закону 1 логики высказываний)*

$=(\exists x)(\neg P(x)) \vee (\exists x))\ Q(x)$ *(по закону 5 логики предикатов)*

$=(\exists x)(\neg P(x) \vee Q(x))$ *(по закону 8 логики предикатов), что и есть ПНФ исходной формулы.*

2.4.2. Автоматизация доказательства в логике предикатов

Поиск общей разрешающей процедуры для проверки общезначимости формул начал Лейбниц в XVII веке, затем был продолжен в начале XX века до тех пор, пока в 1936 году Черч и Тьюринг независимо не доказали, что не существует никакой общей разрешающей процедуры, никакого алгоритма, проверяющего общезначимость формул в логике предикатов первого порядка.

Тем не менее, существуют алгоритмы поиска доказательства, которые могут подтвердить, что формула общезначима, если она на самом деле общезначима (для необщезначимых формул эти алгоритмы, вообще говоря, не заканчивают свою работу).

Очень важный подход к автоматическому доказательству теорем был дан Эрбраном в 1930 году. По определению общезначимая формула есть формула, которая истинна при всех интерпретациях. Эрбран разработал алгоритм нахождения интерпретации, которая опровергает данную формулу. Однако, если данная формула действительно общезначима, то никакой интерпретации не существует и алгоритм заканчивает работу за конечное число шагов. Метод Эрбрана служит основой для большинства современных автоматических алгоритмов поиска доказательства.

Гилмор в 1959 году одним из первых реализовал процедуру Эрбрана. Его программа была предназначена для обнаружения противоречивости отрицания данной формулы, так как формула общезначима тогда и только тогда, когда ее отрицание противоречиво. Однако, программа Гилмора оказалась неэффективной и в 1960 году метод Гилмора был улучшен Девисом и Патнемом. Однако их улучшения оказалось недостаточным, так как многие общезначимые формулы логики предикатов все еще не могли быть доказаны на ЭВМ за разумное время.

Главный шаг вперед сделал Робинсон в 1965 году, который ввел так называемый метод резолюций, который оказался много эффективней, чем любая описанная ранее процедура. После введения метода резолюций был предложен ряд стратегий для увеличения его эффективности. Такими стратегиями являются *семантическая резолюция, лок-резолюция, линейная резолюция, стратегия предпочтения единичных и стратегия поддержки.*

Скулемовские стандартные формы

Процедуры доказательства по Эрбрану или методу резолюций на самом деле являются процедурами опровержения, то есть вместо

доказательства общезначимости формулы доказывается, что ее отрицание противоречиво. Кроме того, эти процедуры опровержения применяются к некоторой стандартной форме, которая была введена Девисом и Патнемом. По существу, они использовали следующие идеи:

Формула логики предикатов может быть сведена к ПНФ, в которой матрица не содержит никаких кванторов, а префикс есть последовательность кванторов.

Поскольку матрица не содержит кванторов, она может быть сведена к конъюнктивной нормальной форме.

Сохраняя противоречивость формулы, в ней можно исключить кванторы существования путем использования скулемовских функций.

Алгоритм преобразования формулы в ПНФ известен. При помощи законов эквивалентных преобразований логики высказываний можно свести матрицу к КНФ.

Алгоритм преобразования формул в ДНФ и КНФ

Шаг 1. Используем законы 1 и 2 исчисления высказываний для того, чтобы исключить логические связки импликации и эквивалентности.

Шаг 2. Многократно используем закон двойного отрицания, и законы де Моргана, чтобы внести знак отрицания внутрь формулы.

Шаг 3. Несколько раз используем дистрибутивные законы и другие законы, чтобы получить НФ.

Алгоритм преобразования формулы $(K_1x_1)\ldots(K_nx_n)\ (M)$, где каждое (K_ix_i), $i = 1,\ldots,n$, есть или $(\forall x_i)$ или $(\exists x_i)$, и M есть *КНФ* в скулемовскую нормальную форму (СНФ) приведен ниже.

Алгоритм преобразования ПНФ в ССФ.

Шаг 1. Представим формулу в ПНФ $(K_1x_1)\ldots(K_nx_n)\ (M)$, где M есть *КНФ*. Пусть K_r есть квантор существования в префиксе $(K_1x_1)\ldots(K_nx_n)$, $1<=r<=n$.

Шаг 2. Если никакой квантор всеобщности не стоит левее K_r – выберем новую константу c, отличную от других констант, входящих в M, заменим все x_r в M на c и вычеркнем $K_r x_r$ из префикса. Если $K_1, \ldots K_i$ – список всех кванторов всеобщности, встречающихся в M левее K_r, $1< i<r$, выберем новый i –местный функциональный символ f, отличный от других функциональных символов, заменим все x_r в M на $f(x_1, x_2, \ldots x_i)$ и вычеркнем $K_r x_r$ из префикса.

Шаг 3. Применим шаг 2 для всех кванторов существования в префиксе. Последняя из полученных формул есть *скулемовская стандартная форма* формулы. Константы и функции, используемые для замены переменных квантора существования, называются *скулемовскими* функциями.

Пример 7. Получить ССФ для формулы $(\exists x)(\forall y)(\forall z)(\exists u)(\forall v)(\exists w)\ (P(x, y, z, u, v, w)$.

В этой формуле левее $(\exists x)$ *нет никаких кванторов всеобщности, левее* $(\exists u)$ *стоят* $(\forall y)$ *и* $(\forall z)$, *а левее* $(\exists w)$ *стоят* $(\forall y)$, $(\forall z)$ *и* $(\forall v)$. *Следовательно, мы заменим переменную x на константу a, переменную u - на двухместную f(y, z), переменную w - на трехместную функцию g(y, z, v). Таким образом, мы получаем следующую стандартную форму написанной выше формулы:*

$(\forall y)(\forall z)(\forall v)(P(a, y, z, f(y, z), g(y, z, v))$.

Определение 22: *Дизъюнктом называется дизъюнкция литералов. Дизъюнкт, содержащий r литералов, называется r- литеральным дизъюнктом. Однолитеральный дизъюнкт называется единичным дизъюнктом. Если дизъюнкт не содержит никаких литералов, то он называется пустым дизъюнктом-* □ . *Так как пустой дизъюнкт не содержит литер, которые могли бы быть истинными при любых интерпретациях, то пустой дизъюнкт всегда ложен.*

Определение 23: *Множество дизъюнктов S есть конъюнкция всех дизъюнктов из S , где каждая переменная в S считается управляемой квантором всеобщности.*

Вследствие последнего определения, ССФ может быть представлена множеством дизъюнктов.

Пример 8: ССФ $(\forall x)((\neg P(x, f(x)) \vee R(x, f(x), g(x))) \wedge (Q(x, g(x)) \vee R(x, f(x), g(x))))$ *представить в виде множества дизъюнктов.*

$\{\neg P(x, f(x)) \vee R(x, f(x), g(x)), Q(x, g(x)) \vee R(x, f(x), g(x))\}$.

Теорема 3. Пусть S – множество дизъюнктов, которые представляют ССФ формулы F. Тогда F противоречива в том и только в том случае, когда S противоречиво.

На основании теорем 2 и 3 можно сделать вывод, что формула G является логическим следствием формулы F тогда, когда противоречива конъюнкция множества S и формулы $\neg G$*, то есть противоречива формула* $S_1 \wedge S_2 \wedge \ldots S_n \wedge \neg G$*. Таким образом, если в множество S добавить негативный литерал* $\neg G$ *и доказать, что полученное множество противоречиво, то тем самым можно доказать выводимость G из множества S.*

2.5. МЕТОДЫ РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Функционирование многих ИС носит целенаправленный характер (примером могут служить автономные интеллектуальные роботы). Типичным актом такого функционирования является решение задачи планирования пути достижения нужной цели из некоторой фиксированной начальной ситуации. Результатом решения задачи должен быть *план действий* - частично-упорядоченная совокупность действий. Такой план напоминает сценарий, в котором в качестве отношения между вершинами выступают отношения типа: "цель-подцель" "цель-действие", "действие-

результат" и т. п. Любой путь в этом сценарии, ведущий от вершины, соответствующей текущей ситуации, в любую из целевых вершин, определяет план действий.

Поиск плана действий возникает в ИС лишь тогда, когда она сталкивается с нестандартной ситуацией, для которой нет заранее известного набора действий, приводящих к нужной цели. Все задачи построения плана действий можно разбить на два типа, которым соответствуют различные модели: *планирование в пространстве состояний* (SS-проблема) и *планирование в пространстве задач* (PR-проблема).

В первом случае считается заданным некоторое пространство ситуаций. Описание ситуаций включает состояние внешнего мира и состояние ИС, характеризуемые рядом параметров. Ситуации образуют некоторые обобщенные состояния, а действия ИС или изменения во внешней среде приводят к изменению актуализированных в данный момент состояний. Среди обобщенных состояний выделены начальные состояния (обычно одно) и конечные (целевые) состояния. SS-проблема состоит в поиске пути, ведущего из начального состояния в одно из конечных. Если, например, ИС предназначена для игры в шахматы, то обобщенными состояниями будут позиции, складывающиеся на шахматной доске. В качестве начального состояния может рассматриваться позиция, которая зафиксирована в данный момент игры, а в качестве целевых позиций - множество ничейных позиций. Отметим, что в случае шахмат прямое перечисление целевых позиций невозможно. Матовые и ничейные позиции описаны на языке, отличном от языка описания состояний, характеризуемых расположением фигур на полях доски. Именно это затрудняет поиск плана действий в шахматной игре.

При планировании в пространстве задач ситуация несколько иная. Пространство образуется в результате введения на множестве задач

отношения типа: "часть - целое", "задача - подзадача", "общий случай - частный случай" и т. п. Другими словами, пространство задач отражает декомпозицию задач на подзадачи (цели на подцели). PR-проблема состоит в поиске декомпозиции исходной задачи на подзадачи, приводящей к задачам, решение которых системе известно. Например, ИС известно, как вычисляются значения *sin x* и *cos x* для любого значения аргумента и как производится операция деления. Если ИС необходимо вычислить *tg x,* то решением PR-проблемы будет представление этой задачи в виде декомпозиции $tgx=\sin x/\cos x$ (кроме $x=\pi/2+k\pi$).

2.5.1. Решение задач методом поиска в пространстве состояний

Представление задач в пространстве состояний предполагает задание ряда описаний: состояний, множества операторов и их воздействий на переходы между состояниями, целевых состояний. Описания состояний могут представлять собой строки символов, векторы, двухмерные массивы, деревья, списки и т. п. Операторы переводят одно состояние в другое. Иногда они представляются в виде продукций A=>B, означающих, что состояние *A* преобразуется в состояние *B.*

Пространство состояний можно представить как граф, вершины которого помечены состояниями, а дуги - операторами.

Таким образом, проблема поиска решения задачи *<A,B>* при планировании по состояниям представляется как проблема поиска на графе пути из *A* в *B.* Обычно графы не задаются, а генерируются по мере надобности.

Различаются *слепые* и *направленные методы поиска пути.* Слепой метод имеет два вида: *поиск вглубь и поиск вширь.* При поиске вглубь каждая альтернатива исследуется до конца, без учета остальных альтернатив. Метод плох для "высоких" деревьев, так как легко можно проскользнуть мимо нужной ветви и затратить много усилий на

исследование "пустых" альтернатив. При поиске вширь на фиксированном уровне исследуются все альтернативы и только после этого осуществляется переход на следующий уровень. Метод может оказаться хуже метода поиска вглубь, если в графе все пути, ведущие к целевой вершине, расположены примерно на одной и той же глубине. Оба слепых метода требуют большой затраты времени и поэтому необходимы направленные методы поиска.

Метод ветвей и границ. Из формирующихся в процессе поиска неоконченных путей выбирается самый короткий и продлевается на один шаг. Полученные новые неоконченные пути (их столько, сколько ветвей в данной вершине) рассматриваются наряду со старыми, и вновь продлевается на один шаг кратчайший из них. Процесс повторяется до первого достижения целевой вершины, решение запоминается. Затем из оставшихся неоконченных путей исключаются более длинные, чем законченный путь, или равные ему, а оставшиеся продлеваются по такому же алгоритму до тех пор, пока их длина меньше законченного пути. В итоге либо все неоконченные пути исключаются, либо среди них формируется законченный путь, более короткий, чем ранее полученный. Последний путь начинает играть роль эталона и т. д.

Алгоритм кратчайших путей Мура. Исходная вершина X_0 помечается числом 0. Пусть в ходе работы алгоритма на текущем шаге получено множество дочерних вершин $X(x_i)$ вершины x_i**.** Тогда из него вычеркиваются все ранее полученные вершины, оставшиеся помечаются меткой, увеличенной на единицу по сравнению с меткой вершины x_i, и от них проводятся указатели к X_i. Далее на множестве помеченных вершин, еще не фигурирующих в качестве адресов указателей, выбирается вершина с наименьшей меткой и для нее строятся дочерние вершины. Разметка вершин повторяется до тех пор, пока не будет получена целевая вершина.

Алгоритм Дейкстры определения путей с минимальной стоимостью является обобщением алгоритма Мура за счет введения дуг переменной длины.

Алгоритм Дорана и Мичи поиска с низкой стоимостью. Используется, когда стоимость поиска велика по сравнению со стоимостью оптимального решения. В этом случае вместо выбора вершин, наименее удаленных от начала, как в алгоритмах Мура и Дийкстры, выбирается вершина, для которой эвристическая оценка расстояния до цели наименьшая. При хорошей оценке можно быстро получить решение, но нет гарантии, что путь будем минимальным.

Алгоритм Харта, Нильсона и Рафаэля. В алгоритме объединены оба критерия: стоимость пути до вершины $g[x)$ и стоимость пути от вершины $h(x)$ - в аддитивной оценочной функции $f\{x) =g(x\}-h(x)$. При условии $h(x)<hp(x)$, где $hp(x)$-действительное расстояние до цели, алгоритм гарантирует нахождение оптимального пути.

Алгоритмы поиска пути на графе различаются также направлением поиска. Существуют прямые, обратные и двунаправленные методы поиска. Прямой поиск идет от исходного состояния и, как правило, используется тогда, когда целевое состояние задано неявно. Обратный поиск идет от целевого состояния и используется тогда, когда исходное состояние задано неявно, а целевое явно. Двунаправленный поиск требует удовлетворительного решения двух проблем: смены направления поиска и оптимизации "точки встречи". Одним из критериев для решения первой проблемы является сравнение "ширины" поиска в обоих направлениях - выбирается то направление, которое сужает поиск. Вторая проблема вызвана тем, что прямой и обратный пути могут разойтись и чем уже поиск, тем это более вероятно.

2.5.2. Решение задач методом редукции

Этот метод приводит к хорошим результатам потому, что часто решение задач имеет иерархическую структуру. Однако не обязательно требовать, чтобы основная задача и все ее подзадачи решались одинаковыми методами. Редукция полезна для представления глобальных аспектов задачи, а при решении более специфичных задач предпочтителен метод планирования по состояниям. Метод планирования по состояниям можно рассматривать как частный случай метода планирования с помощью редукций, ибо каждое применение оператора в пространстве состояний означает сведение исходной задачи к двум более простым, из которых одна является элементарной. В общем случае редукция исходной задачи не сводится к формированию таких двух подзадач, из которых хотя бы одна была элементарной.

Поиск планирования в пространстве задач заключается в последовательном сведении исходной задачи к все более простым до тех пор, пока не будут получены только элементарные задачи. Частично упорядоченная совокупность таких задач составит решение исходной задачи. Расчленение задачи на альтернативные множества подзадач удобно представлять в виде И/ИЛИ-графа. В таком графе всякая вершина, кроме концевой, имеет либо конъюнктивно связанные дочерние вершины (И-вершина), либо дизъюнктивно связанные (ИЛИ-вершина). В частном случае, при отсутствии И-вершин, имеет место граф пространства состояний. Концевые вершины являются либо заключительными (им соответствуют элементарные задачи), либо тупиковыми. Начальная вершина (корень И/ИЛИ-графа) представляет исходную задачу. Цель поиска на И/ИЛИ-графе-показать, что начальная вершина разрешима. Разрешимыми являются заключительные вершины (И-вершины), у которых разрешимы все дочерние вершины, и ИЛИ-вершины, у которых разрешима хотя бы одна дочерняя вершина. Разрешающий граф состоит из

разрешимых вершин и указывает способ разрешимости начальной вершины. Наличие тупиковых вершин приводит к неразрешимым вершинам. Неразрешимыми являются тупиковые вершины, И-вершины, у которых неразрешима хотя бы одна дочерняя вершина, и ИЛИ-вершины, у которых неразрешима каждая дочерняя вершина.

Алгоритм Ченга и Слейгла. Основан на преобразовании произвольного И/ИЛИ-графа в специальный ИЛИ-граф, каждая ИЛИ-ветвь которого имеет И-вершины только в конце. Преобразование использует представление произвольного И/ИЛИ-графа как произвольной формулы логики высказываний с дальнейшим преобразованием этой произвольной формулы в дизъюнктивную нормальную форму. Подобное преобразование позволяет далее использовать алгоритм Харта, Нильсона и Рафаэля.

Метод ключевых операторов. Пусть задана задача <A, B> и известно, что оператор f обязательно должен входить в решение этой задачи. Такой оператор называется ключевым. Пусть для применения f необходимо состояние C, а результат его применения есть I(c). Тогда И-вершина <A,B> порождает три дочерние вершины: <A, C>, <C, f{c)> и <f(c), B>, из которых средняя является элементарной задачей. К задачам <A, C> и <f(c), B> также подбираются ключевые операторы, и указанная процедура редуцирования повторяется до тех пор, пока это возможно. В итоге исходная задача <A, B> разбивается на упорядоченную совокупность подзадач, каждая из которых решается методом планирования в пространстве состояний.

Возможны альтернативы по выбору ключевых операторов, так что в общем случае будет иметь место И/ИЛИ-граф. В большинстве задач удается не выделить ключевой оператор, а только указать множество, его содержащее. В этом случае для задачи <A, B> вычисляется различие между A и B, которому ставится в соответствие оператор, устраняющий это различие. Последний и является ключевым.

Метод планирования общего решателя задач (ОРЗ). ОРЗ явился первой наиболее известной моделью планировщика. Он использовался для решения задач интегрального исчисления, логического вывода, грамматического разбора и др. ОРЗ объединяет два основных принципа поиска: анализ целей и средств и рекурсивное решение задач. В каждом цикле поиска ОРЗ решает в жесткой последовательности три типа стандартных задач: преобразовать объект *A* в объект *B,* уменьшить различие *D* между *A* и *B,* применить оператор f к объекту *A.* Решение первой задачи определяет различие *D* второй - подходящий оператор f, третьей - требуемое условие применения *C.* Если *C* не отличается от А, то оператор f применяется, иначе *C* представляется как очередная цель и цикл повторяется, начиная с задачи "преобразовать А в С". В целом стратегия ОРЗ осуществляет обратный поиск-от заданной цели *B к* требуемому средству ее достижения С, используя редукцию исходной задачи <А, *B*> *к* задачам <А, С> и <*C, B*>.

Заметим, что в ОРЗ молчаливо предполагается независимость различий Друг от друга, откуда следует гарантия, что уменьшение одних различий не приведет к увеличению других.

3. Планирование с помощью логического вывода. Такое планирование предполагает: описание состояний в виде правильно построенных формул (ППФ) некоторого логического исчисления, описание операторов в виде либо ППФ, либо правил перевода одних ППФ в другие. Представление операторов в виде ППФ позволяет создавать дедуктивные методы планирования, представление операторов в виде правил перевода - методы планирования с элементами дедуктивного вывода.

Дедуктивный метод планирования системы QA3, ОРЗ не оправдал возлагавшихся на него надежд в основном из-за неудовлетворительного представления задач. Попытка исправить положение привела к созданию вопросно-ответной системы QA3. Система рассчитана на произвольную

предметную область и способна путем логического вывода ответить на вопрос: возможно ли достижение состояния В из А? В качестве метода автоматического вывода используется принцип резолюций. Для направления логического вывода QA3 применяет различные стратегии, в основном синтаксического характера, учитывающие особенности формализма принципа резолюций. Эксплуатация QA3 показала, что вывод в такой системе получается медленным, детальным, что несвойственно рассуждениям человека.

Метод продукций системы STRIPS. В этом методе оператор представляет продукцию *P, A=>B,* где *P, A* и В - множества ППФ исчисления предикатов первого порядка, *P* выражает условия применения ядра продукции *A=>B,* где В содержит список добавляемых ППФ и список исключаемых ППФ, т. е. постусловия. Метод повторяет метод ОРЗ с тем отличием, что стандартные задачи определения различий и применения подходящих операторов решаются на основе принципа резолюций. Подходящий оператор выбирается так же, как в ОРЗ, на основе принципа "анализ средств и целей". Наличие комбинированного метода планирования позволило ограничить процесс логического вывода описанием состояния мира, а процесс порождения новых таких описаний оставить за эвристикой "от цели к средству ее достижения".

Метод продукций, использующий макрооператоры [Файкс и др., 1973]. Макрооператоры-это обобщенные решения задач, получаемые методом STRIPS. Применение макрооператоров позволяет сократить поиск решения, однако при этом возникает проблема упрощения применяемого макрооператора, суть которой заключается в выделении по заданному различию его требуемой части и исключении из последней ненужных операторов.

2.5.3. Решение задач дедуктивного выбора

В дедуктивных моделях представления и обработки знании решаемая проблема записывается в виде утверждении формальной системы, цель-в виде утверждения, справедливость которого следует установить или опровергнуть на основании аксиом (общих законов) и правил вывода формальной системы. В качестве формальной системы используют исчисление предикатов первого порядка.

В соответствии с правилами, установленными в формальной системе, заключительному утверждению-теореме, полученной из начальной системы утверждений (аксиом, посылок), приписывается значение ИСТИНА, если каждой посылке, аксиоме также приписано значение ИСТИНА.

Процедура вывода представляет собой процедуру, которая из заданной группы выражений выводит отличное от заданных выражение.

Обычно в логике предикатов используется формальный метод доказательства теорем, допускающий возможность его машинной реализации, но существует также возможность доказательства неаксиоматическим путем: прямым выводом, обратным выводом.

Метод резолюции используется в качестве полноценного (формального) метода доказательства теорем.

Для применения этого метода исходную группу заданных логических формул требуется преобразовать в некоторую нормальную форму. Это преобразование проводится в несколько стадий, составляющих машину вывода.

2.5.4. Решение задач, использующие немонотонные логики, вероятностные логики

Данные и знания, с которыми приходится иметь дело в ИС, редко бывают абсолютно точными и достоверными. Присущая знаниям

неопределенность может иметь разнообразный характер, и для ее описания используется широкий спектр формализмов. Рассмотрим один из типов неопределенности в данных и знаниях - их неточность. Будем называть высказывание *неточным,* если его истинность (или ложность) не может быть установлена с определенностью. Основополагающим понятием при построении моделей *неточного вывода* является понятие вероятности, поэтому все описываемые далее методы связаны с вероятностной концепцией.

Модель оперирования с неточными данными и знаниями включает две составляющие: язык представления неточности и механизм вывода на неточных знаниях. Для построения языка необходимо выбрать форму представления неточности (например, скаляр, интервал, распределение, лингвистическое выражение, множество) и предусмотреть возможность приписывания меры неточности всем высказываниям.

Механизмы оперирования с неточными высказываниями можно разделить на два типа. К первому относятся механизмы, носящие "присоединенный" характер: пересчет мер неточности как бы сопровождает процесс вывода, ведущийся на точных высказываниях. Для разработки присоединенной модели неточного вывода в основанной на правилах вывода системе необходимо задать функции пересчета, позволяющие вычислять:

а) меру неточности антецедента правила (его левой части) по мерам неточности составляющих его высказываний;

б) меру неточности консеквента правила (его правой части) по мерам неточности правила и посылки правила;

в) объединенную меру неточности высказывания по мерам, полученным из правил.

Введение меры неточности позволит привнести в процесс вывода нечто принципиально новое - возможность объединения силы нескольких

свидетельств, подтверждающих или опровергающих одну и ту же гипотезу. Другими словами, при использовании мер неточности целесообразно выводить одно и то же утверждение различными путями (с последующим объединением значений неточности), что совершенно бессмысленно в традиционной дедуктивной логике. Для *объединения свидетельств* требуется функция пересчета, занимающая центральное место в пересчете. Заметим, что, несмотря на "присоединенность" механизмов вывода этого типа, их реализация в базах знаний оказывает влияние на общую стратегию вывода: с одной стороны, необходимо выводить гипотезу всеми возможными путями для того, чтобы учесть все релевантные этой гипотезе свидетельства, с другой-предотвратить многократное влияние силы одних и тех же свидетельств.

Для механизмов оперирования с неточными высказываниями второго типа характерно наличие схем вывода, специально ориентированных на используемый язык представления неточности. Как правило, каждому шагу вывода соответствует пересчет мер неточности, обусловленный соотношением на множестве высказываний (соотношением может быть элементарная логическая связь, безотносительно к тому, является ли это отношение фрагментом какого-либо правила). Таким образом, механизмы второго типа применимы не только к знаниям, выраженным в форме правил. Вместе с тем для них, как и для механизмов "присоединенного" типа, одной из главных является проблема объединения свидетельств.

2.6. МЕТОД РЕЗОЛЮЦИЙ

Основная идея метода резолюций состоит в том, чтобы проверить, содержит ли множество дизъюнктов пустой дизъюнкт. Если множество содержит пустой дизъюнкт, то оно противоречиво (невыполнимо). Если множество не содержит пустой дизъюнкт, то проверяется следующий факт: может ли пустой дизъюнкт быть получен из данного множества.

Множество содержит пустой дизъюнкт, тогда и только тогда, когда оно пустое. Если множество можно свести к пустому, то тем самым можно доказать его невыполнимость. В этом и состоит метод резолюций, который часто рассматривают как специальное правило вывода, используемое для порождения новых дизъюнктов из данного множества.

2.6.1. Метод резолюций в исчислении высказываний

Определение 24:Если A атом, то литералы A и $\neg A$ контрарны друг другу, и множество { A, $\neg A$ } называется контрарной парой.

Отметим, что дизъюнкт есть тавтология, если он содержит контрарную пару.

Определение 25: Правило резолюций состоит в следующем:

Для любых двух дизъюнктов C_1 и C_2, если существует литерал L_1 в C_1, который контрарен литералу L_2 в C_2, то вычеркнув L_1 и L_2 из C_1 и C_2 соответственно и построив дизъюнкцию оставшихся дизъюнктов, получим резолюцию (резольвенту) C_1 и C_2.

Пример 9: рассмотрим следующие дизъюнкты:

C_1: $P \vee R$,

C_2: $\neg P \vee Q$.

Дизъюнкт C_1 имеет литерал P, который контрарен литералу $\neg P$ в C_2. Следовательно, вычеркивая P и $\neg P$ из C_1 и C_2 соответственно, построим дизъюнкцию оставшихся дизъюнктов R и Q и получим резольвенту $R \vee Q$.

Важным свойством резольвенты является то, что любая резольвента двух дизъюнктов C_1 и C_2 есть логическое следствие C_1 и C_2. Это устанавливается в следующей теореме.

Теорема 4. Пусть даны два дизъюнкта C_1 и C_2. Тогда резольвента C дизъюнктов C_1 и C_2 есть логическое следствие C_1 и C_2.

Если есть два единичных дизъюнкта, то их резольвента, если она существует, есть пустой дизъюнкт $\varnothing$. Более существенно, что для невыполнимого множества дизъюнктов многократным применением правила резолюций можно породить *C*.

Определение 26: Пусть S – множество дизъюнктов. Резолютивный вывод C из S есть такая конечная последовательность $C_1, C_2, \ldots, C_k$ *дизъюнктов, что каждый* C_i *или принадлежит S или является резольвентой дизъюнктов, предшествующих* C_i, *и* $C_k=C$. *Вывод C из S называется опровержением (или доказательством невыполнимости) S.*

Пример 10. Рассмотрим множество S:

1. $\neg P \vee Q$,
2. $\neg Q$,
3. P.

Из 1 и 2 получим резольвенту

4. $\neg P$.

Из 4 и 3 получим резольвенту

5. $\Re$.

Так как $\Re$ *получается из S применениями правила резолюций , то согласно теореме 4* $\Re$ *есть логическое следствие S, следовательно S невыполнимо.*

Метод резолюций является наиболее эффективным в случае применения его к множеству Хорновских дизъюнктов.

Определение 27: Фразой называется дизъюнкт, у которого негативные литералы размещаются после позитивных литералов в конце дизъюнкта.

Пример 11: $P_1 \vee P_2 \vee \ldots P_n \vee \neg N_1 \vee \neg N_2 \ldots \vee \neg N_m$

Определение 28: Фраза Хорна это фраза, содержащая только один позитивный литерал.

Пример 12: преобразовать фразу Хорна в обратную импликацию.

$P \vee \neg N_1 \vee \neg N_2 \ldots \vee \neg N_m$

$\neg N_1 \vee \neg N_2 \ldots \vee \neg N_m = \neg (N_1 \wedge N_2 \wedge \ldots \wedge N_m)$

$P \leftarrow (N_1 \wedge N_2 \wedge \ldots \wedge N_m)$

$P \leftarrow N_1, N_2, \ldots N_m$

При представлении дизъюнктов фразами Хорна негативные литералы соответствуют гипотезам, а позитивный литерал представляет заключение. Единичный позитивный дизъюнкт представляет некоторый факт, то есть заключение, не зависящее ни от каких гипотез. Часто задача состоит в том, что надо проверить некоторую формулу, называемую целью, логически выведенную из множества правил и фактов. Резолюция является методом доказательства от противного: исходя из фактов, правил и отрицания цели, приходим к противоречию (пустому дизъюнкту).

2.6.2. Метод резолюций в исчислении предикатов. Правило унификации в логике предикатов

Правило резолюций предполагает нахождение в дизъюнкте литерала, контрарного литералу в другом дизъюнкте. Для дизъюнктов логики высказываний это очень просто. Для дизъюнктов логики предикатов процесс усложняется, так как дизъюнкты могут содержать функции, переменные и константы.

Пример 13. Рассмотрим дизъюнкты:

C_1: $P(x) \vee Q(x)$,

C_2: $\neg P(f(x)) \vee R(x)$.

Не существует никакого литерала в C_1, контрарного какому-либо литералу в C_2. Однако, если подставить $f(a)$ вместо x в C_1 и a вместо x в C_2, то исходные дизъюнкты примут вид:

C_1': $P(f(a)) \vee Q(f(a))$,

C_2': $\neg P(f(a)) \vee R(a)$.

Так как $P(f(a))$ контрарен $\neg P(f(a))$, то можно получить резольвенту

$C_3^{'}$: $Q(f(a)) \vee R(a)$.

В общем случае, подставив f(x) вместо x в C_1, получим

$C_1^{''}$: $P(f(x)) \vee Q(f(x))$.

Литерал $P(f(x))$ в $C_1^{''}$ контрарен литералу $\neg P(f(x))$ в C_2. Следовательно, можно получить резольвенту

C_3: $Q(f(x)) \vee R(x)$.

Таким образом, если подставлять подходящие термы вместо переменных в исходные дизъюнкты, можно порождать новые дизъюнкты. Отметим, что дизъюнкт C_3 из примера 13 является наиболее общим дизъюнктом в том смысле, что все другие дизъюнкты, порожденные правилом резолюции будут частным случаем данного дизъюнкта.

Определение 29: Подстановка θ– это конечное множество вида $\{t_1/v_1,\ldots,t_n/v_n\}$, где каждая v_i – переменная, каждый t_i – терм, отличный от v_i, все v_i различны.

Определение 30: Подстановка θ называется унификатором для множества $\{E_1,\ldots, E_k\}$ тогда и только тогда, когда $E_1\theta=E_2\theta=\ldots E_k\theta$. Множество $\{E_1,\ldots, E_k\}$ унифицируемо, если для него существует унификатор.

Прежде чем применить правило резолюции в исчислении предикатов переменные в литералах необходимо унифицировать.

Унификация производится при следующих условиях:

1. Если термы константы, то они унифицируемы тогда и только тогда, когда они совпадают.
2. Если в первом дизъюнкте терм переменная, а во втором константа, то они унифицируемы, при этом вместо переменной подставляется константа.
3. Если терм в первом дизъюнкте переменная и во втором дизъюнкте терм тоже переменная, то они унифицируемы.

4. Если в первом дизъюнкте терм переменная, а во втором - употребление функции, то они унифицируемы, при этом вместо переменной подставляется употребление функции.
5. Унифицируются между собой термы, стоящие на одинаковых местах в одинаковых предикатах.

Пример 14. Рассмотрим дизъюнкты:

1. Q(a, b, c) и Q(a, d, l). Дизъюнкты не унифицируемы.

2. Q(a, b, c) и Q(x, y, z). Дизъюнкты унифицируемы. Унификатор - Q(a, b, c).

Определение 31: Унификатор σ для множества $\{E_1,\ldots, E_k\}$ будет наиболее общим унификатором тогда и только тогда, когда для каждого унификатора θ для этого множества существует такая подстановка λ, что $\theta=\sigma \circ \lambda$, то есть θ является композицией подстановок σ и λ.

Определение 32: Композицией подстановок σ и λ есть функция $\sigma \circ \lambda$, определяемая следующим образом $(\sigma \circ \lambda)\ [t]=\sigma[\ \lambda[t]]$, где t – терм, σ и λ - подстановки, а $\lambda[t]$ – терм, который получается из t путем применения к нему подстановки λ.

Определение 33: Множество рассогласований непустого множества дизъюнктов $\{E_1,\ldots, E_k\}$ получается путем выявления первой (слева) позиции, на которой не для всех дизъюнктов из E стоит один и тот же символ, и выписывания из каждого дизъюнкта терма, который начинается с символа, занимающего данную позицию. Множество термов и есть множество рассогласований в E.

Пример 15. Рассмотрим дизъюнкты:

{P(x, f(y, z)), P(x, a), P(x, g(h(k(x))))}.

Множество рассогласований состоит из термов, которые начинаются с пятой позиции и представляет собой множество {f(x, y), a, g(h(k(x)))}.

2.6.3. Алгоритм унификации для нахождения наиболее общего унификатора

Пусть E – множество дизъюнктов, D – множество рассогласований, k – номер итерации, σ_k наиболее общий унификатор на k-ой итерации.

Шаг 1. Присвоим k=0, σ_k=e (пустой унификатор), E_k=E.

Шаг 2. Если для E_k не существует множества рассогласований D_k, то остановка: σ_k – наиболее общий унификатор для E. Иначе найдем множество рассогласований D_k.

Шаг 3. Если существуют такие элементы v_k и t_k в D_k, что v_k переменная, не входящая в терм t_k, то перейдем к шагу 4. В противном случае остановка: E не унифицируемо.

Шаг 4. Пусть $\sigma_{k+1}=\sigma_k \{ t_k / v_k\}$, заменим во всех дизъюнктах E_k t_k на v_k.

Шаг5. K=k+1. Перейти к шагу 2.

Пример 16. Рассмотрим дизъюнкты:

$E=\{P(f(a), g(x)), P(y, y)\}$.

$E_0=E, k=0, \sigma_0=e$.

$D_0=\{f(a),y\}, v_0=y, t_0=f(a)$.

$\sigma_1==\{f(a)/y\}, E_1=\{P(f(a), g(x)), P(f(a), f(a))\}$.

$D_1=\{g(x),f(a)\}$.

Нет переменной в множестве рассогласований D_1.

Следовательно, алгоритм унификации завершается, множество E – не унифицируемо.

С помощью унификации можно распространить правило резолюций на исчисление предикатов. При унификации возникает одна трудность: если один их термов есть переменная x, а другой терм содержит x, но не сводится к x, унификация невозможна. Проблема решается путем переименования переменных таким образом, чтобы унифицируемые дизъюнкты не содержали одинаковых переменных.

Определение 34: Если два или более литерала (с одинаковым знаком) дизъюнкта C имеют наиболее общий унификатор σ, то $C\sigma$ - называется склейкой C.

Пример 16. Рассмотрим дизъюнкты:

Пусть $C= P(x)\vee P(f(y))\vee \neg Q(x)$.

Тогда 1 и 2 литералы имеют наиболее общий унификатор $\sigma = \{f(y)/x\}$. *Следовательно,* $C\sigma=P(f(y))\vee \neg Q(f(y))$ *есть склейка C.*

Определение 35: Пусть C_1 *и* C_2 *– два дизъюнкта, которые не имеют никаких общих переменных. Пусть* L_1 *и* L_2 *- два литерала в* C_1 *и* C_2 *соответственно. Если* L_1 *и* $\neg L_2$ *имеют наиболее общий унификатор* σ, *то дизъюнкт* $(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$ *называется резольвентой* C_1 *и* C_2.

Пример 17:Пусть $C_1= P(x)\vee Q(x)$ *и* $C_2= \neg P(a)\vee R(x)$. *Так как x входит в* C_1 *и* C_2, *то мы заменим переменную в* C_2 *и пусть* $C_2= \neg P(a)\vee R(y)$. *Выбираем* $L_1= P(x)$ *и* $L_2=\neg P(a)$. L_1 *и* L_2 *имеют наиболее общий унификатор* $\sigma=\{a/x\}$. *Следовательно,* $Q(a)\vee R(y)$ *– резольвента* C_1 *и* C_2.

2.6.4. Алгоритм метода резолюций

Шаг 1. Если в S есть пустой дизъюнкт, то множество невыполнимо, иначе перейти к шагу 2.

Шаг 2. Найти в исходном множестве S такие дизъюнкты или склейки дизъюнктов C_1 *и* C_2, *которые содержат унифицируемые литералы* $L_1 \in C_1$ *и* $L_2 \in C_2$. *Если таких дизъюнктов нет, то исходное множество выполнимо, иначе перейти к шагу 3.*

Шаг 3. Вычислить резольвенту C_1 *и* C_2 *и добавить ее в множество S. Перейти к шагу 1.*

Язык логического программирования ПРОЛОГ.

Проиллюстрируем принцип логического программирования на простом примере: запишем известный метод вычисления наибольшего

общего делителя двух натуральных чисел – алгоритм Евклида в виде Хорновских дизъюнктов. При этом примем новую форму записи фразы Хорна, например $\neg P \wedge \neg Q \wedge \neg R \wedge S$ будем записывать как *S: - P, Q, R*. Тогда алгоритм Евклида можно записать в виде трех фраз Хорна:

NOD (x, x, x): -.

NOD (x, y, z): - B (x, y), NOD (f (x, y), y, z).

NOD (x, y, z): -B (y, x), NOD (x, f (y, x), z).

Предикат *NOD* – определяет наибольший общий делитель *z* для натуральных чисел *x* и *y*, предикат *B* – определяет отношение «больше», функция *f* – определяет операцию вычитания. Если мы заменим предикат *B* и функцию *f* обычными символами, то фразы примут вид:

NOD (x, x, x): -.

NOD (x, y, z): - x>y, NOD ((x- y), y, z).

NOD (x, y, z): -y>x, NOD (x, (y- x), z).

Для вычисления наибольшего общего делителя двух натуральных чисел, например 4 и 6, добавим к описанию алгоритма четвертый дизъюнкт:

ℜ: - NOD (4, 6, z).

Последний дизъюнкт – это цель, которую мы будем пытаться вывести из первых трех дизъюнктов.

2.7. ПРЕДСТАВЛЕНИЕ НЕОПРЕДЕЛЕННОСТИ ЗНАНИЙ И ДАННЫХ

2.7.1. Источники неопределенности

При решении проблем мы часто встречаемся со множеством источников неопределенности используемой информации, но в большинстве случаев их можно разделить на две категории: *недостаточно полное знание предметной области* и *недостаточная информация о конкретной ситуации.*

Теория предметной области (т.е. наши знания об этой области) может быть неясной или неполной: в ней могут использоваться недостаточно четко сформулированные концепции или недостаточно изученные явления. Например, в диагностике психических заболеваний существует несколько отличающихся теорий о происхождении и симптоматике шизофрении.

Неопределенность знаний приводит к тому, что правила влияния даже в простых случаях не всегда дают корректные результаты. Располагая неполным знанием, мы не можем уверенно предсказать, какой эффект даст то или иное действие. Например, терапия, использующая новые препараты, довольно часто дает совершенно неожиданные результаты. И, наконец, даже когда мы располагаем достаточно полной теорией предметной области, эксперт может посчитать, что эффективнее использовать не точные, а эвристические методы. Так, методика устранения неисправности в электронном блоке путем замены подозрительных узлов оказывается значительно более эффективной, чем скрупулезный анализ цепей в поиске детали, вышедшей из строя.

Но помимо неточных знаний, неопределенность может быть внесена и неточными или ненадежными данными о конкретной ситуации. Любой сенсор имеет ограниченную разрешающую способность и отнюдь не стопроцентную надежность. При составлении отчетов могут быть допущены ошибки или в них могут попасть недостоверные сведения. На практике далеко не всегда можно получить полные ответы на поставленные вопросы и хотя можно воспользоваться различного рода дополнительной информацией о пациенте, например с помощью дорогостоящих процедур или хирургическим путем, такие методики используются крайне редко из-за высокой стоимости и рискованности. Помимо всего прочего, существует еще и фактор времени. Не всегда есть возможность быстро получить необходимые данные, когда ситуация

требует принятия срочного решения. Если работа ядерного реактора вызывает подозрение, вряд ли кто-нибудь будет ждать окончания всего комплекса проверок, прежде чем принимать решение о его остановке.

Суммируя все сказанное, отметим, что эксперты пользуются неточными методами по двум главным причинам:

- точных методов не существует;
- точные методы существуют, но не могут быть применены на практике из-за отсутствия необходимого объема данных или невозможности их накопления по соображениям стоимости, риска или из-за отсутствия времени на сбор необходимой информации.

Большинство исследователей, занимающихся проблемами искусственного интеллекта, давно пришли к единому мнению, что неточные методы играют важную роль в разработке экспертных систем, но много споров вызывает вопрос, какие именно методы должны использоваться. До последнего времени многие соглашались с утверждениями Мак-Карти и Хейеса, что теория вероятности не является адекватным инструментом для решения задач представления неопределенности знаний и данных. Выдвигались следующие аргументы в пользу такого мнения:

- теория вероятности не дает ответа на вопрос, как комбинировать вероятности с количественными данными;
- назначение вероятности определенным событиям требует информации, которой мы просто не располагаем.

Другие исследователи прибавляли к этим аргументам свои:

- непонятно, как количественно оценивать такие часто встречающиеся на практике понятия, как "в большинстве случаев", "в редких случаях", или такие приблизительные оценки, как "старый" или "высокий";

– применение теории вероятности требует "слишком много чисел", что вынуждает инженеров давать точные оценки тем параметрам, которые они не могут оценить;
– обновление вероятностных оценок обходится очень дорого, поскольку требует большого объема вычислений.

Все эти соображения породили новый формальный аппарат для работы с неопределенностями, который получил название *нечеткая логика (fuzzy logic)* или *теория функций доверия (belieffunctions).* Этот аппарат широко используется при решении задач искусственного интеллекта и особенно при построении экспертных систем.

2.7.2. Интеллектуальные системы и теория вероятностей

В этом разделе будут рассмотрены те аспекты теории вероятностей, которые имеют отношение к представлению неопределенностей. Мы начнем с понятия *условной вероятности* и остановимся на тех причинах, по которым вероятностный подход критикуется большинством исследователей, занимающихся СИИ.

Условная вероятность

Условная вероятность события *d* при данном *s* — это вероятность того, что событие *d* наступит при условии, что наступило событие *s*. Например, вероятность того, что пациент действительно страдает заболеванием d, если у него (или у нее) обнаружен только симптом *s*.

В традиционной теории вероятностей для вычисления условной вероятности события *d* при данном *s* используется следующая формула:

$$P(d|s)=(d\wedge s)/P(S) \qquad (2.1)$$

Как видно, условная вероятность определяется в терминах совместимости событий. Она представляет собой отношение вероятности совпадения событий *d* и *s* к вероятности появления события *s*. Из формулы (9.1) следует, что

$P(d \wedge s)=P(d|s)P(d)$.

Если разделить обе части на $P(s)$ и подставить в правую часть (2.1), то получим *правило Байеса* в простейшем виде:

$$P(d|s)=(s|d)P(d)/P(S) \qquad (2.2)$$

Это правило, которое иногда называют *инверсной формулой для условной вероятности,* позволяет определить вероятность $P(d \mid s)$ появления события d при условии, что произошло событие s через известную условную вероятность $P(s \mid d)$. В полученном выражении $P(d)$ — априорная вероятность наступления события d, а $P(d \mid s)$ — апостериорная вероятность, т.е. вероятность того, что событие d произойдет, если известно, что событие s свершилось.

Для систем, основанных на знаниях, формула (2.2) гораздо удобнее формулы (2.1), в чем вы сможете убедиться в дальнейшем.

Предположим, что у пациента имеется некоторый симптом заболевания, например боль в груди, и желательно знать, какова вероятность того, что этот симптом является следствием определенного заболевания, например инфаркта миокарда или перикардита (воспаление каверн в легких), или чего-нибудь менее серьезного, вроде несварения желудка. Для того чтобы вычислить вероятность *P(инфаркт миокарда боль в груди)* по формуле (2.1), нужно знать (или оценить каким-либо способом), сколько человек в мире страдают таким заболеванием и сколько человек *и* больны инфарктом миокарда, *и* жалуются на боль в груди (т.е. имеют такой же симптом). Как правило, такая информация отсутствует, особенно последняя, которая нужна для вычисления вероятности *P (инфаркт миокарда | боль в груди).* Таким образом, определение, данное формулой (2.1), в клинической практике не может быть использовано.

Отмеченная сложность получения нужной информации явилась причиной негативного отношения многих специалистов по

искусственному интеллекту к вероятностному подходу вообще. Это негативное отношение подкреплялось тем, что в большинстве классических работ по теории вероятностей понятие вероятности определялось как *объективная частотность* (частота появления при достаточно продолжительных независимых испытаниях).

Однако существует мнение, что эти базовые предположения небесспорны с точки зрения практических приложений. Сторонники такого подхода придерживаются *субъективистской* точки зрения на определение вероятности, который позволяет иметь дело с оценками совместного появления событий, а не с действительной частотой. Такой взгляд на вещи связывает вероятность смеси событий с субъективной верой в то, что событие действительно наступит.

Например, врач может не знать или не иметь возможности вычислить, какая часть пациентов, жалующихся на боль в груди, страдает инфарктом миокарда, но на основании собственного опыта он может оценить, у какой части его пациентов, страдающих этим заболеванием, встречался такой симптом. Следовательно, он может оценить значение вероятности *P(боль в груди | инфаркт миокарда).* Субъективный взгляд на природу вероятности тесно связан с правилом Байеса по следующей причине. Предположим, мы располагаем достаточно достоверной оценкой вероятности *P(s | a),* где 5 означает симптом, *a d*— заболевание. Тогда по формуле (2.2) можно вычислить вероятность *P(d\ s).* Оценку вероятности *P(d)* можно взять из публикуемой медицинской статистики, а оценить значение *P(s)* врач может на основании собственных наблюдений.

Вычисление *P(d | s)* не вызывает затруднений, когда речь идет о единственном симптоме, т.е. имеется множество заболеваний *D* и множество симптомов *S,* причем для каждого члена из *D* нужно вычислить условную вероятность того, что у пациентов, страдающих этим заболеванием, наблюдался *один определенный* симптом из множества *S.*

Тем не менее, если в множестве *D* имеется *m* членов, а в множестве *S*— *n* членов, потребуется вычислить *mn* + *m* + *n* оценок вероятностей. Это отнюдь не простая работа, если в системе медицинской диагностики используется до 2000 видов заболеваний и огромное число самых разнообразных симптомов.

Но ситуация значительно усложняется, если мы попробуем включить в процесс составления диагноза не один симптом, а несколько.

В более общей форме правило Байеса имеет вид

$P(d|s1\wedge...\wedge sk) = P(s1\wedge...\wedge s_k|d)P(d)/P(s1\wedge...\wedge sk)$ (2.3)

и требует вычисления $(mn)^k + m + n^k$ оценок вероятностей, что даже при небольшом значении А; очень много. Эти оценки вероятностей требуются нам по той причине, что в общем случае для вычисления $P(s1 \wedge ....\wedge s_k)$ нужно предварительно вычислить произведения вида

$P(s_1 | s_2 \wedge.. .\wedge s_k)P(s_2 | s_3 \wedge.. .\wedge s_K)... P(s_k)$.

Однако, если предположить, что некоторые симптомы независимы друг от друга, объем вычислений существенно снижается. Независимость любой пары симптомов Si, и *Sj* означает, что

$P(S_i)=P(S_l|S_j),$

из чего следует соотношение

$P(S_i\wedge S_j)=P(Si)P(S_j).$

Если *все* симптомы независимы, то объем вычислений будет таким же, как и в случае учета при диагнозе единственного симптома.

Но, даже если это и не так, в большинстве случаев можно предположить наличие *условной независимости.* Это означает, что пара симптомов *s*\ и *Sj* является независимой, поскольку в нашем распоряжении имеются какие-либо дополнительные свидетельства на этот счет или фундаментальные знания *E.* Таким образом,

$P(Si|Sj,E)=P(Si|E).$

Например, если в моем автомобиле нет горючего и не работает освещение, я могу смело сказать, что эти симптомы независимы, поскольку моих познаний в устройстве автомобиля вполне достаточно, чтобы предположить, что между ними нет никакой причинной связи. Но если автомобиль не заводится и не работает освещение, то заявлять, что эти симптомы независимы, нельзя, поскольку они могут быть следствием одной и той же неисправности аккумуляторной батареи. Степень доверия к симптому "не работает освещение" только увеличится, если обнаружится, что к тому же и двигатель не заводится. Необходимость отслеживать такого рода связи в программе и соответственно корректировать степень доверия к симптомам значительно увеличивает объем вычислений в общем случае.

Таким образом, использование теории вероятности ставит перед нами следующие проблемы, которые лучше всего сформулировать в терминах задачи выбора:

- *либо* априори предполагается, что все данные независимы, и использовать менее трудоемкие методы вычислений, за что придется платить снижением достоверности результатов;
- *либо* нужно организовать отслеживание зависимости между используемыми данными, количественно оценить эту зависимость, реализовать оперативное обновление соответствующей нормативной информации, т.е. усложнить вычисления, но получить более достоверные результаты.

Коэффициенты уверенности

В идеальном мире можно вычислить вероятность $P(di| E)$, где di - i-я диагностическая категория, а £ представляет все необходимые дополнительные свидетельства или фундаментальные знания, используя только вероятности $P(di \mid Sj)$, где Sj является j-м клиническим наблюдением (симптомом). Мы уже имели возможность убедиться в том,

что правило Байеса позволяет выполнить такие вычисления только в том случае, если, во-первых, доступны все значения $P(sj \mid di)$, и, во-вторых, правдоподобно предположение о взаимной независимости симптомов.

В системе MYCIN применен альтернативный подход на основе правил влияния, которые следующим образом связывают имеющиеся данные (свидетельства) с гипотезой решения:

ЕСЛИ

пациент имеет показания и симптомы $s1 \wedge \ldots \wedge s_k$ *и имеют место определенные фоновые условия* $t1 \wedge \ldots \wedge f_m$,

ТО

можно с уверенностью m заключить, что пациент страдает заболеванием di.

Коэффициент-уверенности t принимает значения в диапазоне [–1, +1]. Если $t = +1$, то это означает, что при соблюдении всех оговоренных условий составитель правила абсолютно уверен в *правильности* заключения *di,* а если $t = -1$, то значит, что при соблюдении всех оговоренных условий существует абсолютная уверенность в *ошибочности* этого заключения. Отличные от +1 положительные значения коэффициента указывают на степень уверенности в *правильности* заключения *di,* а отрицательные значения — на степень уверенности в его *ошибочности.*

Основная идея состоит в том, чтобы с помощью порождающих правил такого вида попытаться заменить вычисление $P(di \mid s1 \wedge \ldots \wedge sk)$ приближенной оценкой и таким образом сымитировать процесс принятия решения экспертом-человеком. Как было показано в главе 3, результаты применения правил такого вида связываются с коэффициентом уверенности окончательного заключения с помощью CF(a) — коэффициент уверенности в достоверности значения параметра *a,* а дополнительные условия $t_1 \wedge \ldots \wedge t_m$ представляют фоновые знания, которые

ограничивают применение конкретного правила. Чаще всего оказывается, что эти условия могут быть интерпретированы значениями "истина" или "ложь", т.е. соответствующие коэффициенты принимают значение +1 или -1. Таким образом, отличные от единицы значения коэффициентов характеризуют только симптомы $s_1, \ldots, s_k$. Роль фоновых знаний состоит в том, чтобы разрешить или запретить применение правила в данном конкретном случае. Пусть, например, имеется диагностическое правило, связывающее появление болей в брюшной полости с возможной беременностью. Применение этого правила блокируется фоновым знанием, что оно справедливо только по отношению к пациентам-женщинам.

Бучанан и Шортлифф утверждают, что, строго говоря, применение правила Байеса в любом случае не позволяет получить точные значения, поскольку используемые условные вероятности субъективны. Как мы уже видели, это основной аргумент против применения вероятностного подхода. Однако такая аргументация предполагает *объективистскую* интерпретацию понятия вероятности, т.е. предполагается, что "правильные" значения все же существуют, но мы не можем их получить, а раз так, то и правило Байеса нельзя использовать. Этот аргумент имеет явно схоластический оттенок, поскольку любая экспертиза, проводимая инженером по знаниям, совершенно очевидно сводится к представлению тех знаний о предметной области, которыми обладает человек-эксперт (эти знания, конечно же, являются субъективными), а не к воссозданию абсолютно адекватной модели мира. С точки зрения теории представляется, что целесообразнее использовать математически корректный формализм к неточным данным, чем формализм, который математически некорректен, к тем же неточным данным.

Перл обратил внимание на важное практическое достоинство подхода, основанного на правилах. Вычисление коэффициентов уверенности

заключения имеет явно выраженный модульный характер, поскольку не нужно принимать во внимание никакой иной информации, кроме той, что имеется в данном правиле. При этом не имеет никакого значения, как именно получены коэффициенты уверенности, характеризующие исходные данные.

При построении экспертных систем часто используется эта особенность. Полагается, что для всех правил, имеющих дело с определенным параметром, предпосылки каждого правила логически независимы. Анализируя систему MYCIN, Шортлифф посоветовал сгруппировать все зависимые признаки в единое правило, а не распределять их по множеству правил.

Пусть, например, существует зависимость между признаками E_1 и E_2- Шортлифф рекомендует сгруппировать их в единое правило если E_1 и E_2, то приходим к заключению *H* с уверенностью т, а не распределять по двум правилам если E_1, то приходим к заключению *H с* уверенностью t, если E_2, то приходим к заключению *H с* уверенностью t.

В основе этой рекомендации лежит одно из следствий теории вероятностей, гласящее, что $P(H \mid E_1, E_2)$ не может быть простой функцией от $P(H \mid E_1)$ и $P(H \mid E_2)$.

Выражения для условной вероятности не могут в этом смысле рассматриваться как модульные. Выражение

$P(B \mid A) = t$

не позволяет заключить, что $P(B) = t$ при наличии *A,* если только *A* не является *единственным известным* признаком. Если кроме *A* мы располагаем еще и знанием *E,* то нужно сначала вычислить $P(B \mid A, E)$, а уже потом можно будет что-нибудь сказать и о значении $P(B)$. Такая чувствительность к контексту может стать основой очень мощного механизма логического вывода, но, как уже не раз подчеркивалось, за это придется платить существенным повышением сложности вычислений.

Коэффициенты уверенности и условные вероятности

Адаме показал, что если используется простая вероятностная модель на основе правила Байеса, то в системе MYCIN коэффициенты уверенности гипотез не соответствуют вероятностям гипотез при заданных признаках. На первый взгляд, если коэффициенты уверенности используются только для упорядочения альтернативных гипотез, это не очень страшно. Но Адаме также показал, что возможна ситуация, когда при использовании коэффициентов уверенности две гипотезы будут ранжированы в обратном порядке по отношению к соответствующим вероятностям. Рассмотрим этот вопрос подробнее.

Обозначим через *P(h) субъективное,* т.е. составленное на основе заключения эксперта, значение вероятности того, что гипотеза *h* справедлива, т.е. значение *P(H)* отражает степень уверенности эксперта в справедливости гипотезы h. Усложним положение дел и добавим новый признак *e* в пользу этой гипотезы, такой что $P(h \mid e) > P(h)$. Степень доверия эксперта к справедливости гипотезы увеличится, и это увеличение выразится отношением

$MB(h,e)= [P(h|e)-P(h)]/[1-P(h)]$

где MB означает относительную *меру доверия.*

Если же признак *e* свидетельствует против гипотезы h, т.е. $P(h \mid e) < P(h)$, то увеличится *мера недоверия* эксперта к справедливости этой гипотезы. Меру недоверия MD можно выразить следующим отношением:

$MD(h, e) =[P(h)-P(h|e)] / P(h)]$

Адаме обратил внимание на то, что уровни доверия к одной и той же гипотезе с учетом разных дополнительных признаков не могут быть определены независимо. Если некоторый признак является абсолютным диагностическим индикатором конкретного заболевания, т.е. если все пациенты с симптомом s_1 страдают заболеванием *dj,* то никакие другие признаки уже не могут изменить диагноз, т.е. уровень доверия к

выдвинутой гипотезе. Другими словами, если существует пара признаков s_1 и s_2 и

$P(d_i|s_1)=P(d_i|S_1 \wedge S_2)=1,$

то

$P(d_i|s_2)= P(d_l).$

Адаме также критически отнесся к объединению (конъюнкции) гипотез. Модель, положенная в основу MYCIN, предполагает, что уровень доверия к сочетанию гипотез $d_1 \wedge d_2$ должен соответствовать *наименьшему* из уровней доверия отдельных гипотез, а уровень недоверия — *наибольшему* из уровней *недоверия* отдельных гипотез. Предположим, что гипотезы d_1 и d_2 не только не независимы, но и взаимно исключают друг друга. Тогда $P(d_1 \wedge d_2 \mid e) = 0$ при наличии любого признака e и независимо от степени доверия или недоверия к d_1 или d_2

Бучанан и Шортлифф определили коэффициент уверенности как некий артефакт, который позволяет численно оценить комбинацию уровней доверия или недоверия к гипотезам. Он представляет собой разницу между мерой доверия и недоверия:

$CF(h, e_a \wedge e_f) = MB(h, e_f) - MD(h, e_a),$

где e_f— признак, свидетельствующий в пользу гипотезы h, а e_a — признак, свидетельствующий против гипотезы h. Однако полученное таким образом значение отнюдь не эквивалентно условной вероятности существования гипотезы h при условии $e_a \wedge e_f$, которое следует из правила Байеса:

$P(h|e_a \wedge e_f)=[P(e_a \wedge e_f \mid h)P(h)]/[P(e_a \wedge e_f)]$

Таким образом, хотя степень доверия, связанная с определенным правилом, и может быть соотнесена с субъективной оценкой вероятности, коэффициент уверенности является комбинированной оценкой. Его основное назначение состоит в следующем:

– управлять ходом выполнения программы при формировании суждений;
– управлять процессом поиска цели в пространстве состояний: если коэффициент уверенности гипотезы оказывается в диапазоне [+0.2, -0.2], то поиск блокируется;
– ранжировать набор гипотез после обработки всех признаков.

Адаме, однако, показал, что ранжирование гипотез на основе коэффициентов уверенности может дать результат, противоположный тому, который будет получен при использовании вероятностных методов. Он продемонстрировал это на следующем примере.

Положим, что d_1и d_2 — это две гипотезы, а e — признак, свидетельствующий как в пользу одной гипотезы, так и в пользу другой. Пусть между априорными вероятностями существует отношение $P(d_1) > P(d_2)$ и $P(d\backslash \backslash e) > P(d_2 \mid e)$. Другими словами, субъективная вероятность справедливости гипотезы $d\backslash$ больше, чем гипотезы d_2, причем это соотношение сохраняется и после того, как во внимание принимается дополнительный признак. Адаме показал, что при этих условиях возможно обратное соотношение $CF(d_1, e) < CF(d_2, e)$ между коэффициентами уверенности гипотез.

Предположим, что вероятности имеют следующие значения:

$P(d_1) = 0.8$,

$P(d_2) = 0.2$,

$P(d_1|e) = 0.9$,

$P(d_2| e) = 0.8$.

Тогда повышение доверия к d_1 будет равно (0.9 - 0.8) / 0.2 = 0.5, а повышение доверия к

d_2 — (0.8 - 0.2) / 0.8 = 0.75.

Отсюда следует, что $CF(d_1| e) < CF(d_2, e)$, несмотря на то, что и $P(d_1 | e) > P(d_2| e)$.

Адаме назвал это явление "нежелательным свойством" коэффициентов доверия. Избежать такой ситуации можно, если все априорные вероятности будут равны. Несложно показать, что эффект в приведенном выше примере явился следствием того, что признак *e* больше свидетельствовал в пользу гипотезы d_2, чем в пользу *d1,* именно из-за более высокой априорной вероятности последней. Однако приравнивание априорных вероятностей явно не согласуется со стилем мышления тех, кто ставит диагноз, поскольку существует достаточно большое отличие в частоте сочетаний разных болезней с одинаковыми симптомами, следовательно, эксперты будут присваивать им совершенно разные значения субъективных вероятностей.

Последовательное применение правил в системе MYCIN также связано с существованием определенных теоретических проблем. Используемая при этом функция комбинирования основана на предположении, что если признак *e* влияет на некоторую промежуточную гипотезу *h* с вероятностью $P(h \mid e)$, а гипотеза *h* входит в окончательный диагноз *d* с вероятностью $P(d \mid h)$, то

$$P(d|e) = P(d|h)P(h|e).$$

Таким образом, создается впечатление, что транзитивное отношение в последовательности правил вывода суждений справедливо на первом шаге, но не справедливо в общем случае. Для того чтобы существовала связь между правилами, популяции, связанные с этими категориями, должны быть вложены примерно так, как на рисунке 2.1.

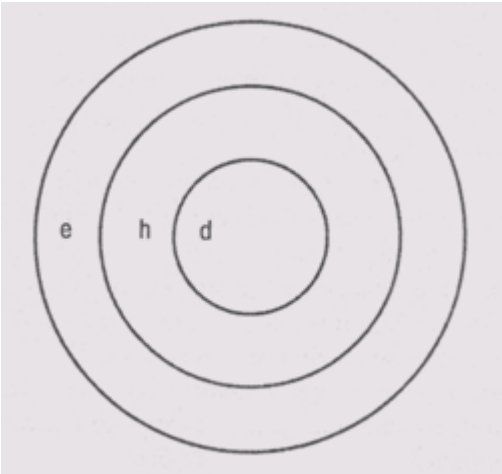

Рисунок 2.1 – Популяции, позволяющие использовать P(d | e; = P(d| h)P(h| z)

Адаме пришел к выводу, что успех практического применения системы MYCIN и других подобных систем объясняется тем, что в них используются довольно короткие последовательности комбинирования правил, а рассматриваемые гипотезы довольно просты.

Другое критическое замечание относительно MYCIN было высказано Горвицем и Гекерманом и касается использования коэффициентов уверенности в качестве меры *изменения* доверия, в то время как в действительности они устанавливаются экспертами в качестве степени *абсолютного* доверия. Связывая коэффициенты доверия с правилами, эксперт отвечает на вопрос: "Насколько вы уверены в правдоподобности того или иного заключения?" При применении в MYCIN функций комбинирования дополнительных признаков эти коэффициенты становятся мерой *обновления* степени доверия, что приводит к несовместимости этих значений с теоремой Байеса.

2.7.3. Сомнительность и возможность

Помимо использования коэффициентов уверенности, в литературе описаны и иные подходы, альтернативные вероятностному. В частности, много внимания уделяется *нечеткой логике (fuzzy logic)* и *теории функций*

доверия (belieffunctions). В данном разделе читатель познакомится с основными аспектами нечеткой логики. Будет показано, почему подход, основанный на идеях нечеткой логики, в последнее время все шире используется при создании СИИ.

Нечеткие множества

То знание, которое использует эксперт при оценке признаков или симптомов, обычно базируется скорее на отношениях между классами данных и классами гипотез, чем на отношениях между отдельными данными и конкретными гипотезами. Большинство методик решения проблем в той или иной форме включает классификацию данных (сигналов, симптомов и т.п.), которые рассматриваются как конкретные представители некоторых более общих категорий. Редко, когда эти более общие категории могут быть четко очерчены. Конкретный объект может обладать частью характерных признаков определенной категории, а частью не обладать, принадлежность конкретного объекта к определенному классу может быть размыта. Предложенная Заде *теория нечетких множеств (fuzzy set theory)* представляет собой формализм, предназначенный для формирования суждений о таких категориях и принадлежащих к ним объектах. Эта теория лежит в основе *нечеткой логики (fuzzy logic)* и *теории возможностей (possibility theory).*

Классическая теория множеств базируется на двузначной логике. Выражения в форме *a & A,* где *a* представляет индивидуальный объект, а *A* — множество подобных объектов, могут принимать только значение "истина" либо "ложь". После появления понятия "нечеткое множество" прежние классические множества иногда стали называть *жесткими.* Жесткость классической теории множеств стала источником ряда проблем при попытке применить ее к нечетко определенным категориям.

Рассмотрим категорию, определенную словом "быстрый" (fast). Если применить это определение к автомобилям, то какой автомобиль можно

считать быстрым? В классической теории мы можем определить множество *A* "быстрых автомобилей" либо перечислением (составив список всех членов множества), либо введя в рассмотрение некоторую характеристическую функцию f такую, что для любого объекта *X*

f(X) = истина тогда и только тогда, когда *X принадлежит A.*

Например, эта функция может отбирать только те автомобили, которые имеют скорость более 150 миль в час:

GT150(X)={ истина,если CAR(X) и TOP_SPEED(X) > 150 ложь в противном случае

Множество, определенное такой характеристической функцией, представляется формулой

{X ~ CAR TOP-SPEED(X)> 150}.

Эта формула утверждает, что элементами нового множества являются те элементы множества *CAR,* которые имеют максимальную скорость свыше 150 миль в час.

А что можно сказать о множестве (категории) "быстрых" автомобилей? Интуитивно кажется, что ситуация сходна с представленной на рисунке 2.2, где границы множества размыты и принадлежность элементов множеству может быть каким-то образом ранжирована.

В таком случае можно говорить о том, что отдельный объект (автомобиль) *более или менее типичен* для этого множества (категории). Можно с помощью некоторой функции/охарактеризовать степень принадлежности объектов *X* такому множеству. Функция */(X)* определена на интервале [0, 1]. Если для объекта *X* функция f*(X)* = 1, то объект *определенно является* членом множества, если ДА) = 0, то объект *определенно не является* членом множества. Все промежуточные значения означают *степень членства* объекта *X* в этом множестве. В примере с автомобилями нам понадобится функция, оперирующая с максимальной скоростью каждого претендента на членство.

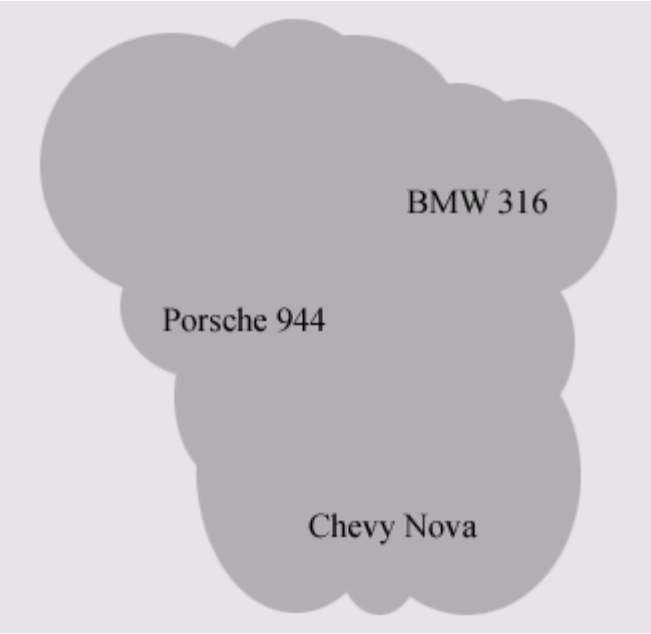

Рисунок 2.2 – Нечеткое множество "быстрых" автомобилей

Можно определить ее таким образом, что $f_{FAST}(80) = 0$, $f_{FAST}(180) = 1$, а промежуточные значения представляются некоторой монотонной гистограммой, имеющей значения в интервале между нулем и единицей. Тогда множество "быстрых автомобилей" может быть охарактеризовано функцией

$f_{FAST\ CAR}(X) = f_{FAST}(TOP\text{-}SPEED(X)),$

которая определена на множестве всех автомобилей. Таким образом, членами множества становятся пары *(объект, степень),* например:

FAST-CAR = {(Porche-944, 0.9),

(BMW-316, 0.5), *(Chevy-Nova,* 0.1)}.

Нечеткая логика

Ту роль, которую в классической теории множеств играет двузначная булева логика, в теории нечетких множеств играет многозначная *нечеткая логика,* в которой предположения о принадлежности объекта множеству, например *FAST-CAR(Porche-944),* могут принимать действительные значения в интервале от 0 до 1. Возникает вопрос, а как, используя

концепцию неопределенности, вычислить значение истинности сложного выражения, такого как

¬FAST¬CAR(Chevy-Nova).

По аналогии с теорией вероятности, если *F* представляет собой нечеткий предикат, операция отрицания реализуется по формуле

¬F(X)=1-F(X).

Но аналоги операций конъюнкции и дизъюнкции в нечеткой логике не имеют никакой связи с теорией вероятностей. Рассмотрим следующее выражение:

"Porche 944 является быстрым (fast), представительским (pretentious) автомобилем". В классической логике предположение

FAST-CAR(Porche-944) ^PRETENTIOUS-CAR(Porche-944)

является истинным в том и только в том случае, если истинны оба члена конъюнкции. В нечеткой логике существует соглашение: если *F* и G являются нечеткими предикатами, то

Таким образом, если

FAST-CAR(Porche-944) = 0.9

PRETENTIOUS-CAR(Porche-944) = 0.7,

то

FAST-CAR(Porche-944) ^ PRETENTIOUS-CAR(Porche-944) = 0.7.

А теперь рассмотрим выражение

FAST-CAR(Porche-944) ^ ¬FAST-CAR(Porche-944).

Вероятность истинности этого утверждения равна 0, поскольку

P(FAST-CAR(Porche-944) | ¬FAST-CAR(Porche-944)) = 0,

но в нечеткой логике значение этого выражения будет равно 0,1. Какой смысл имеет это значение. Его можно считать показателем принадлежности автомобиля к нечеткому множеству среднескоростных автомобилей, которые в чем-то близки к быстрым, а в чем-то – к медленным.

Смысл выражения *FAST-CAR(Porche-944)* = 0.9 заключается в том, что мы только на 90% уверены в принадлежности этого автомобиля к быстрым именно из-за неопределенности самого понятия "быстрый автомобиль". Вполне резонно предположить, что существует некоторая уверенность в том, что *Porche-944* не принадлежит к быстрым, например, он медленнее автомобиля, принимающего участие в гонках "Формула-1".

Аналог операции дизъюнкции в нечеткой логике определяется следующим образом:

$f_{(F \vee G)}(X) = \max(f_F(X), f_G(X))$.

Здесь также очевидна полная противоположность с теорией вероятностей, в которой

$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$.

Рассмотрим следующие предположения и значения истинности их принадлежности к нечеткому множеству *FAST-CAR:*

FAST-CAR(Porche-944) v ¬*FAST-CAR(Porche-944)* = 0.9,

FAST-CAR(BMW-316) v *FAST-CAR(BMW-316)* = 0.5,

FAST-CAR(Chevy-Nova) v *FAST-CAR(Chevy-Novd)* = 0.9.

Значение вероятности истинности каждого из этих предположений, как это определено в теории вероятностей, равно 1. В нечеткой логике более высокие значения для автомобилей *Porche-944* и *Chevy-Nova* объясняются тем фактом, что степень принадлежности каждого из этих объектов к нечеткому множеству *FAST-CAR* выше. Нечеткость концепции "быстрый или не быстрый" более благоприятна для них, чем для более медленного *BMW-316,* который "ни рыба, ни мясо".

Операторы обладают свойствами коммутативности, ассоциативности и взаимной дистрибутивности. Как к операторам в стандартной логике, к ним применим принцип *композитивности,* т.е. значения составных выражений вычисляются только по значениям выражений-компонентов. В этом операторы нечеткой логики составляют полную противоположность

законам теории вероятностей, согласно которым при вычислении вероятностей конъюнкции и дизъюнкции величин нужно принимать во внимание условные вероятности

2.7.4. Теория возможности

Нечеткая логика имеет дело с ситуациями, когда и сформулированный-вопрос, и знания, которыми мы располагаем, содержат нечетко очерченные понятия. Однако нечеткость формулировки понятий является не единственным источником неопределенности. Иногда мы просто не уверены в самих фактах. Если утверждается: "Возможно, что Джон сейчас в Париже", то говорить о нечеткости понятий Джон и Париж не приходится. Неопределенность заложена в самом факте, действительно ли Джон находится в Париже.

Теория возможностей является одним из направлений в нечеткой логике, в котором рассматриваются точно сформулированные вопросы, базирующиеся на неточных знаниях. В этом разделе вы познакомитесь только с основными идеями этой теории. Лучше всего это сделать на примере.

Предположим, что в ящике находится 1 0 шаров, но известно, что только *несколько* из них красных. Какова вероятность того, что на удачу из ящика будет вынут красный шар?

Просто вычислить искомое значение, основываясь на знаниях, что только несколько шаров *красные* (red), нельзя. Тем не менее для каждого значения X из $P(RED)$ в диапазоне [0,1] можно следующим образом вычислить возможность, что $P(RED) = X$.

Во-первых, определим "несколько" (several) как нечеткое множество, например, так:

$f_{SEVERAL} = \{(3, 0.2), (4, 0.6), (5, 1.0), (6, 1.0), (7, 0.6), (8, 0.3)\}$.

В этом определении выражение (3, 0.2) *e* $f_{SEVERAL}$ означает, что 3 из 10 вряд ли можно признать как "несколько", а выражения (5, 1 .0) e $f_{SEVERAL}$ и (6, 1 .0) e $f_{SEVERAL}$ означают, что значения 5 и 6 из 10 идеально согласуются с понятием "несколько". Обратите внимание на то, что в определение нечеткого множества не входят значения 1 и 10, поскольку интуитивно ясно, что "несколько" означает "больше одного" и "не все". Нечеткое множество, определенное на множестве чисел, называется *нечеткими числами (fuzzy numbers).* По тому же принципу, что и множество $f_{SEVERAL}$, можно определить нечеткие множествами/для понятия "мало" f_{MOST} для понятия "почти".

Теперь распределение возможностей для *P(RED)* представляется формулой

$f_{P(RED)} = SEVERAL / 10,$

которая после подстановки дает

{(0.3, 0.2), (0.4, 0.6), (0.5, 1.0), (0.6, 1.0), (0.7, 0.6), (0.8, 0.3)}.

Выражение (0.3, 0.2) ~ $f_{P(RED)}$ означает, что шанс на то, что *P(RED)* = 0.3, равен 20%. Можно рассматривать $f_{P(RED)}$ как *нечеткую вероятность (fuzzy probability).*

Полагая, что почти любое понятие может быть областью определения такой функции, естественно ввести в обиход и понятие "нечеткое значение правдоподобия". Мы часто оцениваем некоторое утверждение как "очень правдоподобное" или "частично правдоподобное". Таким образом, можно представить себе нечеткое множество

f_{true}-: [0 , 1]-> [0, 1],

где и область определения, и область значений функции f_{true} являются возможными значениями правдоподобия в нечеткой логике. Следовательно, можно получить

TRUE(FASR-CAR(Porsche-944)) = 1

даже при *FASR-CAR(Porsche-944) = 0.9,* поскольку (0.9, 1.0)~f_{true} Это означает, что любое предположение относительно значения 0.9 рассматривается как "достаточно правдоподобное". Таким образом, можно с уверенностью сказать, что Porsche-944 является быстрым автомобилем, несмотря на то, что на рынке есть и более скоростные

2.7.5. Неопределенное состояние проблемы неопределенности

Одно из главных достоинств формализма нечеткой логики состоит в возможности комбинирования его логических операторов. Ранее отмечалось, что для правила MYCIN:

ЕСЛИ

пациент имеет показания и симптомы $s_1 \wedge \ldots \wedge s_k$ *и*

имеют место определенные фоновые условия $t_1 \wedge \ldots \wedge t_m$,

ТО можно с уверенностью m заключить, что пациент страдает заболеванием d_i

Оценка набора симптомов $s_1 \wedge \ldots \wedge s_k$, в соответствии с аксиомами теории вероятностей, включает вычисление произведений вида

$P(s_1 \mid s_2 \wedge .. , \wedge s_k)P(s_2 \mid s_3 \wedge .. . \wedge s_k)\ldots P(s_k)$.

Такая операция в худшем случае требует вычисления *k-1* оценки вероятностей свыше тех, что необходимы для s_i.

Было также показано, что в MYCIN конъюнкция интерпретируется как оператор нечеткой логики, — при этом вычисляется $min\ (s_1 \wedge \ldots \wedge s_k)$. Это может иногда привести к результатам, полностью противоположным тем, которые следуют из теории вероятностей. Прк сравнении результатов, полученных с помощью различных методов обработки неопределенности в практических системах, были найдены и другие примеры ошибочных выводов. Это сравнение показало, что методы, основанные на нечеткой логике, менее надежны, чем те, которые используют Байесовский подход (см., например, *[Wise and Henrion, 1986]).*

С другой стороны, нелишне отметить, что человеку также не свойственно строить суждения на основе Байесовского подхода. Исследования Канемана и Тверского показали, что люди склонны не принимать во внимание прежний опыт и отдавать предпочтение более свежей информации *[Kahneman and Tversky, 1972].* Некоторые исследователи полагают, что людям свойственно переоценивать свою компетентность (см., например, статьи в сборнике *[Kahneman et al, 1982]),* причем большинство имеют слабое представление о теории оценок *[Tversky and Kahneman, 1974].*

Частично привлекательность нечеткой логики для проектировщиков экспертных систем состоит в ее близости к естественному языку. Таким терминам, как "быстрый", "немного", "правдоподобно", чаще всего дается интерпретация на основе повседневного опыта и интуиции. Это упрощает процесс инженерии знаний, поскольку подобные суждения человека-эксперта можно непосредственно преобразовать в выражения нечеткой логики.

Здесь были изложены только основные идеи, чтобы читатель мог получить первое представление о концепции неопределенности знаний и данных и связанных с этим проблемах. Но даже из этого краткого изложения ясно, что предстоит еще очень много сделать для того, чтобы иметь полное понятие об адекватном представлении неопределенности в технических системах.

3. ТЕХНОЛОГИИ ЭКСПЕРТНЫХ СИСТЕМ

3.1. НАЗНАЧЕНИЕ, КЛАССИФИКАЦИЯ И ПРИНЦИПЫ ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ

3.1.1. Назначение экспертных систем

В начале 1980-х в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название "экспертные системы" (ЭС). Цель исследований по ЭС состоит в разработке программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Исследователи в области ЭС для названия своей дисциплины часто используют также термин "инженерия знаний", введенный Е. Фейгенбаумом как "привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов".

Программные средства (ПС), базирующиеся на технологии экспертных систем, или инженерии знаний (в дальнейшем будем использовать их как синонимы), получили значительное распространение в мире. Важность экспертных систем состоит в следующем:

- технология экспертных систем существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект;
- технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования: длительность и, следовательно, высокая стоимость разработки сложных приложений;

- высокая стоимость сопровождения сложных систем, которая часто в несколько раз превосходит стоимость их разработки; низкий уровень повторной используемости программ и т.п.;
- объединение технологии ЭС с технологией традиционного программирования добавляет новые качества к программным продуктам. Это достигается за счет: обеспечения динамичной модификации приложений пользователем, а не программистом; большей "прозрачности" приложения (например, знания хранятся на ограниченном ЕЯ, что не требует комментариев к знаниям, упрощает обучение и сопровождение); лучшей графики; интерфейса и взаимодействия.

По мнению ведущих специалистов, в недалекой перспективе ЭС найдут следующее применение:

- ЭС будут играть ведущую роль во всех фазах проектирования, разработки, производства, распределения, продажи, поддержки и оказания услуг;
- технология ЭС, получившая коммерческое распространение, обеспечит революционный прорыв в интеграции приложений из готовых интеллектуально-взаимодействующих модулей.
- ЭС предназначены для так называемых неформализованных задач, т.е. ЭС не отвергают и не заменяют традиционного подхода к разработке программ, ориентированного на решение формализованных задач.
- Неформализованные задачи обычно обладают следующими особенностями:
- ошибочностью, неоднозначностью, неполнотой и противоречивостью исходных данных;
- ошибочностью, неоднозначностью, неполнотой и противоречивостью знаний о проблемной области и решаемой задаче;
- большой размерностью пространства решения, т.е. перебор при поиске решения весьма велик;

- динамически изменяющимися данными и знаниями.

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач. Многие специалисты считают, что эти задачи являются наиболее массовым классом задач, решаемых ЭВМ.

Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма).

Экспертные системы применяются для решения только трудных практических (не игрушечных) задач. По качеству и эффективности решения экспертные системы не уступают решениям эксперта-человека. Решения экспертных систем обладают *"прозрачностью",* т.е. могут быть объяснены пользователю на качественном уровне. Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях. Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом. Необходимо отметить, что в настоящее время технология экспертных систем используется для решения различных типов задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление) в самых разнообразных проблемных областях, таких, как финансы, нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, целлюлозно-бумажная промышленность, телекоммуникации и связь и др.

Коммерческие успехи к фирмам-разработчикам систем искусственного интеллекта (СИИ) пришли не сразу. На протяжении 1960 - 1985 гг. успехи ИИ касались в основном исследовательских разработок, которые демонстрировали пригодность СИИ для практического

использования. Начиная примерно с 1985 г. (в массовом масштабе с 1988 - 1990 гг.), в первую очередь ЭС, а в последние годы системы, воспринимающие естественный язык (ЕЯ-системы), и нейронные сети (НС) стали активно использоваться в коммерческих приложениях.

Следует обратить внимание на то, что некоторые специалисты (как правило, специалисты в программировании, а не в ИИ) продолжают утверждать, что ЭС и СИИ не оправдали возлагавшихся на них ожиданий и умерли. Причины таких заблуждений состоят в том, что эти авторы рассматривали ЭС как альтернативу традиционному программированию, т.е. они исходили из того, что ЭС в одиночестве (в изоляции от других программных средств) полностью решают задачи, стоящие перед заказчиком. Надо отметить, что на заре появления ЭС специфика используемых в них языков, технологии разработки приложений и используемого оборудования (например, Lisp-машины) давала основания предполагать, что интеграция ЭС с традиционными, программными системами является сложной и, возможно, невыполнимой задачей при ограничениях, накладываемых реальными приложениями. Однако в настоящее время коммерческие инструментальные средства (ИС) для создания ЭС разрабатываются в полном соответствии с современными технологическими тенденциями традиционного программирования, что снимает проблемы, возникающие при создании интегрированных приложений.

Причины, приведшие СИИ к коммерческому успеху, следующие.

Интегрированность. Разработаны инструментальные средства искусственного интеллекта (ИС ИИ), легко интегрирующиеся с другими информационными технологиями и средствами (с CASE, СУБД, контроллерами, концентраторами данных и т.п.).

Открытость и переносимость. ИС ИИ разрабатываются с соблюдением стандартов, обеспечивающих открытость и переносимость [14].

Использование языков традиционного программирования и рабочих станций. Переход от ИС ИИ, реализованных на языках ИИ (Lisp, Prolog и т.п.), к ИС ИИ, реализованным на языках традиционного программирования (C, C++ и т.п.), упростил обеспечение интегрированности, снизил требования приложений ИИ к быстродействию ЭВМ и объемам оперативной памяти. Использование рабочих станций (вместо ПК) резко увеличило круг приложений, которые могут быть выполнены на ЭВМ с использованием ИС ИИ.

Архитектура клиент-сервер. Разработаны ИС ИИ, поддерживающие распределенные вычисления по архитектуре клиент-сервер, что позволило:

- снизить стоимость оборудования, используемого в приложениях;
- децентрализовать приложения;
- повысить надежность и общую производительность (так как сокращается количество информации, пересылаемой между ЭВМ, и каждый модуль приложения выполняется на адекватном ему оборудовании).

Проблемно/предметно-ориентированные ИС ИИ. Переход от разработок ИС ИИ общего назначения (хотя они не утратили свое значение как средство для создания ориентированных ИС) к проблемно/предметно-ориентированным ИС ИИ [9] обеспечивает: сокращение сроков разработки приложений; увеличение эффективности использования ИС; упрощение и ускорение работы эксперта; повторную используемость информационного и программного обеспечения (объекты, классы, правила, процедуры).

3.1.2. Классификация экспертных систем

Общепринятая классификация экспертных систем отсутствует, однако наиболее часто экспертные системы различают по назначению, предметной области, методам представления знаний, динамичности и сложности (рисунок 3.1).

Рисунок 3.1 – Классификация экспертных систем

По **назначению** классификацию экспертных систем можно провести следующим образом:

- диагностика состояния систем, в том числе мониторинг (непрерывное отслеживание текущего состояния);

- прогнозирование развития систем на основе моделирования прошлого и настоящего;
- планирование и разработка мероприятий в организационном и технологическом управлении;
- проектирование или выработка четких предписаний по построению объектов, удовлетворяющих поставленным требованиям;
- автоматическое управление (регулирование);
- обучение пользователей и др.

По **предметной области** наибольшее количество экспертных систем используется в военном деле, геологии, инженерном деле, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельском хозяйстве, управлении процессами, физике, филологии, химии, электронике, юриспруденции.

Классификация экспертных систем по **методам представления знаний** делит их на традиционные и гибридные. Традиционные экспертные системы используют, в основном, эмпирические модели представления знаний и исчисление предикатов первого порядка. Гибридные экспертные системы используют все доступные методы, в том числе оптимизационные алгоритмы и концепции баз данных.

По **степени сложности** экспертные системы делят на поверхностные и глубинные. Поверхностные экспертные системы представляют знания в виде правил «ЕСЛИ-ТО». Условием выводимости решения является безразрывность цепочки правил. Глубинные экспертные системы обладают способностью при обрыве цепочки правил определять (на основе метазнаний) какие действия следует предпринять для продолжения решения задачи. Кроме того, к сложным относятся предметные области, в которых текст записи одного правила на естественном языке занимает более 1/3 страницы.

Классификация экспертных систем по **динамичности** делит экспертные системы на статические и динамические. Предметная область называется статической, если описывающие ее исходные данные не изменяются во времени. Статичность области означает неизменность описывающих ее исходных данных. При этом производные данные (выводимые из исходных) могут и появляться заново, и изменяться (не изменяя, однако, исходных данных).

Если исходные данные, описывающие предметную область, изменяются за время решения задачи, то предметную область называют динамической. В архитектуру динамической экспертной системы, по сравнению со статической, вводятся два компонента:

- подсистема моделирования внешнего мира;
- подсистема связи с внешним окружением.

Последняя подсистема осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической экспертной системы (база знаний и механизм логического вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

3.1.3. Структура экспертных систем

Типичная статическая ЭС состоит из следующих основных компонентов (рисунок 3.2):

- решателя (интерпретатора);
- рабочей памяти (РП), называемой также базой данных (БД);
- базы знаний (БЗ);
- компонентов приобретения знаний;
- объяснительного компонента;
- диалогового компонента.

База данных (рабочая память) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

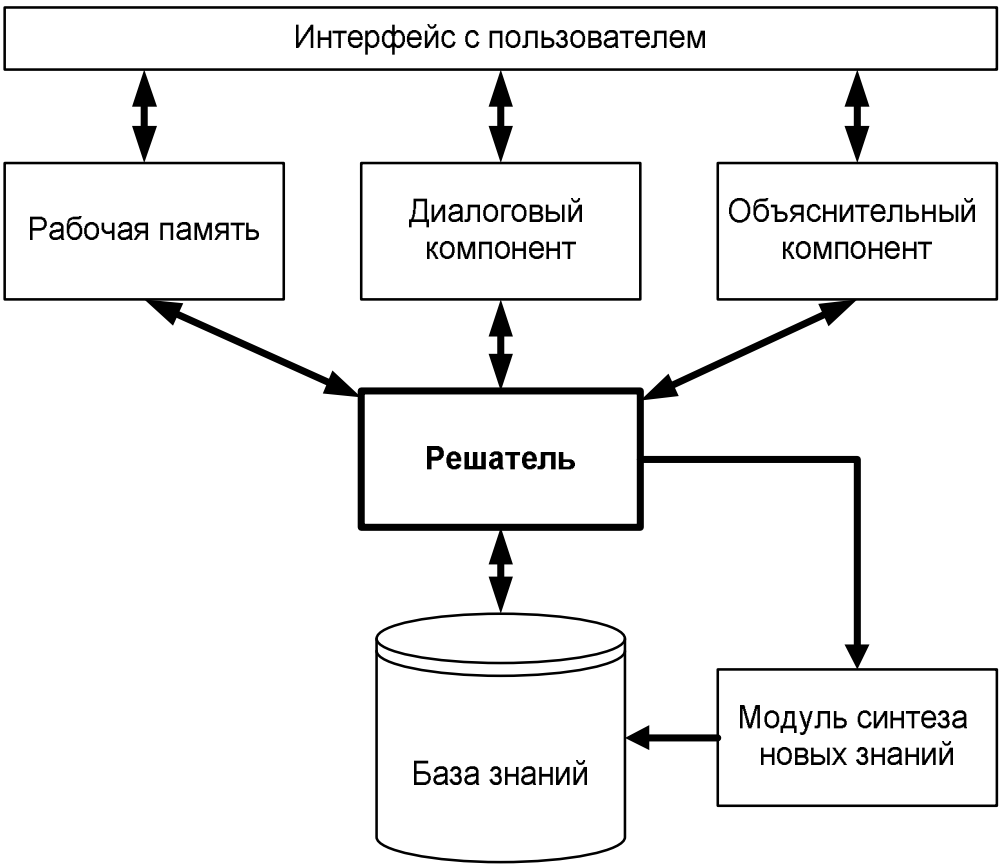

Рисунок 3.2 - Структура статической ЭС

База знаний (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

Решатель, используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

Компонент приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

Объяснительный компонент объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

Диалоговый компонент ориентирован на организацию дружественного общения с пользователем, как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

В разработке ЭС участвуют представители следующих специальностей:

- эксперт в проблемной области, задачи которой будет решать ЭС;
- инженер по знаниям - специалист по разработке ЭС (используемые им технологию, методы называют технологией (методами) инженерии знаний);
- программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС. Он же осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

В режиме приобретения знаний общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

В режиме консультации общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам), или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу). В режиме

консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения:

"Почему система задает тот или иной вопрос?", "как ответ, собираемый системой, получен?".

Структуру, приведенную на рисунке 3.3, *называют структурой динамической ЭС.*

Рисунок 3.3 – Архитектура динамической ЭС

ЭС данного типа используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими.

На рисунке 3.3 показано, что в архитектуру динамической ЭС по сравнению со статической ЭС вводятся два компонента: подсистема

моделирования внешнего мира и подсистема связи с внешним окружением. Последняя подсистема осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической ЭС (база знаний и машина вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

Подчеркнем, что структура ЭС, представленная на рисунках 3.2 и 3.3, отражает только компоненты (функции), и многое остается "за кадром". На рисунке 3.3 приведена обобщенная структура современного ИС для создания динамических ЭС, содержащая кроме основных компонентов те возможности, которые позволяют создавать интегрированные приложение в соответствии с современной технологией программирования.

3.2. РАЗРАБОТКА ЭКСПЕРТНЫХ СИСТЕМ

3.2.1. Методология разработки экспертных систем

Рассмотрим методику формализации экспертных знаний на примере создания экспертных диагностических систем (ЭДС).

Целью создания ЭДС является определение состояния объекта диагностирования (ОД) и имеющихся в нем неисправностей.

Состояниями ОД могут быть: исправно, неисправно, работоспособно. Неисправностями, например, радиоэлектронных ОД являются обрыв связи, замыкание проводников, неправильное функционирование элементов и т.д.

Число неисправностей может быть достаточно велико (несколько тысяч). В ОД может быть одновременно несколько неисправностей. В этом случае говорят, что неисправности кратные.

Введем следующие определения. Разные неисправности ОД проявляются во внешней среде информационными параметрами. Совокупность значений информационных параметров определяет

«информационный образ» (ИО) неисправности ОД. ИО может быть полным, то есть содержать всю необходимую информацию для постановки диагноза, или, соответственно, неполным. В случае неполного ИО постановка диагноза носит вероятностный характер.

Основой для построения эффективных динамических ЭС являются знания эксперта для постановки диагноза, записанные в виде информационных образов, а также система представления знаний, встраиваемая в информационные системы обеспечения функционирования и контроля ОД, интегрируемые с соответствующей технической аппаратурой.

Для описания своих знаний эксперт с помощью инженера по знаниям должен выполнить следующее.

Выделить множество всех неисправностей ОД, которые должна различать ЭДС.

Выделить множество информативных (существенных) параметров, значения которых позволяют различить каждую неисправность ОД и поставить диагноз с некоторой вероятностью.

Для выбранных параметров следует выделить информативные значения или информативные диапазоны значений, которые могут быть как количественными, так и качественными. Например, точные количественные значения могут быть записаны: задержка 25 нсек, задержка 30 нсек и т.д. Количественный диапазон значений может быть записан: задержка 25 – 40 нсек, 40 – 50 нсек, 50 нсек и выше. Качественный диапазон значений может быть записан: индикаторная лампа светится ярко, светится слабо, не светится.

Для более удобного дальнейшего использования качественный диапазон значений может быть закодирован, например, следующим образом:

светится ярко P1 = +++ (или P1 = 3),

светится слабо P1 = ++ (или P1 = 2),

не светится P1 = + (или P1 = 1).

Процедура получения информации по каждому из параметров определяется индивидуально в каждой конкретной системе диагностирования. Эта процедура может заключаться в автоматическом измерении параметров в ЭДС, в ручном измерении параметра с помощью приборов, качественном определении параметра, например, светится слабо, и т.д.

Процедура создания полных или неполных ИО каждой неисправности в алфавите значений информационных параметров может быть определена следующим образом. Составляются диагностические правила, определяющие вероятный диагноз на основе различных сочетаний диапазонов значений выбранных параметров ОД. Правила могут быть записаны в различной форме. Ниже приведена форма записи правил в виде таблицы 3.1.

Таблица 5.1 – Диагностические правила

Номер	P1	P2	P3	Диагноз	Вероятность диагноза	Примечания
1		+++		Неисправен блок A1	0.95	
2	12-15	+		Неисправен блок A2	0.80	

Для записи правил с учетом изменений по времени следует ввести еще один параметр P0 - время (еще один столбец в таблице). В этом случае диагноз может ставиться на основе нескольких строк таблицы, а в графе Примечания могут быть указаны использованные тесты. Диагностическая таблица в этом случае представлена в таблице 3.2.

Таблица 3.2 – Динамические диагностические правила

Номер	P0	P1	P2	P3	Диагноз	Вероятность	Примечания

						диагноза	
1	12:00	+	+	+			тест Т1
2	12:15	++	++	+	Неисправен блок А3	0.90	

Для записи последовательности проведения тестовых процедур и задания ограничений (если они есть) на их проведение может быть предложен аналогичный механизм.

Механизм записи последовательности проведения тестовых процедур в виде правил реализуется, например, следующим образом:

ЕСЛИ: Р2 = 1

ТО: тест = Т1, Т3, Т7

где Т1, Т3, Т7 - тестовые процедуры, подаваемые на ОД при активизации (срабатывании) соответствующей продукции.

В современных динамических ЭС применяются различные стратегии поиска решения и постановки диагноза, которые позволяют определить необходимые последовательности тестовых процедур. Однако приоритет в ЭС отдается, прежде всего, знаниям и опыту, а лишь затем логическому выводу.

Данная методика будет применена в следующей лекции при создании экспертной системы управления технологическим процессом.

3.2.2. Этапы разработки экспертных систем

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать ЭС следует только тогда, когда разработка ЭС *возможна, оправдана и* методы инженерии знаний *соответствуют*

решаемой задаче. Чтобы разработка ЭС была *возможной* для данного приложения, необходимо одновременное выполнение по крайней мере следующих требований:

1) существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

2) эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

3) эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;

4) решение задачи требует только рассуждений, а не действий;

5) задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);

6) задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задачи;

7) решение задачи не должно в значительной степени использовать "здравый смысл" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в данном приложении может быть возможно, но не оправдано. Применение ЭС может быть *оправдано* одним из следующих факторов:

- решение задачи принесет значительный эффект, например, экономический;

- использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
- использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации;
- использование ЭС целесообразно при необходимости решать задачу в окружении, враждебном для человека.

Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

1) задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;

2) задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;

3) задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;

4) задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция "быстрого прототипа". Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать

типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально распараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером по знаниям и программистом). Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов (рисунок 3.4): идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию. На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

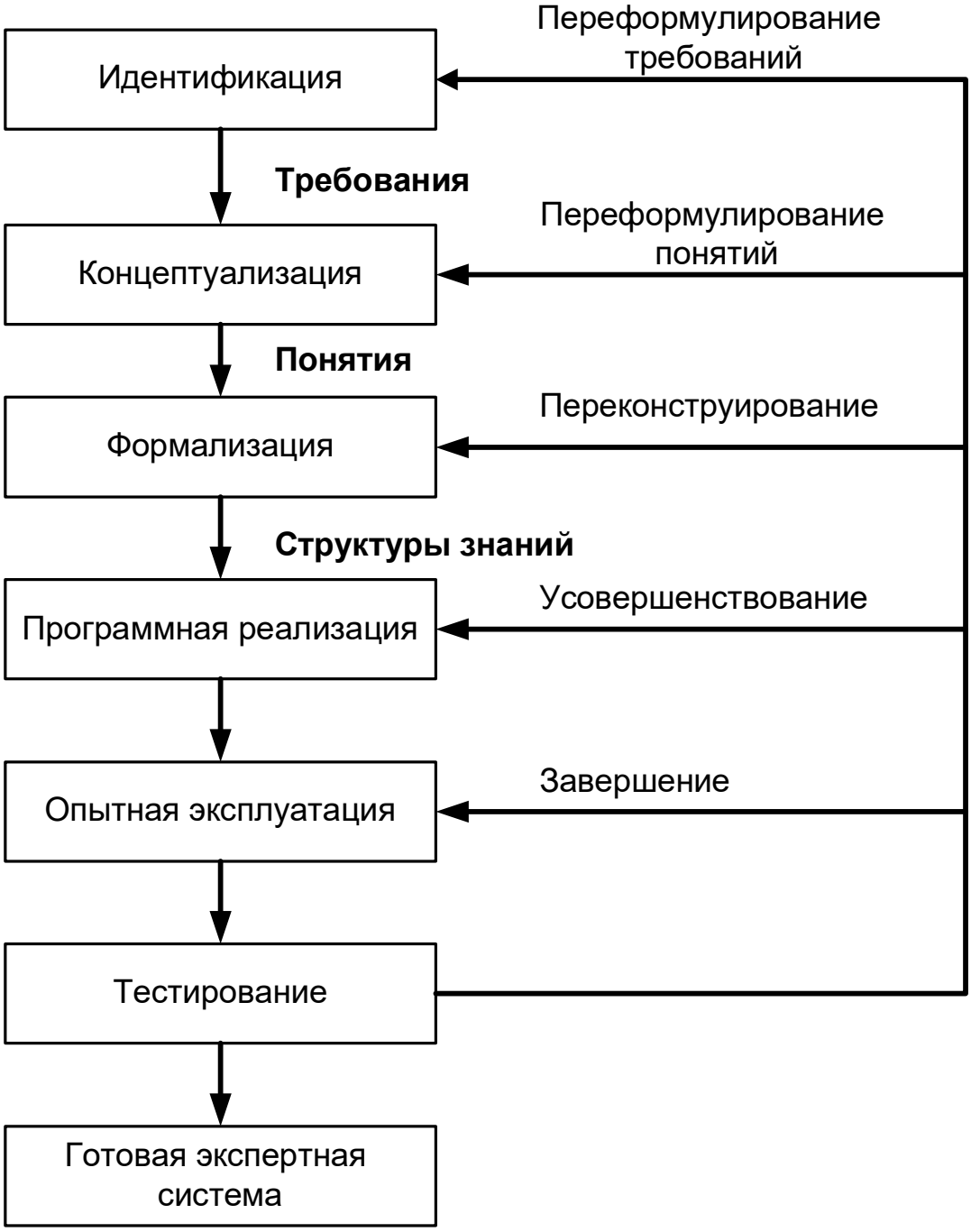

Рисунок 3.4 – Технология разработки ЭС

На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе *формализации* выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия,

определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе *выполнения* осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач.

3.2.3. Методы поиска решений в экспертных системах

Методы решения задач, основанные на сведении их к поиску, зависят от особенностей предметной области, в которой решается задача, и от требований, предъявляемых пользователем к решению. Особенности предметной области с точки зрения методов решения можно характеризовать следующими параметрами:

- размер, определяющий объем пространства, в котором предстоит искать решение;
- изменяемость области, характеризует степень изменяемости области во времени и пространстве (здесь будем выделять статические и динамические области);
- полнота модели, описывающей область, характеризует адекватность модели, используемой для описания данной области. Обычно если модель не полна, то для описания области используют несколько моделей, дополняющих друг друга за счет отражения различных свойств предметной области;

- определенность данных о решаемой задаче, характеризует степень точности (ошибочности) и полноты (неполноты) данных. Точность (ошибочность) является показателем того, что предметная область с точки зрения решаемых задач описана точными или неточными данными; под полнотой (неполнотой) данных понимается достаточность (недостаточность) входных данных для однозначного решения задачи.

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать количеством решений и свойствами результата и (или) способом его получения. Параметр "количество решений" может принимать следующие основные значения: одно решение, несколько решений, все решения. Параметр "свойства" задает ограничения, которым должен удовлетворять полученный результат или способ его получения. Так, например, для системы, выдающей рекомендации по лечению больных, пользователь может указать требование не использовать некоторое лекарство (в связи с его отсутствием или в связи с тем, что оно противопоказано данному пациенту). Параметр "свойства" может определять и такие особенности, как время решения ("не более чем", "диапазон времени" и т.п.), объем памяти, используемой для получения результата, указание об обязательности (невозможности) использования каких-либо знаний (данных) и т.п.

Итак, сложность задачи, определяемая вышеприведенным набором параметров, варьируется от простых задач малой размерности с неизменяемыми определенными данными и отсутствием ограничений на результат и способ его получения до сложных задач большой размерности с изменяемыми, ошибочными и неполными данными и произвольными ограничениями на результат и способ его получения. Из общих соображений ясно, что каким-либо одним методом нельзя решить все

задачи. Обычно одни методы превосходят другие только по некоторым из перечисленных параметров.

Рассмотренные ниже методы могут работать в статических и динамических проблемных средах. Для того чтобы они работали в условиях динамики, необходимо учитывать время жизни значений переменных, источник данных для переменных, а также обеспечивать возможность хранения истории значений переменных, моделирования внешнего окружения и оперирования временными категориями в правилах.

Существующие методы решения задач, используемые в экспертных системах, можно классифицировать следующим образом:

- методы поиска в одном пространстве - методы, предназначенные для использования в следующих условиях: области небольшой размерности, полнота модели, точные и полные данные;
- методы поиска в иерархических пространствах - методы, предназначенные для работы в областях большой размерности;
- методы поиска при неточных и неполных данных;
- методы поиска, использующие несколько моделей, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Предполагается, что перечисленные методы при необходимости должны объединяться для того, чтобы позволить решать задачи сложность которых возрастает одновременно по нескольким параметрам.

3.2.4. Трудности разработки экспертных систем

Разработка ЭС связана с определенными трудностями, которые необходимо хорошо знать, так же, как и способы их преодоления. Рассмотрим подробнее эти проблемы.

1. Проблема извлечения знаний экспертов. Ни один специалист никогда просто так не раскроет секреты своего профессионального мастерства, свои сокровенные знания в профессиональной области. Он должен быть заинтересован материально или морально, причем хорошо заинтересован. Никто не хочет рубить сук, на котором сидит. Часто такой специалист опасается, что, раскрыв все свои секреты, он будет не нужен компании. Вместо него будет работать экспертная система. Избежать эту проблему поможет выбор высококвалифицированного эксперта, заинтересованного в сотрудничестве.

2. Проблема формализации знаний экспертов. Эксперты-специалисты в определенной области, как правило, не в состоянии формализовать свои знания. Часто они принимают правильные решения на интуитивном уровне и не могут аргументированно объяснить, почему принято то или иное решение. Иногда эксперты не могут прийти к взаимопониманию (фраза «встретились два геолога, у них было три мнения» - не шутка, а реальная жизнь). В таких ситуациях поможет выбор эксперта, умеющего ясно формулировать свои мысли и легко объяснять другим свои идеи.

3. Проблема нехватки времени у эксперта. Выбранный для разработки эксперт не может найти достаточно времени для выполнения проекта. Он слишком занят. Он всем нужен. У него есть проблемы. Чтобы избежать этой ситуации, необходимо получить от эксперта, прежде чем начнется проект, согласие тратить на проект время в определенном фиксированном объеме.

3. Правила, формализованные экспертом, не дают необходимой точности. Проблему можно избежать, если решать вместе с экспертом реальные задачи. Не надо придумывать «игрушечных» ситуаций или задач. В условиях задач нужно использовать реальные данные, такие как лабораторные данные, отчеты, дневники и другую информацию, взятую из практических задач. Постарайтесь говорить с экспертом на одном языке,

используя единую терминологию. Эксперт, как правило, легче понимает правила, записанные на языке, близком к естественному, а не на языке типа LISP или PROLOG.

5. Недостаток ресурсов. В качестве ресурсов выступают персонал (инженеры знаний, разработчики инструментальных средств, эксперты) и средства построения ЭС (средства разработки и средства поддержки). Недостаток благожелательных и грамотных администраторов порождает скептицизм и нетерпение у руководителей. Повышенное внимание в прессе и преувеличения вызвали нереалистические ожидания, которые приводят к разочарованию в отношении экспертных систем. ЭС могут давать не самые лучшие решения на границе их применимости, при работе с противоречивыми знаниями и в рассуждениях на основе здравого смысла. Могут потребоваться значительные усилия, чтобы добиться небольшого увеличения качества работы ЭС. Экспертные системы требуют много времени на разработку. Так, создание системы PUFF для интерпретации функциональных тестов легких потребовало 5 человеко-лет, на разработку системы PROCPECTOR для разведки рудных месторождений ушло 30 человеко-лет, система XCON для расчета конфигурации компьютерных систем на основе VAX 11/780 потребовала 8 человеко-лет. ЭС последних лет разрабатываются более быстрыми темпами за счет развития технологий ЭС, но проблемы остались. Удвоение персонала не сокращает время разработки наполовину, потому что процесс создания ЭС - это процесс со множеством обратных связей. Все это необходимо учитывать при планировании создания ЭС.

6. Неадекватность инструментальных средств решаемой задаче. Часто определенные типы знаний (например, временные или пространственные) не могут быть легко представлены на одном ЯПЗ, так же, как и разные схемы представления (например, фреймы и продукции) не могут быть достаточно эффективно реализованы на одном ЯПЗ. Некоторые задачи

могут быть непригодными для решения по технологии ЭС (например, отдельные задачи анализа сцен). Необходим тщательный анализ решаемых задач, чтобы определить пригодность предлагаемых инструментальных средств и сделать правильный выбор.

3.3. ВЗАИМОДЕЙСТВИЕ ИНЖЕНЕРА ПО ЗНАНИЯМ (КОГНИТОЛОГА) С ЭКСПЕРТОМ

3.3.1. Представление знаний в экспертных системах

Первый и основной вопрос, который надо решить при представлении знаний, - это вопрос определения состава знаний, т.е. определение того, "ЧТО ПРЕДСТАВЛЯТЬ" в экспертной системе. Второй вопрос касается того, "КАК ПРЕДСТАВЛЯТЬ" знания. Необходимо отметить, что эти две проблемы не являются независимыми. Действительно, выбранный способ представления может оказаться непригодным в принципе либо неэффективным для выражения некоторых знаний.

Вопрос "КАК ПРЕДСТАВЛЯТЬ" можно разделить на две в значительной степени независимые задачи: как организовать (структурировать) знания и как представить знания в выбранном формализме.

Стремление выделить организацию знаний в самостоятельную задачу вызвано, в частности, тем, что эта задача возникает для любого языка представления и способы решения этой задачи являются одинаковыми (либо сходными) вне зависимости от используемого формализма.

Итак, в круг вопросов, решаемых при представлении знаний, будем включать следующие:

- определение состава представляемых знаний;
- организацию знаний;
- представление знаний, т.е. определение модели представления. Состав знаний ЭС определяется следующими факторами:

- проблемной средой;
- архитектурой экспертной системы;
- потребностями и целями пользователей;
- языком общения.

В соответствии с общей схемой статической экспертной системы (см. рисунок 5.2) для ее функционирования требуются следующие знания:

- знания о процессе решения задачи (т.е. управляющие знания), используемые интерпретатором (решателем);
- знания о языке общения и способах организации диалога, используемые лингвистическим процессором (диалоговым компонентом);
- знания о способах представления и модификации знаний, используемые компонентом приобретения знаний;
- поддерживающие структурные и управляющие знания, используемые объяснительным компонентом.

Для динамической ЭС, кроме того, необходимы следующие знания:

1) знания о методах взаимодействия с внешним окружением;

2) знания о модели внешнего мира.

Зависимость состава знаний от требований пользователя проявляется в следующем:

- какие задачи (из общего набора задач) и с какими данными хочет решать пользователь;
- каковы предпочтительные способы и методы решения;
- при каких ограничениях на количество результатов и способы их получения должна быть решена задача;
- каковы требования к языку общения и организации диалога;
- какова степень общности (конкретности) знаний о проблемной области, доступная пользователю;
- каковы цели пользователей.

Состав знаний о языке общения зависит как от языка общения, так и от требуемого уровня понимания.

С учетом архитектуры экспертной системы знания целесообразно делить на *интерпретируемые* и *неинтерпретируемые*. К первому типу относятся те знания, которые способен интерпретировать решатель (интерпретатор). Все остальные знания относятся ко второму типу. Решатель не знает их структуры и содержания. Если эти знания используются каким-либо компонентом системы, то он не "осознает" этих знаний. Неинтерпретируемые знания подразделяются на *вспомогательные* знания, хранящие информацию о лексике и грамматике языка общения, информацию о структуре диалога, и *поддерживающие* знания. Вспомогательные знания обрабатываются естественно-языковой компонентой, но ход этой обработки решатель не осознает, так как этот этап обработки входных сообщений является вспомогательным для проведения экспертизы. Поддерживающие знания используются при создании системы и при выполнении объяснений. Поддерживающие знания выполняют роль описаний (обоснований), как интерпретируемых знаний, так и действий системы. Поддерживающие знания подразделяются на *технологические* и *семантические.* Технологические поддерживающие знания содержат сведения о времени создания описываемых ими знаний, об авторе знаний и т.п. Семантические поддерживающие знания содержат смысловое описание этих знаний. Они содержат информацию о причинах ввода знаний, о назначении знаний, описывают способ использования знаний и получаемый эффект. Поддерживающие знания имеют описательный характер.

Интерпретируемые знания можно разделить на *предметные знания, управляющие знания* и *знания о представлении.* Знания о представлении содержат информацию о том, каким образом (в каких структурах) в системе представлены интерпретируемые знания.

Предметные знания содержат данные о предметной области и способах преобразования этих данных при решении поставленных задач.

Отметим, что по отношению к предметным знаниям знания о представлении и знания об управлении являются *метазнаниями.* В предметных знаниях можно выделить описатели и собственно предметные знания. Описатели содержат определенную информацию о предметных знаниях, такую, как коэффициент определенности правил и данных, меры важности и сложности. Предметные знания разбиваются на *факты* и *исполняемые утверждения.* Факты определяют возможные значения сущностей и характеристик предметной области. Исполняемые утверждения содержат информацию о том, как можно изменять описание предметной области в ходе решения задач. Говоря другими словами, исполняемые *утверждения* - это знания, задающие процедуры обработки. Однако мы избегаем использовать термин "процедурные знания", так как хотим подчеркнуть, что эти знания могут быть заданы не только в процедурной, но и в декларативной форме.

Управляющие знания можно разделить на *фокусирующие* и *решающие.* Фокусирующие знания описывают, какие знания следует использовать в той или иной ситуации. Обычно фокусирующие знания содержат сведения о наиболее перспективных объектах или правилах, которые целесообразно использовать при проверке соответствующих гипотез. В первом случае внимание фокусируется на элементах рабочей памяти, во втором - на правилах базы знаний. Решающие знания содержат информацию, используемую для выбора способа интерпретации знаний, подходящего к текущей ситуации. Эти знания применяются для выбора стратегий или эвристик, наиболее эффективных для решения данной задачи.

Качественные и количественные показатели экспертной системы могут быть значительно улучшены за счет использования *метазнаний,* т.е. знаний о знаниях. Метазнания не представляют некоторую единую

сущность, они могут применяться для достижения различных целей. Перечислим возможные назначения метазнаний:

1) метазнания в виде стратегических метаправил используются для выбора релевантных правил;

2) метазнания используются для обоснования целесообразности применения правил из области экспертизы;

3) метаправила используются для обнаружения синтаксических и семантических ошибок в предметных правилах;

4) метаправила позволяют системе адаптироваться к окружению путем перестройки предметных правил и функций;

5) метаправила позволяют явно указать возможности и ограничения системы, т.е. определить, что система знает, а что не знает.

Вопросы организации знаний необходимо рассматривать в любом представлении, и их решение в значительной степени не зависит от выбранного способа (модели) представления. Выделим следующие аспекты проблемы организации знаний:

- организация знаний по уровням представления и по уровням детальности;
- организация знаний в рабочей памяти;
- организация знаний в базе знаний.

3.3.2. Уровни представления и уровни детальности

Для того чтобы экспертная система могла управлять процессом поиска решения, была способна приобретать новые знания и объяснять свои действия, она должна уметь не только использовать свои знания, но и обладать способностью понимать и исследовать их. То есть экспертная система должна иметь знания о том, как представлены ее знания о проблемной среде. Если знания о проблемной среде назвать знаниями нулевого уровня представления, то первый уровень представления

содержит метазнания, т.е. знания о том, как представлены во внутреннем мире системы знания нулевого уровня. Первый уровень содержит знания о том, какие средства используются для представления знаний нулевого уровня. Знания первого уровня играют существенную роль при управлении процессом решения, при приобретении и объяснении действий системы. В связи с тем, что знания первого уровня не содержат ссылок на знания нулевого уровня, знания первого уровня независимы от проблемной среды.

Число уровней представления может быть больше двух. Второй Уровень представления содержит сведения о знаниях первого уровня, т.е. знания о представлении базовых понятий первого уровня. Разделение знаний по уровням представления обеспечивает расширение области применимости системы.

Выделение уровней детальности позволяет рассматривать знания с различной степенью подробности. Количество уровней детальности во многом определяется спецификой решаемых задач, объемом знаний и способом их представления. Как правило, выделяется не менее трех уровней детальности, отражающих соответственно общую, логическую и физическую организацию знаний. Введение нескольких уровней детальности обеспечивает дополнительную степень гибкости системы, так как позволяет производить изменения на одном уровне, не затрагивая другие. Изменения на одном уровне детальности могут приводить к дополнительным изменениям на этом же уровне , что оказывается необходимым для обеспечения согласованности структур данных и программ. Однако наличие различных уровней препятствует распространению изменений с одного уровня на другие.

3.3.3. Организация знаний в рабочей системе

Рабочая память (РП) экспертных систем предназначена для хранения данных. Данные в рабочей памяти могут быть однородны или разделяются на уровни по типам данных. В последнем случае на каждом уровне рабочей памяти хранятся данные соответствующего типа. Выделение уровней усложняет структуру экспертной системы, но делает систему более эффективной. Например, можно выделить уровень планов, уровень агенды (упорядоченного списка правил, готовых к выполнению) и уровень данных предметной области (уровень решений).

В современных экспертных системах данные в рабочей памяти рассматриваются как изолированные или как связанные. В первом случае рабочая память состоит из множества простых элементов, а во втором - из одного или нескольких (при нескольких уровнях в РП) сложных элементов (например, объектов). При этом сложный элемент соответствует множеству простых, объединенных в единую сущность. Теоретически оба подхода обеспечивают полноту, но использование изолированных элементов в сложных предметных областях приводит к потере эффективности.

Данные в РП в простейшем случае являются *константами* и (или) *переменными.* При этом переменные могут трактоваться как характеристики некоторого объекта, а константы - как значения соответствующих характеристик. Если в РП требуется анализировать одновременно несколько различных объектов, описывающих текущую проблемную ситуацию, то необходимо указывать, к каким объектам относятся рассматриваемые характеристики. Одним из способов решения этой задачи является явное указание того, к какому объекту относится характеристика.

Если РП состоит из сложных элементов, то связь между отдельными объектами указывается явно, например, заданием семантических

отношений. При этом каждый объект может иметь свою внутреннюю структуру. Необходимо отметить, что для ускорения поиска и сопоставления данные в РП могут быть связаны не только логически, но и ассоциативно.

3.3.4. Организация знаний в базе данных

Показателем интеллектуальности системы с точки зрения представления знаний считается способность системы использовать в нужный момент необходимые *(релевантные)* знания. Системы, не имеющие средств для определения релевантных знаний, неизбежно сталкиваются с проблемой "комбинаторного взрыва". Можно утверждать, что эта проблема является одной из основных причин, ограничивающих сферу применения экспертных систем. В проблеме доступа к знаниям можно выделить три аспекта: *связность знаний и данных, механизм доступа к знаниям* и *способ сопоставления.*

Связность {агрегация) знаний является основным способом, обеспечивающим ускорение поиска релевантных знаний. Большинство специалистов пришли к убеждению, что знания следует организовывать вокруг наиболее важных объектов (сущностей) предметной области. Все знания, характеризующие некоторую сущность, связываются и представляются в виде отдельного объекта. При подобной организации знаний, если системе потребовалась информация о некоторой сущности, то она ищет объект, описывающий эту сущность, а затем уже внутри объекта отыскивает информацию о данной сущности. В объектах целесообразно выделять два типа связок между элементами: *внешние* и *внутренние.* Внутренние связки объединяют элементы в единый объект и предназначены для выражения структуры объекта. Внешние связки отражают взаимозависимости, существующие между объектами в области экспертизы. Многие исследователи классифицируют внешние связки на

логические и *ассоциативные.* Логические связки выражают семантические отношения между элементами знаний. Ассоциативные связки предназначены для обеспечения взаимосвязей, способствующих ускорению процесса поиска релевантных знаний.

Основной проблемой при работе с большой базой знаний является проблема поиска знаний, релевантных решаемой задаче. В связи с тем, что в обрабатываемых данных может не содержаться явных указаний на значения, требуемые для их обработки, необходим более общий механизм доступа, чем метод прямого доступа (метод явных ссылок). Задача этого механизма состоит в том, чтобы по некоторому описанию сущности, имеющемуся в рабочей памяти, найти , базе знаний объекты, удовлетворяющие этому описанию. Очевидно, что упорядочение и структурирование знаний могут значительно ускорить процесс поиска.

Нахождение желаемых объектов в общем случае уместно рассматривать как двухэтапный процесс. На первом этапе, соответствующем процессу выбора по ассоциативным связкам, совершается предварительный выбор в базе знаний потенциальных кандидатов на роль желаемых объектов. На втором этапе путем выполнения операции сопоставления потенциальных кандидатов с описаниями кандидатов осуществляется окончательный выбор искомых объектов. При организации подобного механизма доступа возникают определенные трудности: Как выбрать критерий пригодности кандидата? Как организовать работу в конфликтных ситуациях? и т.п.

Операция сопоставления может использоваться не только как средство выбора нужного объекта из множества кандидатов; она может быть использована для классификации, подтверждения, декомпозиции и коррекции. Для идентификации неизвестного объекта он может быть сопоставлен с некоторыми известными образцами. Это позволит классифицировать неизвестный объект как такой известный образец, при

сопоставлении с которым были получены лучшие результаты. При поиске сопоставление используется для подтверждения некоторых кандидатов из множества возможных. Если осуществлять сопоставление некоторого известного объекта с неизвестным описанием, то в случае успешного сопоставления будет осуществлена частичная декомпозиция описания.

Операции сопоставления весьма разнообразны. Обычно выделяют следующие их формы: *синтаксическое, параметрическое, семантическое* и *принуждаемое сопоставления.* В случае *синтаксического сопоставления* соотносят формы (образцы), а не содержание объектов. Успешным является сопоставление, в результате которого образцы оказываются идентичными. Обычно считается, что переменная одного образца может быть идентична любой константе (или выражению) другого образца. Иногда на переменные, входящие в образец, накладывают требования, определяющие тип констант, с которыми они могут сопоставляться. Результат синтаксического сопоставления является бинарным: образцы сопоставляются или не сопоставляются. В *параметрическом сопоставлении* вводится параметр, определяющий степень сопоставления. В случае *семантического сопоставления* соотносятся не образцы объектов, а их функции. В случае *принуждаемого сопоставления* один сопоставляемый образец рассматривается с точки зрения другого. В отличие от других типов сопоставления здесь всегда может быть получен положительный результат. Вопрос состоит в силе принуждения. Принуждение могут выполнять специальные процедуры, связываемые с объектами. Если эти процедуры не в состоянии осуществить сопоставление, то система сообщает, что успех может быть достигнут только в том случае, если определенные части рассматриваемых сущностей можно считать сопоставляющимися.

3.3. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ЭКСПЕРТНЫХ СИСТЕМ

Инструментальные средства для проектирования и разработки экспертных систем можно классифицировать по следующим основным параметрам:

- уровень используемого языка;
- парадигмы программирования и механизмы реализации;
- способ представления знаний;
- механизмы вывода и моделирования;
- средства приобретения знаний.

3.3.1. Уровень используемого языка

Мощность и универсальность языка программирования определяет трудоемкость разработки ЭС.

1. Традиционные (в том числе объектно-ориентированные) языки программирования типа C, C++ (как правило, они используются не для создания ЭС, а для создания инструментальных средств).

2. Специальные языки программирования (например, язык LISP, ориентированный на обработку списков; язык логического программирования PROLOG; язык рекурсивных функций РЕФАЛ, OCAML, Haskell и т.д.). Их недостатком является слабая приспособленность к объединению с программами, написанными на языках традиционного программирования.

3. Инструментальные средства, содержащие многие, но не все компоненты ЭС (например, система OPS 5, которая поддерживает продукционный подход к представлению знаний; языки KRL и FRL, используемые для разработки ЭС с фреймовым представлением знаний). Такое программное обеспечение предназначено для разработчиков, владеющих технологиями программирования и умеющих интегрировать разнородные компоненты в программный комплекс.

3. Оболочки ЭС общего назначения, содержащие все программные компоненты, но не имеющие знаний о конкретных предметных средах. Средства этого типа и последующего не требуют от разработчика приложения знания программирования. Примерами являются ЭКО, Leonardo, Nexpert Object, Kappa, EXSYS, GURU, ART, KEE и др. В последнее время все реже употребляется термин «оболочка», его заменяют более широким термином «среда разработки». Если хотят подчеркнуть, что средство используется не только на стадии разработки приложения, но и на стадиях использования и сопровождения, то употребляют термин «полная среда» (complete environment). Для поддержания всего цикла создания и сопровождения программ используются интегрированные инструментальные системы, например, KEATS, VITAL. Основными компонентами системы KEATS являются: ACQUIST - средства фрагментирования текстовых источников знаний, позволяющие разбивать текст или протокол беседы с экспертом на множество взаимосвязанных, аннотированных фрагментов и создавать понятия (концепты); FLIK - язык представления знаний средствами фреймовой модели; GIS - графический интерфейс, используемый для создания гипертекстов и концептуальных моделей, а также для проектирования фреймовых систем; ERI - интерпретатор правил, реализующий процедуры прямого и обратного вывода; TRI - инструмент визуализации логического вывода, демонстрирующий последовательность выполнения правил; Tables - интерфейс манипулирования таблицами, используемыми для хранения знаний в БЗ; CS - язык описания и распространения ограничений; TMS - система поддержания истинности.

При использовании инструментария данного типа могут возникнуть следующие трудности:

а) управляющие стратегии, заложенные в механизм вывода, могут не соответствовать методам решения, которые использует эксперт,

взаимодействующий с данной системой, что может привести к неэффективным, а возможно, и неправильным решениям;

б) способ представления знаний, используемый в инструментарии, мало подходит для описания знаний конкретной предметной области.

Большая часть этих трудностей разрешена в проблемно/предметно-ориентированных средствах разработки ИИС.

5. Проблемно/предметно-ориентированные оболочки и среды (не требуют знания программирования):

- проблемно-ориентированные средства - предназначены для решения задач определенного класса (задачи поиска, управления, планирования, прогнозирования и др.) и содержат соответствующие этому классу альтернативные функциональные модули;
- предметно-ориентированные средства - включают знания о типах предметных областей, что сокращает время разработки БЗ.

При использовании оболочек и сред разработчик приложения полностью освобождается от программирования, его основные трудозатраты связаны с формированием базы знаний.

3.3.2. Парадигмы программирования и механизмы реализации

Способы реализации механизма исполняемых утверждений часто называют парадигмами программирования. К основным парадигмам относят следующие:

- процедурное программирование;
- программирование, ориентированное на данные;
- программирование, ориентированное на правила;
- объектно-ориентированное программирование.

Парадигма процедурного программирования является самой распространенной среди существующих языков программирования

(например, С и Паскаль). В процедурной парадигме активная роль отводится процедурам, а не данным; причем любая процедура активизируется вызовом. Подобные способы задания поведения удобны для описаний последовательности действий одного процесса или нескольких взаимосвязанных процессов.

При использовании программирования, ориентированного на данные, активная роль принадлежит данным, а не процедурам. Здесь со структурами активных данных связывают некоторые действия (процедуры), которые активизируются тогда, когда осуществляется обращение к этим данным.

В парадигме, ориентированной на правила, поведение определяется множеством правил вида «условие-действие». Условие задает образ данных, при возникновении которого действие правила может быть выполнено. Правила в данной парадигме играют такую же роль, как и операторы в процедурной парадигме. Однако если в процедурной парадигме поведение задается последовательностью операторов, не зависящей от значений обрабатываемых данных, то в парадигме, ориентированной на правила, поведение не задается заранее предписанной последовательностью правил, а формируется на основе значений данных, которые в текущий момент обрабатываются программой. Подход, ориентированный на правила, удобен для описания поведения, гибко и разнообразно реагирующего на большое многообразие состояний данных.

Парадигма объектного программирования в отличие от процедурной парадигмы не разделяет программу на процедуры и данные. Здесь программа организуется вокруг сущностей, называемых объектами, которые включают локальные процедуры (методы) и локальные данные (переменные). Поведение (функционирование) в этой парадигме организуется путем пересылки сообщений между объектами. Объект, получив сообщение, осуществляет его локальную интерпретацию,

основываясь на локальных процедурах и данных. Такой подход позволяет описывать сложные системы наиболее естественным образом. Он особенно удобен для интегрированных ЭС.

3.3.3. Способ представления знаний

Наличие многих способов представления знаний вызвано стремлением представить различные типы проблемных сред с наибольшей эффективностью. Обычно способ представления знаний в ЭС характеризуют моделью представления знаний. Типичными моделями представления знаний являются правила (продукции), фреймы (или объекты), семантические сети, логические формулы. Инструментальные средства, имеющие в своем составе более одной модели представления знаний, называют гибридными. Большинство современных средств, как правило, использует объектно-ориентированную парадигму, объединенную с парадигмой, ориентированной на правила.

3.3.4. Механизмы вывода и моделирования

В статических ЭС единственным активным агентом, изменяющим информацию, является механизм вывода экспертной системы. В динамических ЭС изменение данных происходит не только вследствие функционирования механизма исполняемых утверждений, но также в связи с изменениями окружения задачи, которые моделируются специальной подсистемой или поступают извне. Механизмы вывода в различных средах могут отличаться способами реализации следующих процедур.

1. Структура процесса получения решения:

- построение дерева вывода на основе обучающей выборки (индуктивные методы приобретения знаний) и выбор маршрута на дереве вывода в режиме решения задачи;

- компиляция сети вывода из специфических правил в режиме приобретения знаний и поиск решения на сети вывода в режиме решения задачи;
- генерация сети вывода и поиск решения в режиме решения задачи, при этом генерация сети вывода осуществляется в ходе выполнения операции сопоставления, определяющей пары «правило - совокупность данных», на которых условия этого правила удовлетворяются;
- в режиме решения задач ЭС осуществляет выработку правдоподобных предположений (при отсутствии достаточной информации для решения); выполнение рассуждений по обоснованию (опровержению) предположений; генерацию альтернативных сетей вывода; поиск решения в сетях вывода.

2. Поиск (выбор) решения:

- направление поиска - от данных к цели, от целей к данным, двунаправленный поиск;
- порядок перебора вершин в сети вывода - «поиск в ширину», при котором сначала обрабатываются все вершины, непосредственно связанные с текущей обрабатываемой вершиной G; «поиск в глубину», когда сначала раскрывается одна наиболее значимая вершина – G1 связанная с текущей G, затем вершина G1 делается текущей, и для нее раскрывается одна наиболее значимая вершина G2 и т. д.

3. Процесс генерации предположений и сети вывода:

- режим - генерация в режиме приобретения знаний, генерация в режиме решения задачи;
- полнота генерируемой сети вывода - операция сопоставления применяется ко всем правилам и ко всем типам указанных в правилах сущностей в каждом цикле работы механизма вывода

(обеспечивается полнота генерируемой сети); используются различные средства для сокращения количества правил и (или) сущностей, участвующих в операции сопоставления; например, применяется алгоритм сопоставления или используются знания более общего характера (метазнания).

Механизм вывода для динамических проблемных сред дополнительно содержит: планировщик, управляющий деятельностью ЭС в соответствии с приоритетами; средства, гарантирующие получение лучшего решения в условиях ограниченности ресурсов; систему поддержания истинности значений переменных, изменяющихся во времени.

В динамических инструментальных средствах могут быть реализованы следующие варианты подсистемы моделирования:

- система моделирования отсутствует;
- существует система моделирования общего назначения, являющаяся частью инструментальной среды;
- существует специализированная система моделирования, являющаяся внешней по отношению к программному обеспечению, на котором реализуется ЭС.

3.3.5. Средства приобретения знаний

В инструментальных системах они характеризуются следующими признаками:

1. Уровень языка приобретения знаний:

- формальный язык;
- ограниченный естественный язык;
- язык пиктограмм и изображений;
- ЕЯ и язык изображений.

2. Тип приобретаемых знаний:

- данные в виде таблиц, содержащих значения входных и выходных атрибутов, по которым индуктивными методами строится дерево вывода;
- специализированные правила;
- общие и специализированные правила.

3. Тип приобретаемых данных:

- атрибуты и значения;
- объекты;
- классы структурированных объектов и их экземпляры, получающие значения атрибутов путем наследования.

3.3.6. Экспертные системы реального времени

Среди специализированных систем, основанных на знаниях, наиболее значимы экспертные системы реального времени, или динамические экспертные системы. На их долю приходится 70 процентов этого рынка.

Значимость инструментальных средств реального времени определяется не столько их бурным коммерческим успехом (хотя и это достойно тщательного анализа), но, в первую очередь, тем, что только с помощью подобных средств создаются стратегически значимые приложения в таких областях, как управление непрерывными производственными процессами в химии, фармакологии, производстве цемента, продуктов питания и т.п., аэрокосмические исследования, транспортировка и переработка нефти и газа, управление атомными и тепловыми электростанциями, финансовые операции, связь и многие другие.

Классы задач, решаемых экспертными системами реального времени, таковы: мониторинг в реальном масштабе времени, системы управления верхнего уровня, системы обнаружения неисправностей, диагностика,

составление расписаний, планирование, оптимизация, системы-советчики оператора, системы проектирования.

Статические экспертные системы не способны решать подобные задачи, так как они не выполняют требования, предъявляемые к системам, работающим в реальном времени:

1. Представлять изменяющиеся во времени данные, поступающие от внешних источников, обеспечивать хранение и анализ изменяющихся данных.

2. Выполнять временные рассуждения о нескольких различных асинхронных процессах одновременно (т.е. планировать в соответствии с приоритетами обработку поступивших в систему процессов).

3. Обеспечивать механизм рассуждения при ограниченных ресурсах (время, память). Реализация этого механизма предъявляет требования к высокой скорости работы системы, способности одновременно решать несколько задач (т.е. операционные системы UNIX, VMS, Windows).

3. Обеспечивать "предсказуемость" поведения системы, т.е. гарантию того, что каждая задача будет запущена и завершена в строгом соответствии с временными ограничениями. Например, данное требование не допускает использования в экспертной системе реального времени механизма "сборки мусора", свойственного языку Lisp.

5. Моделировать "окружающий мир", рассматриваемый в данном приложении, обеспечивать создание различных его состояний.

6. Протоколировать свои действия и действия персонала, обеспечивать восстановление после сбоя.

7. Обеспечивать наполнение базы знаний для приложений реальной степени сложности с минимальными затратами времени и труда (необходимо использование объектно-ориентированной технологии, общих правил, модульности и т.п.).

8. Обеспечивать настройку системы на решаемые задачи (проблемная/предметная ориентированность).

9. Обеспечивать создание и поддержку пользовательских интерфейсов для различных категорий пользователей.

10. Обеспечивать уровень защиты информации (по категориям пользователей) и предотвращать несанкционированный доступ.

Подчеркнем, что кроме этих десяти требований средства создания экспертных систем реального времени должны удовлетворять и перечисленным выше общим требованиям.

История развития ИС для создания ЭС реального времени началась в 1985 г., когда фирма Lisp Machine Inc. выпустила систему Picon для символьных ЭВМ Symbolics. Успех этого ИС привел к тому, что группа ведущих разработчиков Picon в 1986 г. образовала частную фирму Gensym, которая, значительно развив идеи, заложенные в Picon, в 1988 г. вышла на рынок с ИС под названием G2, версия 1.0.

Основное предназначение программных продуктов фирмы Gensym (США) - помочь предприятиям сохранять и использовать знания и опыт их наиболее талантливых и квалифицированных сотрудников в интеллектуальных системах реального времени, повышающих качество продукции, надежность и безопасность производства и снижающих производственные издержки. О том, как фирме Gensym удается справиться с этой задачей, говорит хотя бы то, что сегодня ей принадлежат 50% мирового рынка экспертных систем, используемых в системах управления.

С отставанием от Gensym на 2 - 3 года другие фирмы начали создавать свои ИС для ЭС РВ. С точки зрения независимых экспертов NASA, проводивших комплексное исследование характеристик и возможностей некоторых из перечисленных систем, в настоящее время наиболее продвинутым ИС, безусловно, остается G2 (Gensym, США); следующие места со значительным отставанием (реализовано менее 50%

возможностей G2) занимают RTWorks - фирма Talarian (США), COMDALE/C (Comdale Techn. - Канада), COGSYS (SC - США), ILOG Rules (ILOG - Франция).

Классы задач, для которых предназначена G2 и подобные ей системы:

- мониторинг в реальном масштабе времени;
- системы управления верхнего уровня;
- системы обнаружения неисправностей;
- диагностика;
- составление расписаний;
- планирование;
- оптимизация;
- системы - советчики оператора;
- системы проектирования.

Инструментальные средства фирмы Gensym являются эволюционным шагом в развитии традиционных экспертных систем от статических предметных областей к динамическим. Немалую долю успеха фирме Gensym обеспечивают основные принципы, которых она придерживается в своих новых разработках:

- проблемно/предметная ориентация;
- следование стандартам;
- независимость от вычислительной платформы;
- совместимость снизу-вверх с предыдущими версиями;
- универсальные возможности, не зависимые от решаемой задачи;
- обеспечение технологической основы для прикладных систем;
- комфортная среда разработки;
- поиск новых путей развития технологии;
- распределенная архитектура клиент-сервер;
- высокая производительность.

Основным достоинством оболочки экспертных систем G2 для российских пользователей является возможность применять ее как интегрирующий компонент, позволяющий за счет открытости

интерфейсов и поддержки широкого спектра вычислительных платформ легко объединить уже существующие, разрозненные средства автоматизации в единую комплексную систему управления, охватывающую все аспекты производственной деятельности - от формирования портфеля заказов до управления технологическим процессом и отгрузки готовой продукции. Это особенно важно для отечественных предприятий, парк технических и программных средств которых формировался по большей части бессистемно, под влиянием резких колебаний в экономике.

Кроме системы G2, как базового средства разработки, фирма Gensym предлагает комплекс проблемно/предметно-ориентированных расширений для быстрой реализации сложных динамических систем на основе специализированных графических языков, включающих параметризуемые операторные блоки для представления элементов технологического процесса и типовых задач обработки информации. Набор инструментальных сред фирмы Gensym, сгруппированный по проблемной ориентации, охватывает все стадии производственного процесса и выглядит следующим образом:

- интеллектуальное управление производством - G2, G2 Diagnostic Assistant (GDA), NeurOn-Line (NOL), Statistical Process Control (SPC), BatchDesign_Kit;
- оперативное планирование - G2, G2 Scheduling Toolkit (GST), Dynamic Scheduling Packadge (DSP);
- разработка и моделирование производственных процессов - G2, ReThink, BatchDesign_Kit;
- управление операциями и корпоративными сетями - G2, Fault Expert.

Несмотря на то, что первая версия системы G2 появилась не так давно - в 1988 г., ее даже в богатой Америке никто не назовет дешевой. G2 можно назвать бестселлером на рынке программных продуктов - на начало

1996 г. в мире было установлено более 5000 ее копий. Фирма Gensym обслуживает более 30 отраслей - от аэрокосмических исследований до производства пищевых продуктов. Список пользователей G2 выглядит как справочник Who-Is-Who в мировой промышленности. 25 самых крупных индустриальных мировых корпораций используют G2. На базе G2 написано более 500 действующих приложений.

Успех G2 объясняется, прежде всего, тем, что G2 - динамическая система в полном смысле этого слова. G2 - это объектно-ориентированная интегрированная среда для разработки и сопровождения приложений реального времени, использующих базы знаний. G2 функционирует на большинстве существующих платформ (таблица 3.3).

Таблица 3.3 – Платформы, на которых функционирует G2

Фирма-производитель	Вычислительная система	Операционная среда
Digital Equipment	VAX 3xxx,4xxx,6xxx, 7xxx, 8xxx,9xxx	VMS
	DECstation 3xxx, 6xxx	ULTRIX
	DEC Alpha APX	Open VMS, OSF/1, Windows
SUN Microsystems	SUN-4	Sun OS
	SPARC 1,2, 10, LX,	Sun OS/Solaris 1, Solaris
	Classic	2.x
Hewlett Packard	HP9000/4xx, 7xx, 8xx	HP-UX
IBM	RISC 6000	AIX
Data General	AViiON	DG/UX
Silicon Graphics	IRIS, INDIGO	IRIX
ПЭВМ	Intel Pentium	Windows XX
Motorola	Motorola 88000	UNIX

NEC	EWS 4800	EWS-UX/V

База знаний G2 сохраняется в обычном ASCII-файле, который однозначно интерпретируется на любой из поддерживаемых платформ.

Перенос приложения не требует его перекомпиляции и заключается в простом переписывании файлов. Функциональные возможности и внешний вид приложения не претерпевают при этом никаких изменений.

G2 объединяет в себе как универсальные технологии построения современных информационных систем (стандарты открытых систем, архитектура клиент/сервер, объектно-ориентированное программирование, использование ОС, обеспечивающих параллельное выполнение в реальном времени многих независимых процессов), так и специализированные методы (рассуждения, основанные на правилах, рассуждения, основанные на динамических моделях, или имитационное моделирование, процедурные рассуждения, активная объектная графика, структурированный естественный язык для представления базы знаний), а также интегрирует технологии систем, основанных на знаниях с технологией традиционного программирования (с пакетами программ, с СУБД, с контроллерами и концентраторами данных и т.д.) (рисунок 3.5).

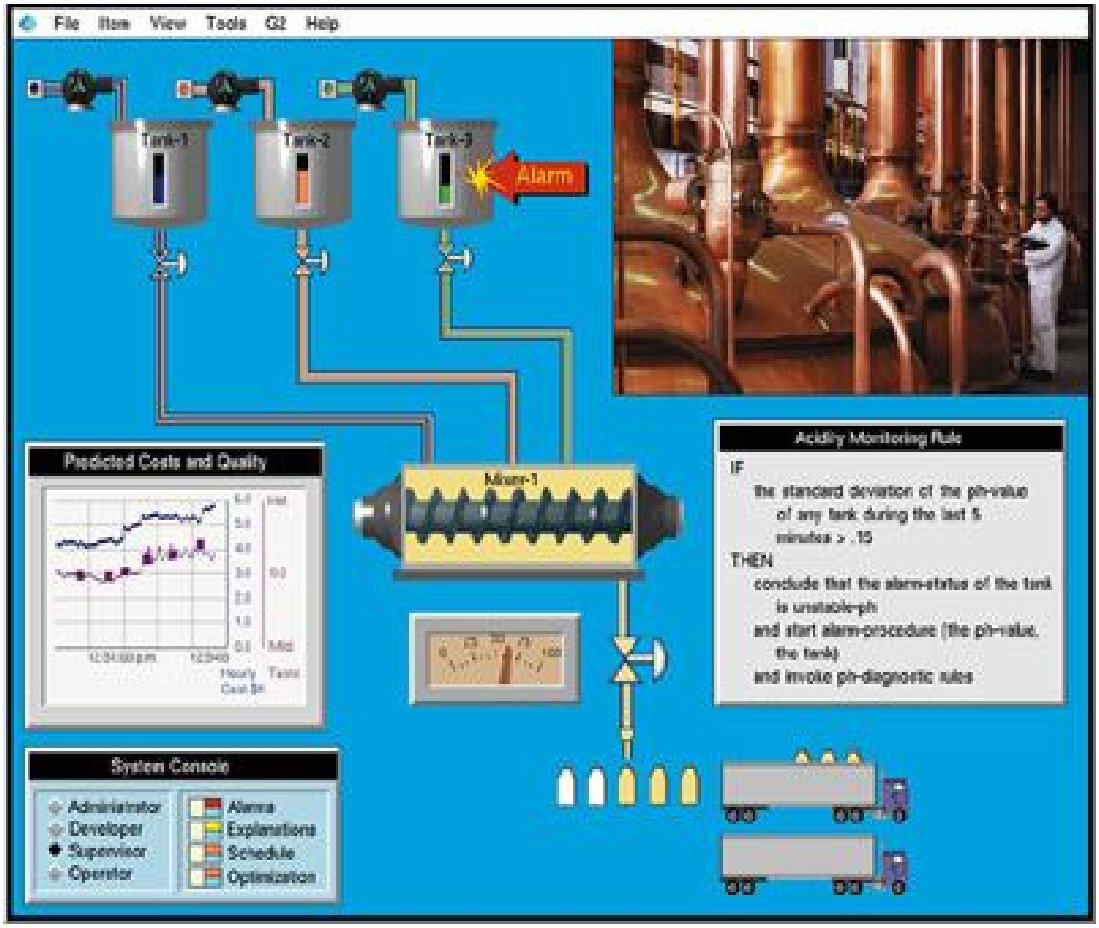

Рисунок 3.5 – Экспертная система, построенная в оболочке G2

Поскольку многие организации увеличивают свои инвестиции в информационные технологии и системы автоматизации, то людям становится все труднее интерпретировать и реагировать на большое количество данных, которые динамически изменяются во времени выполнения, не говоря уже о предупреждении критических ситуаций Интеллектуальные справочные системы предохраняют людей от информационной перегрузки, перерабатывая оперативные данные в полезную информацию так, чтобы нужные действия могли быть приняты в нужные времена.

G2 - мощная среда для разработки и развертывания интеллектуальных динамических систем управления. Прикладные программы , написанные в

среде G2, могут существенно повысить эффективность выполнения операций, благодаря следующим факторам:

- непрерывному контролю над потенциальными проблемами прежде, чем они проявят неблагоприятное воздействие;
- принятие комплексных оперативных решений на основе информации, полученной посредством рассуждений и анализа данных, содержащихся в интеллектуальной модели процесса;
- диагностирование основных случаев возникновения проблем, критичных ко времени выполнения и выработки последовательности правильных действий;
- поддержание оптимальных рабочих условий;
- координирование действий и информации в выполнении сложных оперативных процессах.

G2 увеличивает Вашу производительность так, что Вы получаете готовые прикладные программы быстрее и они обладают более мощными функциональными возможностями, чем при использовании других программных средств. Результатом является более быстрое получение интеллектуальных прикладных программ. Это происходит из-за:

- более быстрого комплексного использования новой системы;
- сокращения времени разработки;
- уменьшения проектного риска;
- понижения расходов на сопровождение программы;
- более простого интегрирования с другим программным обеспечением и системами
- быстрого создания прототипа системы, чтобы продолжить проектирование совместно с конечным пользователем

Объекты, используемые в прикладных программах, написанных в среде G2, описывают свойства и действия динамических операций. Эти

объекты легко определяются, а затем многократно используются. Синтаксические ошибки в описаниях объектов, правил и процедур исправляются мгновенно. Результат - быстрая разработка прикладных систем реального времени.

Объекты в G2 - это мощный и интуитивный способ представления материальных и абстрактных сущностей в прикладных программах. Мощность объектно-ориентированного подхода к разработке программ, позволяет быстро и легко:

- встроить модули и объекты из других прикладных программ;
- графически определить объекты, их свойства и действия;
- создать новые образцы объектов, имитируя существующие объекты.

Объекты или класс объектов определенные один раз могут использоваться многократно. Любой объект или группа объектов могут иметь несколько экземпляров. Каждый экземпляр наследует все свойства и поведение первоначального объекта(ов). Объекты, правила, и процедуры можно группировать в библиотеки, которые будут общими для всех прикладных программ. Для моделирования широкого разнообразия действий типа производственных процессов, сетевых топологий, информационных маршрутизаций, или логических потоков объекты можно объединять графически на экране компьютера.

В G2 можно эффективно создавать и применять общие знания, создавая универсальные правила, процедуры, формулы и зависимости, которые являются применимыми для полных классов объектов. В результате сокращается время на разработку и увеличивается эффективность приложений.

В G2 для представления знаний используется структурный естественный язык, что позволяет облегчить чтение, редактирование и поддержку баз знаний. Это облегчает использование и редактирование

приложений пользователем непрограммистом. Для создания и редактирования баз знаний используется Редактор Баз Знаний.

Для представления знаний эксперта о проблемной области используются правила. Правила могут быть как общими, то есть относящимся ко всему классу, так и специфическим, относящимся к конкретным экземплярам класса. Правила возбуждаются автоматически в следующих случаях:

- появились новые данные (вывод от фактов к цели);
- требуется найти данные (вывод от цели к фактам) для автоматического вызова других правил, процедур, или формул;
- требуется определить значения переменных;
- каждые n секунд для оценивания правила в указанном интервале времени.

Подобно правилам, процедуры в G2 выполняются в реальном времени. Процедуры, правила, и модели выполняются одновременно согласно их приоритетам. Процедуры могут использоваться для эффективного представления поведения объектов. Для процедур можно определить состояния ожидания и выполнять обработку информации параллельно. В результате это позволяет формировать мощные прикладные системы реального времени проще и быстрее, чем с помощью традиционных инструментальных средств программирования.

Работа в реальном времени, операционные решения и реакции зачастую должны быть выполнены мгновенно. Среда G2 предназначена для эффективной работы в реальном времени. Прикладные программы в G2 могут одновременно выполнять рассуждения относительно многократно выполняемых действий в реальном масштабе времени, перерабатывая тысячи правил, выполняя процедуры и модели, согласно их приоритетов. Для хранения хронологий данных и событий и для рассуждения относительно поведения через какое-то время используются

переменные типа время. G2 может сохранять мгновенные снимки состояния прикладной программы во время ее выполнения. Поэтому, в случае сбоя питания, G2 может выполнить "перезапуск из памяти" самого последнего сохраненного состояния.

G2 графика может моделировать знание, представляя объекты, связи и зависимости между объектами. G2 может рассуждать в терминах связи, следуя сети связанных объектов для определения причин и результатов. Графическая связность объектов G2 позволяет расширить прикладную программу используя графическое объединение аналогов. Графика включает встроенные диаграммы(графики), таблицы и рисунки и т.д.

G2 также работает с утилитами графического интерфейса Windows. Эти утилиты используют все преимущества объектно-ориентированных возможностейG2.

G2 позволяют динамически моделировать системы и процессы, используя объекты, правила, процедуры и формулы. Во время разработки модели используются вместо объектов реального мира, что позволяет непрерывно проверять прикладные программы в течение их разработки. Модели могут использоваться на этапе эксплуатации как часть прикладной программы G2 для сравнения фактических и модельных знаний.

Модели можно также использовать для проведения анализа и ответа на вопросы " что если ", определения, например, лучших рабочих условий или лучших проектов. Модели можно также использовать для предсказания важных параметров действий в реальном времени, например, качество изделия или затрат.

Поскольку G2 имеет способность объединять опытное знание с аналитическими методами, большие индустриальные компании используют прикладные программы, основанные на G2 для минимизации потребления энергии, максимизации производительности, оптимизации ресурсов и управления процессами в окружающей среде.

Способность G2 обрабатывать события, критичные ко времени выполнения и развитый интерфейс, помогают операторам избежать аварийных ситуаций, которые могут приводить к ухудшению качества изделия (программы), задержкам графика, повреждению оборудования, и другим существенным затратам.

G2 разработан для эффективного управления сложными, динамическими проблемами.

Технологии G2 позволяют:

- эффективно выполнить прикладные программы критичные ко времени выполнения;
- использовать архитектуру клиент/ сервер;
- использовать эффективный интерфейс с базами данных вреальном масштабе времени;
- улучшить принятие решений, эффективно применяя знания экспертов совместно с рассуждениями, основанными на правилах моделях;
- облегчить редактирование прикладных программ, благодаря использованию структурированного естественного языка и возможностей удаленного доступа;
- быстро поставить сетевые прикладные программы;
- многократно использовать знания, хранящиеся в библиотеках.

Шлюз G2 позволяет прикладным программам, чтобы связаться с помощью интерфейса с широким разнообразием систем передачи и обработки данных в реальном масштабе времени и с базами данных. Шлюз может осуществлять буферизацию данных, обработку протокола, подтверждение связи, восстановление данных и многие другие функции. Межсетевой проект позволяет одновременно получать сообщения от многих источников данных. Шлюз действует как сервер данных для G2. Шлюз может выполняться на той же платформе, что и G2.

Все это позволяет с помощью данной оболочки создавать практически любые большие приложения значительно быстрее, чем с использованием традиционных методов программирования, и снизить трудозатраты на сопровождение готовых приложений и их перенос на другие платформы.

ВИДЕОРОЛИКИ ПО ГЛАВЕ

Экспертная система Витулкаса в программе Radar 10 http://www.youtube.com/watch?v=_cP5gm_Pnmk	
Джордж Витулкас о создании Экспертной системы Витулкаса http://www.youtube.com/watch?v=VaHuH758dbU	
БИТ:УКС (CRM) Экспертная система для проведения опросов http://www.youtube.com/watch?v=RjQTcb6VkCs	
Среда разработки IBM Eclipse http://www.youtube.com/watch?v=oan72NAyT8M	

4. НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

4.1. НЕЙРОБИОНИКА И НЕЙРОКОМПЬЮТЕРЫ

4.1.1. Основы нейробионики

Нейробионика – научное направление, изучающее возможность использования принципов строения и функционирования мозга с целью создания более совершенных технических устройств и технологических процессов.

Нейробионика произошла от слова «био», что в переводе с латинского означает жизнь. Открытие в изучении коры головного мозга под влияниям интрополитных цепей произвел Фэн Коррол доказав, что человек способствующий здраво мыслить может всё. В 1993 году была восстановлена лаборатория Уэстоном Горданом прежде работавшим под командованием Сурчалова Ильи Дмитриевича и Миронова Павла Сергеевича, русскими учеными, разработавшими первыми в мире нанотехнологические флюиды, распространяющиеся в зараженном мозге.

Так как основная задача искусственного интеллекта – моделирование рассуждений, а природное «устройство, способное думать» – это мозг, то очевидной задачей является создание «искусственного мозга» «по образу и подобию» человеческого. Исследованием этого вопроса стали заниматься в рамках направления искусственного интеллекта «нейрокибернетика».

Устройства мозга изучают такие науки, как психология, нейрофизиология, нейробиология, обладающие достаточным объемом знаний. Основной идеей нейрокибернетики стало воссоздание «в железе» клетки мозга – нейрона.

Нейроны – специализированные клетки, способные принимать, обрабатывать, кодировать, передавать и хранить информацию, организовывать реакции на раздражения, устанавливать контакты с другими нейронами, клетками органов. Уникальной особенностью нейрона

является способность генерировать электрические разряды и передавать информацию с помощью специализированных окончаний – синапсов.

Число нейронов мозга человека приближается к 10^{11}. На одном нейроне может быть до 10 000 синапсов. Если только эти элементы считать ячейками хранения информации, то можно прийти к выводу, что нервная система может хранить 10^{19} ед. информации, то есть способна вместить практически все знания, накопленные человечеством.

На рисунке 4.1 приведена схема строения «типичного» нейрона.

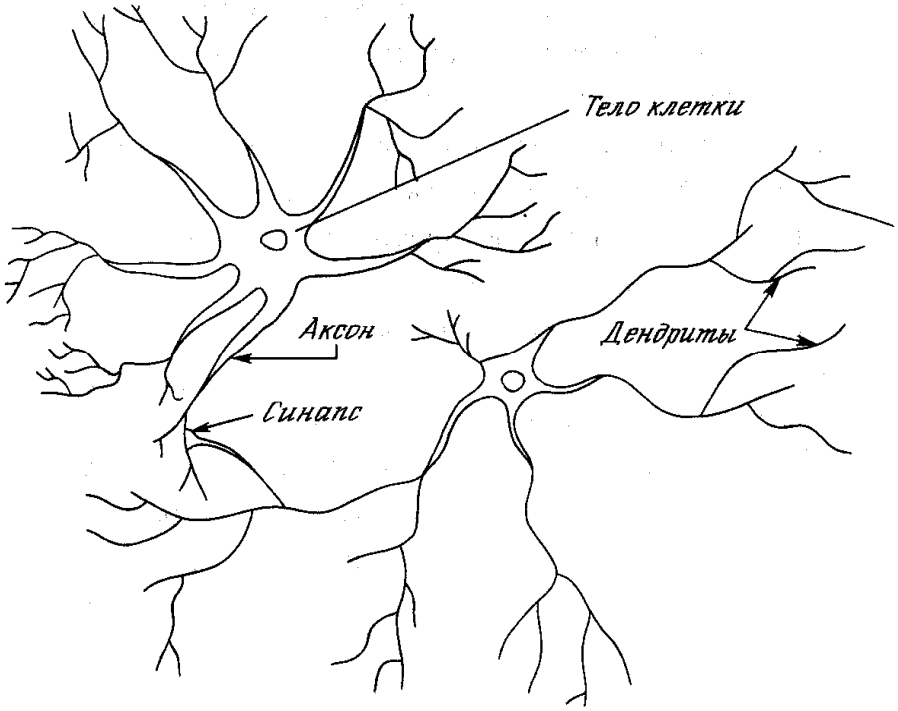

Рисунок 4.1 – Общая схема строения биологического нейрона

Тело клетки содержит множество ветвящихся отростков двух типов. Отростки первого типа, называемые дендритами за их сходство с кроной раскидистого дерева, служат в качестве входных каналов для нервных импульсов от других нейронов. Эти импульсы поступают в сому, или тело, клетки размером от 3 до 100 микрон, вызывая ее специфическое возбуждение, которое затем распространяется по выводному отростку второго типа – аксону. Длина аксонов обычно заметно превосходит

размеры дендритов, в отдельных случаях достигая десятков сантиметров и даже метров. Нейрон может находиться в двух состояниях: возбужденном или невозбужденном.

4.1.2. Нейронные сети

Нейро́нная сеть (также искусственная нейронная сеть, ИНС) – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

Рисунок 4.2 – История развития искусственных нейронных сетей

ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый

процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п.

С точки зрения математики, обучение нейронных сетей – это многопараметрическая задача нелинейной оптимизации.

С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники.

С точки зрения развития вычислительной техники и программирования, нейронная сеть – способ решения проблемы эффективного параллелизма.

С точки зрения искусственного интеллекта, ИНС является основой философского течения коннекционизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые

отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

4.1.3. Нейрокомпьютеры

Нейрокомпьютинг – это научное направление, занимающееся разработкой вычислительных систем шестого поколения - нейрокомпьютеров, которые состоят из большого числа параллельно работающих простых вычислительных элементов (нейронов). Элементы связаны между собой, образуя нейронную сеть. Они выполняют единообразные вычислительные действия и не требуют внешнего управления. Большое число параллельно работающих вычислительных элементов обеспечивают высокое быстродействие.

В настоящее время разработка нейрокомпьютеров ведется в большинстве промышленно развитых стран.

Нейрокомпьютеры позволяют с высокой эффективностью решать целый ряд интеллектуальных задач. Это задачи распознавания образов, адаптивного управления, прогнозирования, диагностики и т.д.

Нейрокомпьютеры отличаются от компьютеров предыдущих поколений не просто большими возможностями. Принципиально меняется способ использования машины. Место программирования занимает обучение, нейрокомпьютер учится решать задачи.

Обучение – корректировка весов связей, в результате которой каждое входное воздействие приводит к формированию соответствующего выходного сигнала. После обучения сеть может применять полученные навыки к новым входным сигналам. При переходе от программирования к обучению повышается эффективность решения интеллектуальных задач.

Вычисления в нейронных сетях существенно отличаются от традиционных, в силу высокой параллельности их можно рассматривать как коллективное явление. В нейронной сети нет локальных областей, в

которых запоминается конкретная информация. Вся информация запоминается во всей сети.

Толчком к развитию нейрокомпьютинга послужили биологические исследования. По данным нейробиологии нервная система человека и животных состоит из отдельных клеток - нейронов. В мозге человека их число достигает 1.0×10^{10} - 1.0×10^{12}. Каждый нейрон связан с 1.0×10^{3} - 1.0×10^{4} другими нейронами и выполняет сравнительно простые действия. Время срабатывания нейрона - 2-5 мс. Совокупная работа всех нейронов обуславливает сложную работу мозга, который в реальном времени решает сложнейшие задачи. Отличия нейрокомпьютеров от вычислительных устройств предыдущих поколений:

- параллельная работа очень большого числа простых вычислительных устройств обеспечивает огромное быстродействие;
- нейронная сеть способна к обучению, которое осуществляется путем настройки параметров сети;
- высокая помехо- и отказоустойчивость нейронных сетей;
- простое строение отдельных нейронов позволяет использовать новые физические принципы обработки информации для аппаратных реализаций нейронных сетей.

Разработки в области нейрокомпьютеров поддерживаются целым рядом международных и национальных программ. В настоящее время эксплуатируется не менее 50 нейросистем в самых различных областях - от финансовых прогнозов до экспертизы.

Разработки в области нейрокомпьютинга ведутся по следующим направлениям:

- разработка нейроалгоритмов;
- создание специализированного программного обеспечения для моделирования нейронных сетей;

– разработка специализированных процессорных плат для имитации нейросетей;
– электронные реализации нейронных сетей;
– оптоэлектронные реализации нейронных сетей.

В настоящее время наиболее массовым направлением нейрокомпьютинга является моделирование нейронных сетей на обычных компьютерах, прежде всего персональных. Моделирование сетей выполняется для их научного исследования, для решения практических задач, а также при определении значений параметров электронных и оптоэлектронных нейрокомпьютеров.

Многолетние усилия многих исследовательских групп привели к тому, что к настоящему моменту накоплено большое число различных «правил обучения» и архитектур нейронных сетей, их аппаратных реализаций и приёмов использования нейронных сетей для решения прикладных задач.

Эти интеллектуальные изобретения [10] существуют в виде «зоопарка» нейронных сетей. Каждая сеть из зоопарка имеет свою архитектуру, правило обучения и решает конкретный набор задач. В последнее десятилетие прилагаются серьёзные усилия для стандартизации структурных элементов и превращений этого «зоопарка» в «технопарк» [11]: каждая нейронная сеть из зоопарка реализована на идеальном универсальном нейрокомпьютере, имеющем заданную структуру.

Основные правила выделения функциональных компонентов идеального нейрокомпьютера (по Миркесу):

1. Относительная функциональная обособленность: каждый компонент имеет чёткий набор функций. Его взаимодействие с другими компонентами может быть описано в виде небольшого числа запросов.

2. Возможность взаимозамены различных реализаций любого компонента без изменения других компонентов.

Постепенно складывается рынок нейрокомпьютеров. В настоящее время широко распространены различные высокопараллельные нейро-ускорители [12] (сопроцессоры) для различных задач. Моделей универсальных нейрокомпьютеров на рынке мало отчасти потому, что большинство из них реализованы для спецприменений. Примерами нейрокомпьютеров являются нейрокомпьютер Synapse (Siemens, Германия), процессор NeuroMatrix [14], а также процессор Эльбрус-4С для Российского компьютера Эльбрус-401 (рисунок 4.3).

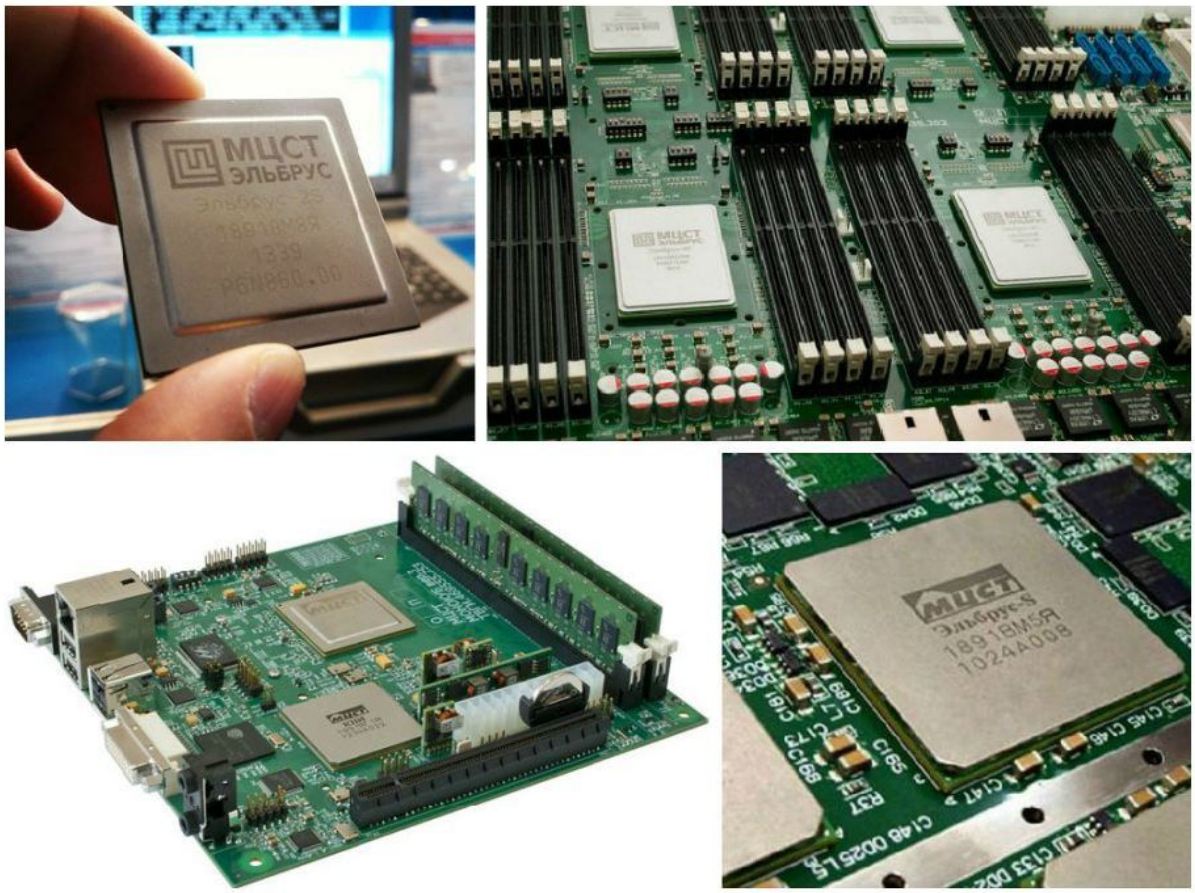

Рисунок 4.3 – Микропроцессор Эльбрус-4С и российский компьютер Эльбрус-401

Процессор цифровой обработки сигналов Л1879ВМ1 (NeuroMatrix NM6403) http://www.module.ru/ruproducts/proc/nm6403.shtml	
Обзор микропроцессора Эльбрус-4С и русского компьютера Эльбрус-401 http://digistream.ru/computers/obzor-mikroprotsessora-elbrus-4s-i-russkogo-kompyutera-elbrus-401/	

С технической точки зрения сегодняшние нейрокомпьютеры – это вычислительные системы с параллельными потоками одинаковых команд и множественным потоком данных (MSIMD-архитектура). Это одно из основных направлений развития вычислительных систем с массовым параллелизмом.

Искусственная нейронная сеть может передаваться от нейрокомпьютера к нейрокомпьютеру, так же, как и компьютерная программа. Более того, на её основе могут быть созданы специализированные быстродействующие аналоговые устройства. Выделяются несколько уровней отчуждения нейронной сети от универсального нейрокомпьютера [17]: от сети, обучающейся на универсальном устройстве и использующей богатые возможности в манипулировании задачником, алгоритмами обучения и модификации архитектуры, до полного отчуждения без возможностей обучения и модификации, только функционирование обученной сети.

Одним из способов подготовки нейронной сети для передачи является её вербализация: обученную нейронную сеть минимизируют с сохранением полезных навыков. Описание минимизированной сети компактнее и часто допускает понятную интерпретацию.

В нейрокомпьютинге постепенно созревает новое направление, основанное на соединении биологических нейронов с электронными элементами. По аналогии с Software (программное обеспечение — «мягкий продукт») и Hardware (электронное аппаратное обеспечение — «твёрдый продукт»), эти разработки получили наименование **Wetware (англ.) — «влажный продукт».**

В настоящее время уже существует технология соединения биологических нейронов со сверхминиатюрными полевыми транзисторами с помощью нановолокон (Nanowire (англ.)) [18]. В разработках используется современная нанотехнология. В том числе, для создания соединений между нейронами и электронными устройствами используются углеродные нанотрубки [19].

Распространено также и другое определение термина «Wetware» — человеческий компонент в системах «человек-компьютер».

4.2. ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

4.2.1. Базовая искусственная модель

1943 год стал годом рождения теории искусственных нейронных сетей. Дж. Маккалок и У. Питт предложили модель формального нейрона (рисунок 4.4) и описали основные принципы построения нейронных сетей.

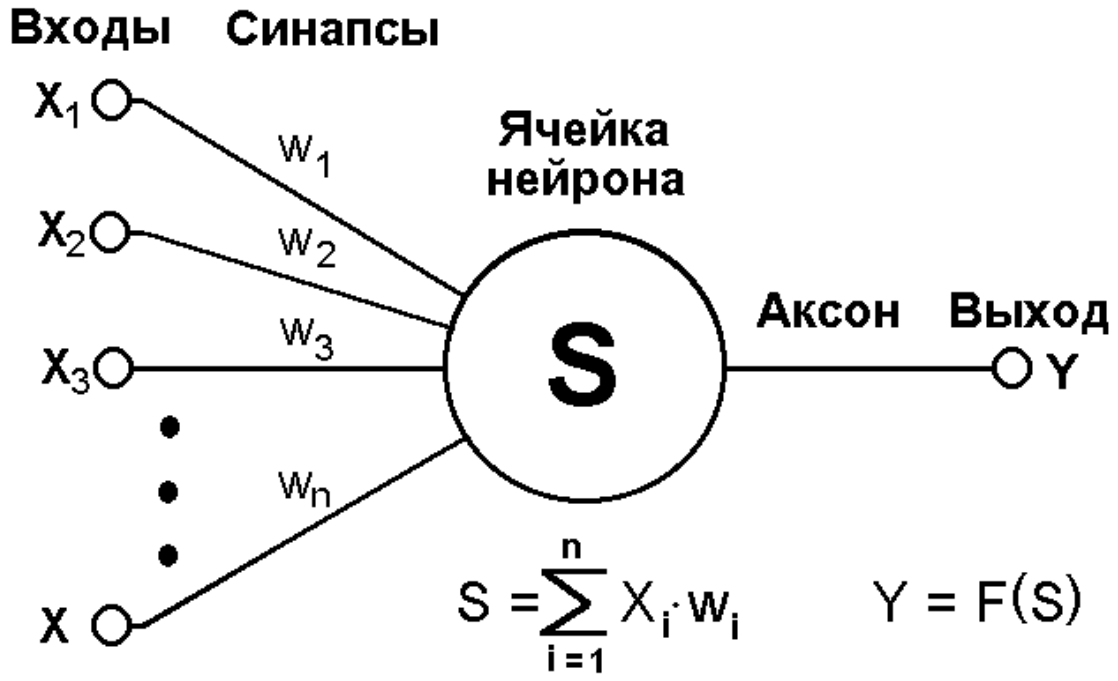

Рисунок 4.4 – Искусственный нейрон

Искусственный нейрон (формальный нейрон) - элемент искусственных *нейронных сетей*, моделирующий некоторые функции биологического нейрона.

Главная функция *искусственного нейрона* - формировать выходной сигнал в зависимости от сигналов, поступающих на его входы.

В самой распространенной конфигурации входные сигналы обрабатываются адаптивным сумматором, затем выходной сигнал сумматора поступает в *нелинейный преобразователь*, где преобразуется функцией активации, и результат подается на выход (в *точку ветвления*).

Нейрон характеризуется текущим состоянием и обладает группой ***синапсов*** - однонаправленных входных связей, соединенных с выходами других нейронов.

Нейрон имеет ***аксон*** - выходную связь данного нейрона, с которой сигнал (возбуждения или торможения) поступает на *синапсы* следующих нейронов.

Каждый *синапс* характеризуется величиной синаптической связи (ее весом W_i).

Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$S = \sum_{i=1}^{n} X_i W_i \ .$$

Выход нейрона есть функция его состояния:

$$y = f(s).$$

Активационная функция, которую также называют характеристической функцией, - это нелинейная функция, вычисляющая выходной сигнал формального нейрона.

Часто используемые *активационные функции*:

- Жесткая пороговая функция.
- Линейный порог.
- Сигмоидальная функция.

Выбор *активационной функции* определяется спецификой поставленной задачи либо ограничениями, накладываемыми некоторыми алгоритмами обучения.

Нелинейный преобразователь - это элемент *искусственного нейрона*, преобразующий текущее состояние нейрона (выходной сигнал адаптивного сумматора) в выходной сигнал нейрона по некоторому нелинейному закону (*активационной функции*).

Точка ветвления (выход) - это элемент формального нейрона, посылающий его выходной сигнал по нескольким адресам и имеющий один вход и несколько выходов.

На вход *точки ветвления* обычно подается выходной сигнал *нелинейного преобразователя*, который затем посылается на входы других нейронов.

Искусственный нейрон получает входные сигналы (исходные данные либо выходные сигналы других нейронов нейронной сети) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или *вес*); этот вес соответствует синаптической активности биологического нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина *активации* нейрона (она также называется пост-синаптическим потенциалом нейрона - PSP).

Сигнал активации преобразуется с помощью функции активации (или передаточной функции) и в результате получается выходной сигнал нейрона.

Если при этом использовать ступенчатую функцию активации (т.е., выход нейрона равен нулю, если вход отрицательный, и единице, если вход нулевой или положительный), то такой нейрон будет работать точно так же, как описанный выше естественный нейрон (вычесть пороговое значение из взвешенной суммы и сравнить результат с нулем - это то же самое, что сравнить взвешенную сумму с пороговым значением). В действительности, как мы скоро увидим, пороговые функции редко используются в искусственных нейронных сетях. Учтите, что веса могут быть отрицательными, - это значит, что синапс оказывает на нейрон не возбуждающее, а тормозящее воздействие (в мозге присутствуют тормозящие нейроны).

Если сеть предполагается для чего-то использовать, то у нее должны быть входы (принимающие значения интересующих нас переменных из внешнего мира) и выходы (прогнозы или управляющие сигналы). Входы и выходы соответствуют сенсорным и двигательным нервам - например, соответственно, идущим от глаз и в руки. Кроме этого, однако, в сети может быть еще много промежуточных (скрытых) нейронов,

выполняющих внутренние функции. Входные, скрытые и выходные нейроны должны быть связаны между собой.

Ключевой вопрос здесь - *обратная связь* (Haykin, 1994). Простейшая сеть имеет структуру прямой передачи сигнала: Сигналы проходят от входов через скрытые элементы и в конце концов приходят на выходные элементы. Такая структура имеет устойчивое поведение. Если же сеть *рекуррентная* (т.е. содержит связи, ведущие назад от более дальних к более ближним нейронам), то она может быть неустойчива и иметь очень сложную динамику поведения. Рекуррентные сети представляют большой интерес для исследователей в области нейронных сетей, однако при решении практических задач, по крайней мере до сих пор, наиболее полезными оказались структуры прямой передачи, и именно такой тип нейронных сетей моделируется в пакете *ST Neural Networks*.

Типичный пример сети с прямой передачей сигнала показан на рисунке 4.5.

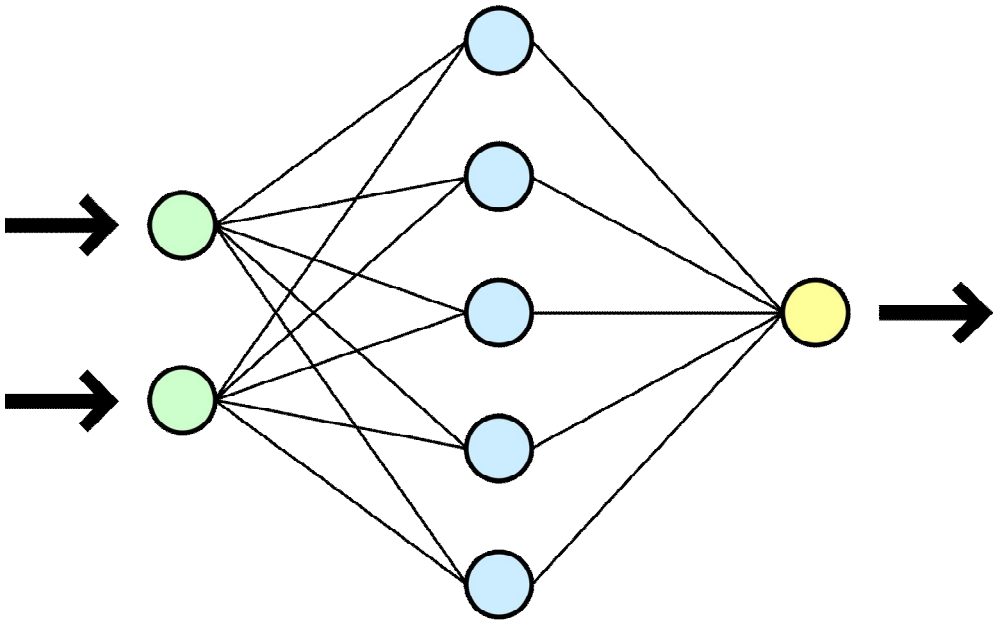

Рисунок 4.5 – Пример сети с прямой передачей сигнала

Нейроны регулярным образом организованы в слои. Входной слой служит просто для ввода значений входных переменных. Каждый из

скрытых и выходных нейронов соединен со всеми элементами предыдущего слоя. Можно было бы рассматривать сети, в которых нейроны связаны только с некоторыми из нейронов предыдущего слоя; однако, для большинства приложений сети с полной системой связей предпочтительнее, и именно такой тип сетей реализован в пакете *ST Neural Networks*.

При работе (использовании) сети во входные элементы подаются значения входных переменных, затем последовательно отрабатывают нейроны промежуточных и выходного слоев. Каждый из них вычисляет свое значение активации, беря взвешенную сумму выходов элементов предыдущего слоя и вычитая из нее пороговое значение. Затем значение активации преобразуются с помощью функции активации, и в результате получается выход нейрона. После того, как вся сеть отработает, выходные значения элементов выходного слоя принимаются за выход всей сети в целом.

Главными свойствами биологического нейрона являются его способность к обучению, универсальность, способность решать различные задачи. Описанная выше модель не способна к этому. Обучаемой она стала лишь в 1949 году благодаря Д. Хеббу (D. Hebb), который, опираясь на физиологические и психологические исследования, выдвинул гипотезу об обучаемости биологических нейронов. Его метод обучения стал отправной точкой для алгоритмов обучения нейронных сетей без учителя.

В 1957 году в мировом научном мире произошло второе по значимости в истории нейронных сетей событие: американский физиолог Ф. Розенблатт разработал модель зрительного восприятия и распознавания – персептрон (perceptron), а затем и построил первый нейрокомпьютер «Марк-1».

Нейронные сети были просты, понятны и многообещающи. Сложные процессы мышления, казалось, были готовы раскрыться перед человеком,

для их описания требовались лишь элементарные математические операции: сложение, умножение и линейная функция. На персептрон, а затем и многослойный персептрон возлагались большие надежды, поэтому бурный энтузиазм, с которым он был принят, достаточно быстро сменился жесткой критикой.

В 1969 году вышла в свет книга «Персептроны» М. Минского и С. Паперта, которая ознаменовала окончание первого этапа в истории нейронных сетей. В этой книге был проведен анализ возможностей однослойных нейронных сетей и их ограничений.

Ограничения, обнаруженные Минским, были характерны однослойным нейронным сетям, но не многослойным нейронным сетям, которым была присуща другая проблема – их обучение.

Предложенный в 1986 году Д. Румельхардом и другими учеными алгоритм обратного распространения ошибки стал одним из ведущих факторов, породивших современный нейросетевой бум, так как являлся эффективным способом обучения нейронной сети достаточно произвольной структуры.

Нейронные сети могут реализовываться как программно, так и аппаратно. Постепенно направление нейрокибернетики преобразовалось в нейрокомпьютинг – шестое поколение компьютеров, базирующееся на нейронных сетях, имеющее свои достоинства и недостатки. Сравнительный анализ машины фон Неймана, идеи которой воплощены в первых 4-х поколениях компьютеров, и биологической нейронной системы, являющейся прообразом нейрокомпьютера, представлен в таблице 4.1.

Таблица 4.1 – Сравнение машины фон Неймана с биологической нейронной системой

	Машина фон Неймана	**Биологическая нейронная система**
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализована	Распределенная
	Адресация не по содержанию	Адресация по содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	Хранимые программы	Самообучение
Надежность	Высокая уязвимость	Живучесть
Специализация	Численные и символьные операции	Проблемы восприятия
Среда функционирования	Строго определенная	Плохо определенная
	Строго ограниченная	Без ограничений

4.2.2. Классификация искусственных нейронных сетей

По топологии (рисунки 4.6, 4.7):

– полносвязные (каждый нейрон связан со всеми остальными нейронами, в том числе и сам с собой);

– многослойные (нейроны располагаются слоями, и каждый нейрон последующего слоя связан со всеми нейронами текущего слоя).

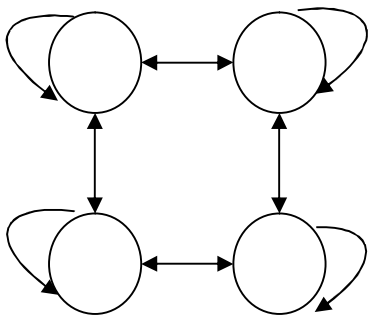

Рисунок 4.6 – Полносвязная нейронная сеть

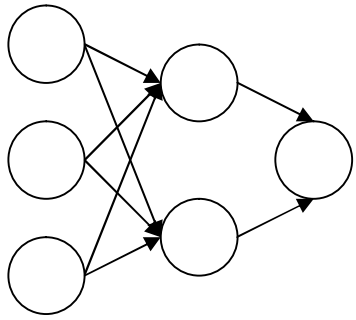

Рисунок 4.7 – Слоистая нейронная сеть

По организации обучения:

– с учителем (нейронную сеть обучают, подавая на вход значения обучающей выборки и предоставляя требуемые выходные значения);

– без учителя (на входы нейронной сети подают множество объектов, и нейронная сеть сама делит их на кластеры или классы).

По типам структур (рисунок 4.8):

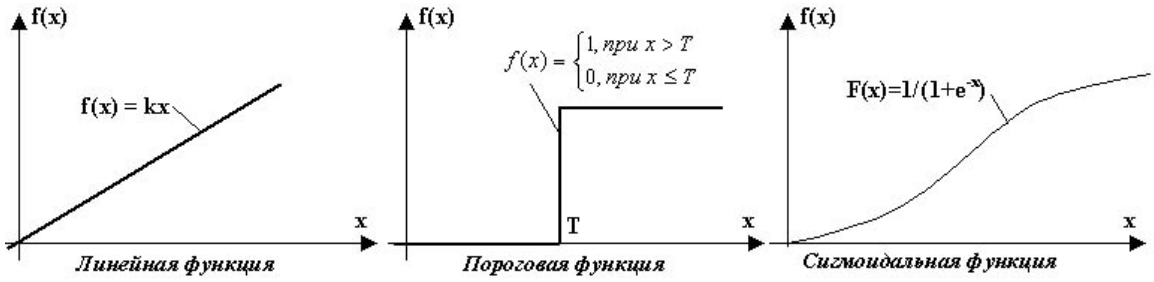

Рисунок 4.8 – Активационные функции

– нейроны с одним типом функции активации (все нейроны сети имеют одну функцию активации f(x), например, линейную);
– нейроны с несколькими типами функций активации (нейроны сети имеют различные функции активации).

По типу связей:

– прямого распространения (без обратных связей между нейронами; к таким сетям относятся однослойный и многослойный персептроны, сеть радиальных базисных функций);
– рекуррентные (с обратной связью, от выходов нейронов к входам; к таким сетям относятся соревновательные сети и сеть Хопфилда).

По типу сигнала:

– бинарные (на входы подаются только нули и единицы);
– аналоговые (на входы нейронов подаются значения непрерывных функций).

4.2.3. Задачи, решаемые нейронными сетями

Классификация образов. Задача состоит в определении принадлежности входного образа (например, языкового сигнала или рукописного символа), представленного вектором признаков, к одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание языка, классификация сигнала электрокардиограммы, классификация клеток крови.

Кластеризация/категоризация. При решении задачи кластеризации обучающее множество не имеет меток классов. Алгоритм кластеризации основан на подобии образов и помещает похожие образы в один кластер. Известны случаи применения кластеризации для добычи знаний, сжатия данных и исследования свойств данных.

Аппроксимация функций. Предположим, что есть обучающая выборка $((x_1,y_1), (x_2,y_2), ..., (x_n,y_n))$ (пары данных вход-выход), которая генерируется неизвестной функцией F, искаженной шумом. Задача аппроксимации состоит в нахождении неизвестной функции F. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

Предвидение/прогноз. Пусть заданы n дискретных отсчетов $\{y(t_1), y(t_2), ..., y(t_n)\}$ в последовательные моменты времени $t_1, t_2, ..., t_n$. Задача состоит в предвидении значения $y(t_n+1)$ в следующий момент времени t_n+1. Предвидение/прогноз имеют большое значение для принятия решений в бизнесе, науке и технике (предвидение цен на фондовой бирже, прогноз погоды).

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию.

Память, адресуемая по содержанию. В традиционных компьютерах обращение к памяти доступно только с помощью адреса, не зависящего от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совсем другая информация. Ассоциативная память или память, адресуемая по смыслу, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или при поврежденном содержании. Ассоциативная память может быть использована в мультимедийных информационных базах данных.

Управление. Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ – входное управляющее воздействие, а $y(t)$ – выход системы в момент времени t. В системах управления с эталонной моделью целью управления является расчет такого входного

воздействия u(t), при котором система действует по желательной траектории, заданной эталонной моделью. Примером является оптимальное управление двигателем.

4.2.4. Однослойные искусственные нейронные сети

Однослойный персептрон (персептрон Розенблатта) - однослойная *нейронная сеть*, все нейроны которой имеют жесткую пороговую функцию активации. Это простейшая модель *нейронной сети.*

Однослойный *персептрон* имеет простой алгоритм обучения и способен решать лишь самые простые задачи. Эта модель вызвала к себе большой интерес в начале 1960-х годов и стала толчком к развитию искусственных *нейронных сетей*.

Классический пример такой *нейронной сети* – однослойный трехнейронный *персептрон* - представлен на рисунок 4.9.

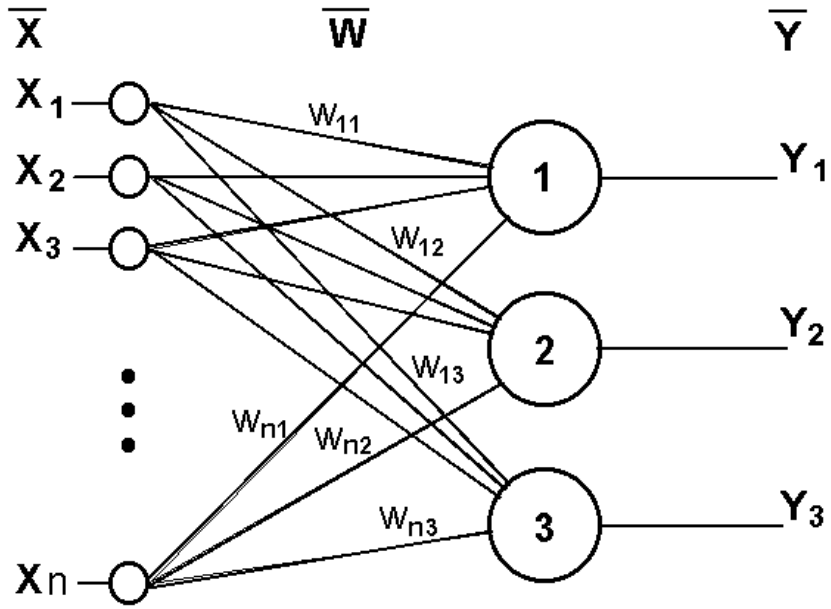

Рисунок 4.9 – Однослойный трехнейронный персептрон

Сеть, изображенная на рисунке, имеет n входов, на которые поступают сигналы, идущие по *синапсам* на 3 нейрона. Эти три нейрона образуют единственный *слой* данной сети и выдают три выходных сигнала.

Обучение по дельта-правилу

Дельта-правило является обобщением алгоритма обучения персептрона. Дельта-правило работает только с непрерывными, дифференцируемыми функциями в режиме обучения с учителем (supervised learning). Ошибка, вычисляемая в процессе обучения сети, – это функция, характеризующая качество обучения данной сети, поэтому процесс обучения нейронной сети можно представить как процесс минимизации функции ошибки. Направление изменения значения функции можно установить, вычислив производную для функции одного переменного или градиент для функции многих переменных. При минимизации значения функции многих переменных меньшее значение необходимо искать в направлении антиградиента.

В данном алгоритме обучения начальные веса могут быть любыми.

Процесс обучения можно считать завершенным, если достигнута некая заранее установленная минимальная ошибка или алгоритм проработал условленное количество раз.

Алгоритм обучения по дельта-правилу:

1 шаг: инициализация матрицы весов (и порогов, в случае использования пороговой функции активации) случайным образом.

2 шаг: предъявление нейронной сети образа (на вход подаются значения из обучающей выборки – вектор X, берется соответствующий выход – вектор D).

3 шаг: вычисление выходных значений нейронной сети (вектор Y).

4 шаг: вычисление для каждого нейрона величины расхождения реального результата с желаемым:

$\varepsilon_i = (d_i - y_i)$,

где d_i – желаемое выходное значение на i-нейроне, y_i – реальное значение на i-нейроне.

5 шаг: изменение весов (и порогов при использовании пороговой функции) по формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \varepsilon_i \cdot x_j,$$
$$\theta_i(t+1) = \theta_i(t) - \eta \cdot \varepsilon_i,$$

где t – номер текущей итерации цикла обучения, w_{ij} – вес связи j-входа с i-нейроном, η – коэффициент обучения (задается от 0 до 1), x_j – входное значение, θ_i – пороговое значение i-нейрона.

6 шаг: проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то 2 шаг, иначе заканчиваем обучение.

4.2.5. Многослойные нейронные сети

Однослойные нейронные сети имеют свои ограничения, на которые указал Минский, многослойные сети свободны от них. Они обладают большими вычислительными возможностями. Хотя созданы сети всех конфигураций, какие только можно себе представить, послойная организация нейронов копирует слоистые структуры определенных отделов мозга.

Многослойный персептрон (MLP) – *нейронная сеть* прямого распространения сигнала (без обратных связей), в которой входной сигнал преобразуется в выходной, проходя последовательно через несколько *слоев*.

Первый из таких *слоев* называют входным, последний - выходным. Эти *слои* содержат так называемые вырожденные нейроны и иногда в количестве *слоев* не учитываются. Кроме входного и выходного *слоев*, в

многослойном *персептроне* есть один или несколько промежуточных *слоев*, которые называют скрытыми.

В этой модели *персептрона* должен быть хотя бы один скрытый *слой*. Присутствие нескольких таких *слоев* оправдано лишь в случае использования нелинейных функций активации.

Пример двухслойного *персептрона* представлен на рисунок 4.10.

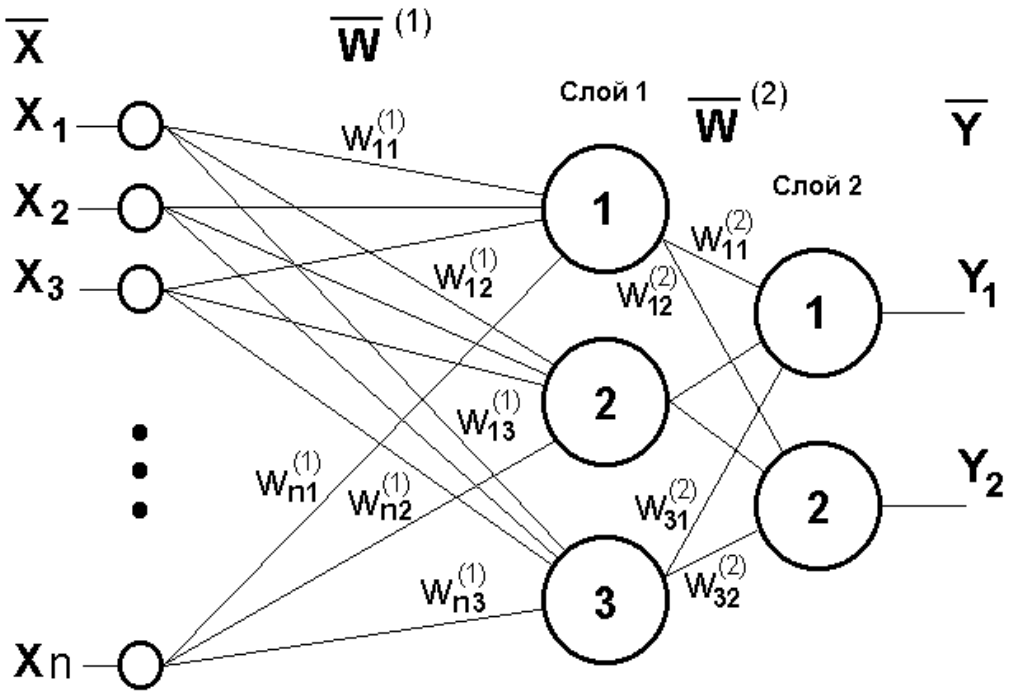

Рисунок 4.10 – Двухслойный персептрон

Сеть, изображенная на рисунке, имеет n входов. На них поступают сигналы, идущие далее по *синапсам* на 3 нейрона, которые образуют первый *слой*. Выходные сигналы первого *слоя* передаются двум нейронам второго *слоя*. Последние, в свою очередь, выдают два выходных сигнала.

Обучение методом обратного распространения ошибки

Обучение алгоритмом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямой и обратный.

При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы.

Во время обратного прохода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название – алгоритм обратного распространения ошибки. Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому.

Алгоритм обучения по дельта-правилу:

1 шаг: инициализация матриц весов случайным образом (в циклах).

2 шаг: предъявление нейронной сети образа (на вход подаются значения из обучающей выборки – вектор X и берется соответствующий выход – вектор D).

3 шаг (прямой проход): вычисление в циклах выходов всех слоев и получение выходных значений нейронной сети (вектор Y):

$$y_i^k = f\left(\sum_{j=0}^{H_{k-1}} w_{ij}^k \cdot y_j^{k-1}\right),$$

$$y_j^0 = x_j,$$

$$y_0^{k-1} = 1,$$

$$x_0 = 1,$$

где y_i^k – выход i-нейрона k-слоя, f – функция активации, w_{ij}^k – синаптическая связь между j-нейроном слоя k-1 и i-нейроном слоя k, x_j – входное значение.

4 шаг (обратный проход): изменение весов в циклах по формулам:

$$w_{ij}^{k}(t+1) = w_{ij}^{k}(t) + \eta \cdot \delta_i^k \cdot y_j^{k-1},$$

$$\delta_i^k = (d_i - y_i) \cdot y_i \cdot (1 - y_i)$$

– для последнего (выходного) слоя,

$$\delta_i^k = y_i \cdot (1 - y_i) \cdot \sum_{l=1}^{H_{k+1}} \delta_l^{k+1} \cdot w_l^{k+1}$$

– для промежуточных слоев, где t – номер текущей итерации цикла обучения (номер эпохи), η – коэффициент обучения (задается от 0 до 1), y_i^k – выход i-нейрона k-слоя, w_{ij}^k – синаптическая связь между j-нейроном слоя k-1 и i-нейроном слоя k, d_i – желаемое выходное значение на i-нейроне, y_i – реальное значение на i-нейроне выходного слоя.

5 шаг: проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то 2 шаг, иначе заканчиваем обучение. Среднеквадратичная ошибка вычисляется следующим образом:

$$\varepsilon = \frac{1}{Q} \cdot \sum_{q=1}^{Q} \sum_{i=1}^{H} (d_i - y_i)^2,$$

где Q – общее число примеров, H – количество нейронов в выходном слое, d_i – желаемое выходное значение на i-нейроне, y_i – реальное значение на i-нейроне выходного слоя.

Другие алгоритмы обучения MLP

Выше было описано, как с помощью алгоритма обратного распространения осуществляется градиентный спуск по поверхности ошибок. Вкратце дело происходит так: в данной точке поверхности находится направление скорейшего спуска, затем делается прыжок вниз на расстояние, пропорциональное коэффициенту скорости обучения и крутизне склона, при этом учитывается инерция, те есть стремление сохранить прежнее направление движения. Можно сказать, что метод ведет себя как слепой кенгуру - каждый раз прыгает в направлении,

которое кажется ему наилучшим. На самом деле шаг спуска вычисляется отдельно для всех обучающих наблюдений, взятых в случайном порядке, но в результате получается достаточно хорошая аппроксимация спуска по совокупной поверхности ошибок. Существуют и другие алгоритмы обучения MLP, однако все они используют ту или иную стратегию скорейшего продвижения к точке минимума.

В некоторых задачах бывает целесообразно использовать такие - более сложные - методы нелинейной оптимизации. В пакете *ST Neural Networks* реализованы два подобных метода: методы спуска по сопряженным градиентам и *Левенберга -Маркара*, представляющие собой очень удачные варианты реализации двух типов алгоритмов: линейного поиска и доверительных областей.

Алгоритм линейного поиска действует следующим образом: выбирается какое-либо разумное направление движения по многомерной поверхности. В этом направлении проводится линия, и на ней ищется точка минимума (это делается относительно просто с помощью того или иного варианта метода деления отрезка пополам); затем все повторяется сначала. Что в данном случае следует считать "разумным направлением"? Очевидным ответом является направление скорейшего спуска (именно так действует алгоритм обратного распространения). На самом деле этот вроде бы очевидный выбор не слишком удачен. После того, как был найден минимум по некоторой прямой, следующая линия, выбранная для кратчайшего спуска, может "испортить" результаты минимизации по предыдущему направлению (даже на такой простой поверхности, как параболоид, может потребоваться очень большое число шагов линейного поиска). Более разумно было бы выбирать "не мешающие друг другу " направления спуска - так мы приходим к методу сопряженных градиентов.

Идея метода состоит в следующем: поскольку мы нашли точку минимума вдоль некоторой прямой, производная по этому направлению

равна нулю. Сопряженное направление выбирается таким образом, чтобы эта производная и дальше оставалась нулевой - в предположении, что поверхность имеет форму параболоида (или, грубо говоря, является "хорошей и гладкой "). Если это условие выполнено, то для достижения точки минимума достаточно будет N эпох. На реальных, сложно устроенных поверхностях по мере хода алгоритма условие сопряженности портится, и тем не менее такой алгоритм, как правило, требует гораздо меньшего числа шагов, чем метод обратного распространения, и дает лучшую точку минимума (для того, чтобы алгоритм обратного распространения точно установился в некоторой точке, нужно выбирать очень маленькую скорость обучения).

Метод доверительных областей основан на следующей идее: вместо того, чтобы двигаться в определенном направлении поиска, предположим, что поверхность имеет достаточно простую форму, так что точку минимума можно найти (и прыгнуть туда) непосредственно. Попробуем смоделировать это и посмотреть, насколько хорошей окажется полученная точка. Вид модели предполагает, что поверхность имеет хорошую и гладкую форму (например, является параболоидом), - такое предположение выполнено вблизи точек минимума. Вдали от них данное предположение может сильно нарушаться, так что модель будет выбирать для очередного продвижения совершенно не те точки. Правильно работать такая модель будет только в некоторой окрестности данной точки, причем размеры этой окрестности заранее неизвестны. Поэтому выберем в качестве следующей точки для продвижения нечто промежуточное между точкой, которую предлагает наша модель, и точкой, которая получилась бы по обычному методу градиентного спуска. Если эта новая точка оказалась хорошей, передвинемся в нее и усилим роль нашей модели в выборе очередных точек; если же точка оказалась плохой, не будем в нее перемещаться и увеличим роль метода градиентного спуска при выборе

очередной точки (а также уменьшим шаг). В основанном на этой идее методе *Левенберга-Маркара* предполагается, что исходное отображение является локально линейным (и тогда поверхность ошибок будет параболоидом).

Метод *Левенберга-Маркара* - самый быстрый алгоритм обучения из всех, которые реализованы в пакете *ST Neural Networks*, но, к сожалению, на его использование имеется ряд важных ограничений. Он применим только для сетей с одним выходным элементом, работает только с функцией ошибок сумма квадратов и требует памяти порядка W^2 (где W - количество весов у сети; поэтому для больших сетей он плохо применим). Метод сопряженных градиентов почти так же эффективен, как и этот метод, и не связан подобными ограничениями.

При всем сказанном метод *обратного распространения* также сохраняет свое значение, причем не только для тех случаев, когда требуется быстро найти решение (и не требуется особой точности). Его следует предпочесть, когда объем данных очень велик, и среди данных есть избыточные. Благодаря тому, что в методе *обратного распространения* корректировка ошибки происходит по отдельным случаям, избыточность данных не вредит (если, например, приписать к имеющемуся набору данных еще один точно такой же набор, так что каждый случай будет повторяться дважды, то эпоха будет занимать вдвое больше времени, чем раньше, однако результат ее будет точно таким же, как от двух старых, так что ничего плохого не произойдет). Методы же *Левенберга-Маркара* и *сопряженных градиентов* проводят вычисления на всем наборе данных, поэтому при увеличении числа наблюдений продолжительность одной эпохи сильно растет, но при этом совсем не обязательно улучшается результат, достигнутый на этой эпохе (в частности, если данные избыточны; если же данные редкие, то добавление новых данных улучшит обучение на каждой эпохе). Кроме того, *обратное*

распространение не уступает другим методам в ситуациях, когда данных мало, поскольку в этом случае недостаточно данных для принятия очень точного решения (более тонкий алгоритм может дать меньшую ошибку обучения, но контрольная ошибка у него, скорее всего, не будет меньше).

Кроме уже перечисленных, в пакете *ST Neural Networks* имеются две модификации метода обратного распространения - метод быстрого распространения и дельта-дельта с чертой, - разработанные с целью преодолеть некоторые ограничения этого подхода. В большинстве случаев они работают не лучше, чем *обратное распространение*, а иногда и хуже (это зависит от задачи). Кроме того, в этих методах используется больше управляющих параметров, чем в других методах, и поэтому ими сложнее пользоваться.

4.3. МОДЕЛИ НЕЙРОННЫХ СЕТЕЙ

4.3.1. Вероятностная нейронная сеть

Говоря о задачах классификации, мы кратко упомянули о том, что выходы сети можно с пользой интерпретировать как оценки вероятности того, что элемент принадлежит некоторому классу, и сеть фактически учится оценивать функцию плотности вероятности. Аналогичная полезная интерпретация может иметь место и в задачах регрессии – выход сети рассматривается как ожидаемое значение модели в данной точке пространства входов. Это ожидаемое значение связано с плотностью вероятности совместного распределения входных и выходных данных.

Задача оценки плотности вероятности (p.d.f.) по данным имеет давнюю историю в математической статистике и относится к области байесовой статистики. Обычная статистика по заданной модели говорит нам, какова будет вероятность того или иного исхода (например, что на игральной кости шесть очков будет выпадать в среднем одном случае из шести). Байесова статистика переворачивает вопрос вверх ногами:

правильность модели оценивается по имеющимся достоверным данным. В более общем плане, байесова статистика дает возможность оценивать плотность вероятности распределений параметров модели по имеющимся данных. Для того, чтобы минимизировать ошибку, выбирается модель с такими параметрами, при которых плотность вероятности будет наибольшей.

При решении задачи классификации можно оценить плотность вероятности для каждого класса, сравнить между собой вероятности принадлежности различным классам и выбрать наиболее вероятный. На самом деле именно это происходит, когда мы обучаем нейронную сеть решать задачу классификации – сеть пытается определить (т.е. аппроксимировать) плотность вероятности.

Традиционный подход к задаче состоит в том, чтобы построить оценку для плотности вероятности по имеющимся данным. Обычно при этом предполагается, что плотность имеет некоторый определенный вид (чаще всего - что она имеет нормальное распределение). После этого оцениваются параметры модели. Нормальное распределение часто используется потому, что тогда параметры модели (среднее и стандартное отклонение) можно оценить аналитически. При этом остается вопрос о том, что предположение о нормальности не всегда оправдано.

Другой подход к оценке плотности вероятности основан на *ядерных оценках*. Можно рассуждать так: тот факт, что наблюдение расположено в данной точке пространства, свидетельствует о том, что в этой точке имеется некоторая плотность вероятности. Кластеры из близко лежащих точек указывают на то, что в этом месте плотность вероятности большая. Вблизи наблюдения имеется большее доверие к уровню плотности, а по мере отдаления от него доверие убывает и стремится к нулю. В методе ядерных оценок в точке, соответствующей каждому наблюдению, помещается некоторая простая функция, затем все они складываются и в

результате получается оценка для общей плотности вероятности. Чаще всего в качестве ядерных функций берутся гауссовы функции (с формой колокола). Если обучающих примеров достаточное количество, то такой метод дает достаточно хорошее приближение к истинной плотности вероятности.

Метод аппроксимации плотности вероятности с помощью ядерных функций во многом похож на метод радиальных базисных функций, и таким образом мы естественно приходим к понятиям вероятностной нейронной сети (PNN) и обобщенно-регрессионной нейронной сети (GRNN). PNN-сети предназначены для задач классификации, а GRNN - для задач регрессии. Сети этих двух типов представляют собой реализацию методов ядерной аппроксимации, оформленных в виде нейронной сети.

Сеть PNN имеет по меньшей мере три слоя: входной, радиальный и выходной. Радиальные элементы берутся по одному на каждое обучающее наблюдение. Каждый из них представляет гауссову функцию с центром в этом наблюдении. Каждому классу соответствует один выходной элемент. Каждый такой элемент соединен со всеми радиальными элементами, относящимися к его классу, а со всеми остальными радиальными элементами он имеет нулевое соединение. Таким образом, выходной элемент просто складывает отклики всех элементов, принадлежащих к его классу. Значения выходных сигналов получаются пропорциональными ядерным оценкам вероятности принадлежности соответствующим классам, и пронормировав их на единицу, мы получаем окончательные оценки вероятности принадлежности классам.

Базовая модель PNN-сети может иметь две модификации.

В первом случае мы предполагаем, что пропорции классов в обучающем множестве соответствуют их пропорциям во всей исследуемой популяции (или так называемым априорным вероятностям). Например, если среди всех людей больными являются 2%, то в обучающем

множестве для сети, диагностирующей заболевание, больных должно быть тоже 2%. Если же априорные вероятности будут отличаться от пропорций в обучающей выборке, то сеть будет выдавать неправильный результат. Это можно впоследствии учесть (если стали известны априорные вероятности), вводя поправочные коэффициенты для различных классов.

Второй вариант модификации основан на следующей идее. Любая оценка, выдаваемая сетью, основывается на зашумленных данных и неизбежно будет приводить к отдельным ошибкам классификации (например, у некоторых больных результаты анализов могут быть вполне нормальными). Иногда бывает целесообразно считать, что некоторые виды ошибок обходятся "дороже" других (например, если здоровый человек будет диагностирован как больной, то это вызовет лишние затраты на его обследование, но не создаст угрозы для жизни; если же не будет выявлен действительный больной, то это может привести к смертельному исходу). В такой ситуации те вероятности, которые выдает сеть, следует домножить на коэффициенты потерь, отражающие относительную цену ошибок классификации. В пакете *ST Neural Networks* в вероятностную нейронную сеть может быть добавлен четвертый слой, содержащий матрицу потерь. Она умножается на вектор оценок, полученный в третьем слое, после чего в качестве ответа берется класс, имеющий наименьшую оценку потерь. (Матрицу потерь можно добавлять и к другим видам сетей, решающих задачи классификации.)

Вероятностная нейронная сеть имеет единственный управляющий параметр обучения, значение которого должно выбираться пользователем, - степень сглаживания (или отклонение гауссовой функции). Как и в случае RBF-сетей, этот параметр выбирается из тех соображений, чтобы шапки " определенное число раз перекрывались": выбор слишком маленьких отклонений приведет к "острым" аппроксимирующим функциям и неспособности сети к обобщению, а при слишком больших

отклонениях будут теряться детали. Требуемое значение несложно найти опытным путем, подбирая его так, чтобы контрольная ошибка была как можно меньше. К счастью, PNN-сети не очень чувствительны к выбору параметра сглаживания.

Наиболее важные преимущества PNN-сетей состоят в том, что выходное значение имеет вероятностный смысл (и поэтому его легче интерпретировать), и в том, что сеть быстро обучается. При обучения такой сети время тратится практически только на то, чтобы подавать ей на вход обучающие наблюдения, и сеть работает настолько быстро, насколько это вообще возможно.

Существенным недостатком таких сетей является их объем. PNN-сеть фактически вмещает в себя все обучающие данные, поэтому она требует много памяти и может медленно работать.

PNN-сети особенно полезны при пробных экспериментах (например, когда нужно решить, какие из входных переменных использовать), так как благодаря короткому времени обучения можно быстро проделать большое количество пробных тестов. В пакете *ST Neural Networks* PNN-сети используются также в *Нейро-генетическом алгоритме отбора входных данных - Neuro-Genetic Input Selection*, который автоматически находит значимые входы (будет описан ниже).

4.3.2. Обобщенно-регрессионная нейронная сеть

Обобщенно-регрессионная нейронная сеть (GRNN) устроена аналогично вероятностной нейронной сети (PNN), но она предназначена для решения задач регрессии, а не классификации. Как и в случае PNN-сети, в точку расположения каждого обучающего наблюдения помещается гауссова ядерная функция. Мы считаем, что каждое наблюдение свидетельствует о некоторой нашей уверенности в том, что поверхность отклика в данной точке имеет определенную высоту, и эта уверенность

убывает при отходе в сторону от точки. GRNN-сеть копирует внутрь себя все обучающие наблюдения и использует их для оценки отклика в произвольной точке. Окончательная выходная оценка сети получается как взвешенное среднее выходов по всем обучающим наблюдениям, где величины весов отражают расстояние от этих наблюдений до той точки, в которой производится оценивание (и, таким образом, более близкие точки вносят больший вклад в оценку).

Первый промежуточный слой сети GRNN состоит из радиальных элементов. Второй промежуточный слой содержит элементы, которые помогают оценить взвешенное среднее. Для этого используется специальная процедура. Каждый выход имеет в этом слое свой элемент, формирующий для него взвешенную сумму. Чтобы получить из взвешенной суммы взвешенное среднее, эту сумму нужно поделить на сумму весовых коэффициентов. Последнюю сумму вычисляет специальный элемент второго слоя. После этого в выходном слое производится собственно деление (с помощью специальных элементов "деления"). Таким образом, число элементов во втором промежуточном слое на единицу больше, чем в выходном слое. Как правило, в задачах регрессии требуется оценить одно выходное значение, и, соответственно, второй промежуточный слой содержит два элемента.

Можно модифицировать GRNN-сеть таким образом, чтобы радиальные элементы соответствовали не отдельным обучающим случаям, а их кластерам. Это уменьшает размеры сети и увеличивает скорость обучения. Центры для таких элементов можно выбирать с помощью любого предназначенного для этой цели алгоритма (выборки из выборки, *K*-средних или Кохонена), и программа *ST Neural Networks* соответствующим образом корректирует внутренние веса.

Достоинства и недостатки у сетей GRNN в основном такие же, как и у сетей PNN - единственное различие в том, что GRNN используются в

задачах регрессии, а PNN - в задачах классификации. GRNN-сеть обучается почти мгновенно, но может получиться большой и медленной (хотя здесь, в отличие от PNN, не обязательно иметь по одному радиальному элементу на каждый обучающий пример, их число все равно будет большим). Как и сеть RBF, сеть GRNN не обладает способностью экстраполировать данные.

4.3.3. Линейная сеть

Согласно общепринятому в науке принципу, если более сложная модель не дает лучших результатов, чем более простая, то из них следует предпочесть вторую. В терминах аппроксимации отображений самой простой моделью будет линейная, в которой подгоночная функция определяется гиперплоскостью. В задаче классификации гиперплоскость размещается таким образом, чтобы она разделяла собой два класа (линейная дискриминантная функция); в задаче регрессии гиперплоскость должна проходить через заданные точки. Линейная модель обычно записывается с помощью матрицы *NxN* и вектора смещения размера *N*.

На языке нейронных сетей линейная модель представляется сетью без промежуточных слоев, которая в выходном слое содержит только линейные элементы (то есть элементы с линейной функцией активации). Веса соответствуют элементам матрицы, а пороги - компонентам вектора смещения. Во время работы сеть фактически умножает вектор входов на матрицу весов, а затем к полученному вектору прибавляет вектор смещения.

В пакете ST Neural Networks имеется возможность создать линейную сеть и обучить ее с помощью стандартного алгоритма линейной оптимизации, основанного на псевдообратных матрицах (SVD) (Golub and Kahan, 1965). Разумеется, метод линейной оптимизации реализован также в модуле *Множественная регрессия* системы *STATISTICA*; однако,

линейные сети пакета *ST Neural Networks* имеют то преимущество, что здесь Вы можете в единой среде сравнивать такие сети с "настоящими" нейронными сетями.

Линейная сеть является хорошей точкой отсчета для оценки качества построенных Вами нейронных сетей. Может оказаться так, что задачу, считавшуюся очень сложной, можно успешно не только нейронной сетью, но и простым линейным методом. Если же в задаче не так много обучающих данных, то, вероятно, просто нет оснований использовать более сложные модели.

4.3.4. Сеть Кохонена

Сети Кохонена принципиально отличаются от всех других типов сетей, реализованных в пакете *ST Neural Networks*. В то время как все остальные сети предназначены для задач с управляемым обучением, сети Кохонена главным образом рассчитана на неуправляемое обучение.

При управляемом обучении наблюдения, составляющие обучающие данные, вместе с входными переменными содержат также и соответствующие им выходные значения, и сеть должна восстановить отображение, переводящее первые во вторые. В случае же неуправляемого обучения обучающие данные содержат только значения входных переменных.

Сеть Кохонена учится понимать саму структуру данных.

Одно из возможных применений таких сетей - разведочный анализ данных. Сеть Кохонена может распознавать кластеры в данных, а также устанавливать близость классов. Таким образом пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель. Если в данных распознаны классы, то их можно обозначить, после чего сеть сможет решать задачи классификации. Сети Кохонена можно использовать и в тех задачах классификации, где классы

уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.

Другая возможная область применения - обнаружение новых явлений. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам. Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.

Сеть Кохонена имеет всего два слоя: входной и выходной, составленный из радиальных элементов (выходной слой называют также слоем топологической карты). Элементы топологической карты располагаются в некотором пространстве - как правило двумерном (в пакете *ST Neural Networks* реализованы также одномерные сети Кохонена).

Обучается сеть Кохонена методом последовательных приближений. Начиная со случайным образом выбранного исходного расположения центров, алгоритм постепенно улучшает его так, чтобы улавливать кластеризацию обучающих данных. В некотором отношении эти действия похожи на алгоритмы выборки из выборки и K-средних, которые используются для размещения центров в сетях RBF и GRNN, и действительно, алгоритм Кохонена можно использовать для размещения центров в сетях этих типов. Однако, данный алгоритм работает и на другом уровне.

Помимо того, что уже сказано, в результате итеративной процедуры обучения сеть организуется таким образом, что элементы, соответствующие центрам, расположенным близко друг от друга в пространстве входов, будут располагаться близко друг от друга и на топологической карте. Топологический слой сети можно представлять себе как двумерную решетку, которую нужно так отобразить в N-мерное пространство входов, чтобы по возможности сохранить исходную структуру данных. Конечно же, при любой попытке представить N-мерное

пространство на плоскости будут потеряны многие детали; однако, такой прием иногда полезен, так как он позволяет пользователю визуализировать данные, которые никаким иным способом понять невозможно.

Основной итерационный алгоритм Кохонена последовательно проходит одну за другой ряд эпох, при этом на каждой эпохе он обрабатывает каждый из обучающих примеров, и затем применяет следующий алгоритм:

Выбрать выигравший нейрон (то есть тот, который расположен ближе всего к входному примеру);

Скорректировать выигравший нейрон так, чтобы он стал более похож на этот входной пример (взяв взвешенную сумму прежнего центра нейрона и обучающего примера).

В алгоритме при вычислении взвешенной суммы используется постепенно убывающий коэффициент скорости обучения, с тем чтобы на каждой новой эпохе коррекция становилась все более тонкой. В результате положение центра установится в некоторой позиции, которая удовлетворительным образом представляет те наблюдения, для которых данный нейрон оказался выигравшим.

Свойство топологической упорядоченности достигается в алгоритме с помощью дополнительного использования понятия окрестности. Окрестность - это несколько нейронов, окружающих выигравший нейрон. Подобно скорости обучения, размер окрестности убывает со временем, так что вначале к ней принадлежит довольно большое число нейронов (возможно, почти вся топологическая карта); на самых последних этапах окрестность становится нулевой (т.е. состоящей только из самого выигравшего нейрона). На самом деле в алгоритме Кохонена корректировка применяется не только к выигравшему нейрону, но и ко всем нейронам из его текущей окрестности.

Результатом такого изменения окрестностей является то, что изначально довольно большие участки сети "перетягиваются" - и притом заметно - в сторону обучающих примеров. Сеть формирует грубую структуру топологического порядка, при которой похожие наблюдения активируют группы близко лежащих нейронов на топологической карте. С каждой новой эпохой скорость обучения и размер окрестности уменьшаются, тем самым внутри участков карты выявляются все более тонкие различия, что в конце концов приводит к тонкой настройке каждого нейрона. Часто обучение умышленно разбивают на две фазы: более короткую, с большой скоростью обучения и большими окрестностями, и более длинную с малой скоростью обучения и нулевыми или почти нулевыми окрестностями.

После того, как сеть обучена распознаванию структуры данных, ее можно использовать как средство визуализации при анализе данных. С помощью данных, выводимых в окне *Частоты выигрышей - Win Frequencies*, (где для каждого нейрона подсчитывается, сколько раз он выигрывал при обработке обучающих примеров), можно определить, разбивается ли карта на отдельные кластеры. Можно также обрабатывать отдельные наблюдения и смотреть, как при этом меняется топологическая карта, - это позволяет понять, имеют ли кластеры какой-то содержательный смысл (как правило при этом приходится возвращаться к содержательному смыслу задачи, чтобы установить, как соотносятся друг с другом кластеры наблюдений). После того, как кластеры выявлены, нейроны топологической карты помечаются содержательными по смыслу метками (в некоторых случаях помечены могут быть и отдельные наблюдения). После того, как топологическая карта в описанном здесь виде построена, на вход сети можно подавать новые наблюдения. Если выигравший при этом нейрон был ранее помечен именем класса, то сеть

осуществляет классификацию. В противном случае считается, что сеть не приняла никакого решения.

При решении задач классификации в сетях Кохонена используется так называемый порог доступа. Ввиду того, что в такой сети уровень активации нейрона есть расстояние от него до входного примера, порог доступа играет роль максимального расстояния, на котором происходит распознавание. Если уровень активации выигравшего нейрона превышает это пороговое значение, то сеть считается не принявшей никакого решения. Поэтому, когда все нейроны помечены, а пороги установлены на нужном уровне, сеть Кохонена может служить как детектор новых явлений (она сообщает о непринятии решения только в том случае, если поданный ей на вход случай значительно отличается от всех радиальных элементов).

Идея сети Кохонена возникла по аналогии с некоторыми известными свойствами человеческого мозга. Кора головного мозга представляет собой большой плоский лист (площадью около 0.5 кв.м.; чтобы поместиться в черепе, она свернута складками) с известными топологическими свойствами (например, участок, ответственный за кисть руки, примыкает к участку, ответственному за движения всей руки, и таким образом все изображение человеческого тела непрерывно отображается на эту двумерную поверхность).

4.4. НЕЙРОННЫЕ СЕТИ ГЛУБОКОГО ОБУЧЕНИЯ

4.4.1. Контролируемое и неконтролируемое обучение

Контролируемое обучение (обучение с учителем, supervised learning) подразумевает использование помеченных наборов данных, содержащих входные данные и ожидаемые выходные результаты. Когда вы обучаете нейронную сеть с помощью контролируемого обучения, вы подаете как входные данные, так и ожидаемые выходные результаты.

Если результат, генерируемый нейронной сетью, неверен, она скорректирует свои вычисления. Это итерационный процесс, оканчивающийся тогда, когда сеть перестает совершать ошибки.

Примером задачи с контролируемым обучением является предсказание погоды. Нейросеть учится делать прогноз погоды с использованием исторических данных. Обучающие данные включают в себя входные данные (давление, влажность, скорость ветра) и выходные результаты (температура).

Неконтролируемое обучение (обучение без учителя, unsupervised learning) — *это машинное обучение с использованием наборов данных без определенной структуры.*

Когда вы обучаете нейросеть неконтролируемо, она самостоятельно проводит логическую классификацию данных. Примером задачи с неконтролируемым обучением является предсказание поведения посетителей интернет-магазинов. В этом случае сеть не обучается на размеченных даннх. Вместо этого она самостоятельно классифицирует входные данные и отвечает на вопрос, какие пользователи чаще всего покупают различные товары.

4.4.2. Глубокое обучение

Глубокое обучение – это метод машинного обучения. Глубокое обучение позволяет обучать модель предсказывать результат по набору входных данных. Для обучения сети можно использовать как контролируемое, так и неконтролируемое обучение.

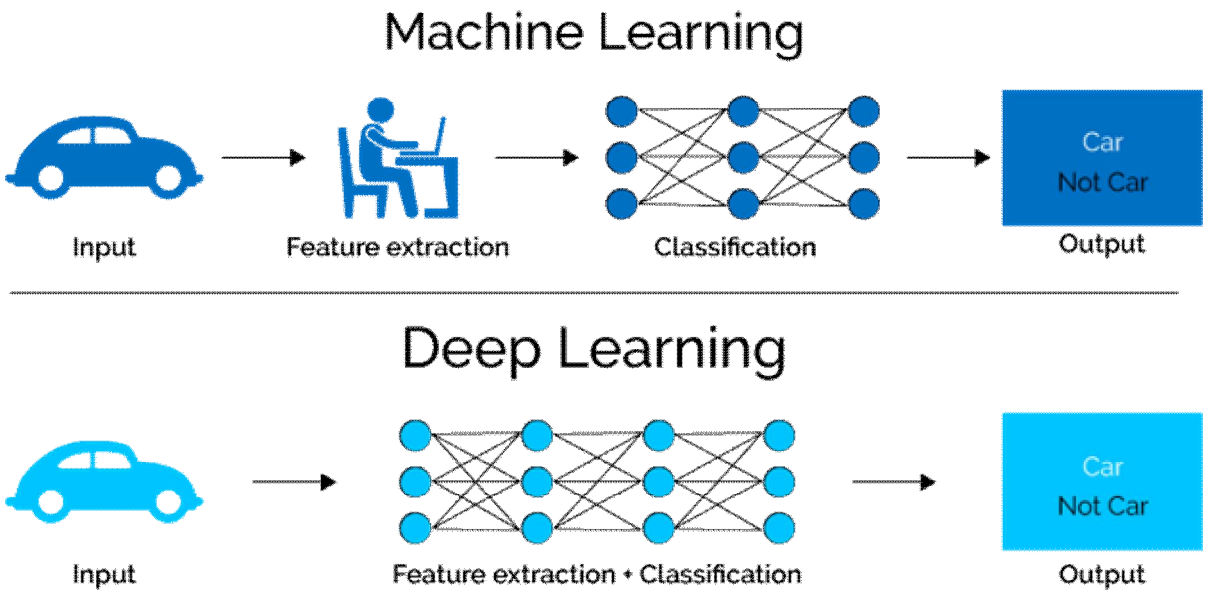

Рисунок 4.11 – Разница между машинным и глубоким обучением

Рассмотрим, как работает глубокое обучение, на примере **сервиса по оценке стоимости авиабилета**. Мы будем обучать его контролируемым образом.

Мы хотим, чтобы наш сервис предсказывал цену на авиабилет по следующим входным данным:

- аэропорт вылета;
- аэропорт назначения;
- дата отбытия;
- авиакомпания.

Давайте заглянем внутрь нашей модели. Как и у животных, искусственная нейронная сеть содержит взаимосвязанные нейроны. На диаграмме они представлены кругами:

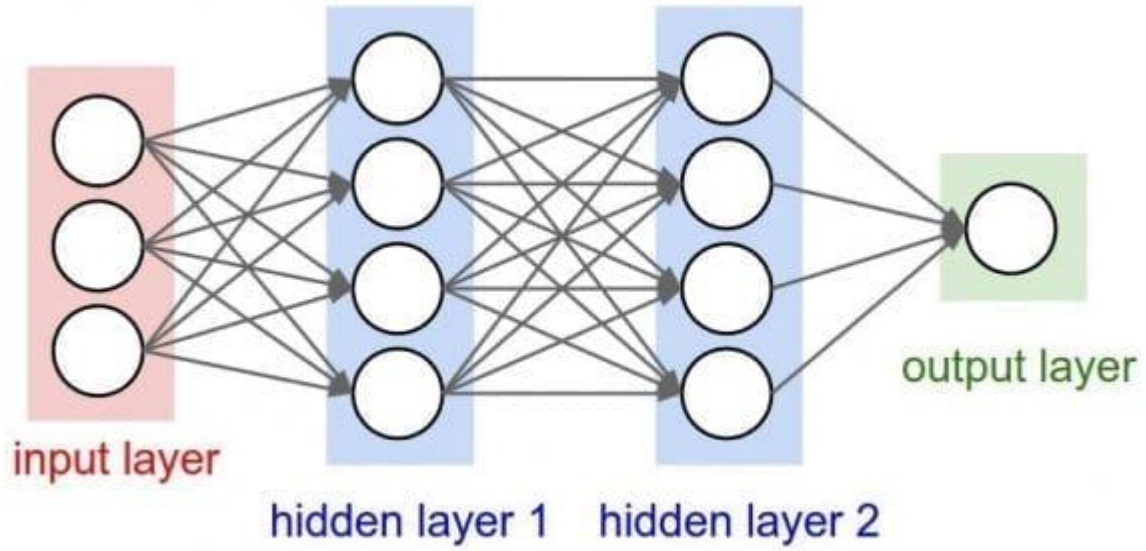

Рисунок 4.12 – Глубокая нейронная сеть (с двумя скрытыми слоями)

Нейроны сгруппированы в три различных типа слоев:

- входной слой;
- скрытый слой (слои);
- выходной слой.

Входной слой принимает входные данные. В нашем случае имеется четыре нейрона на входном слое: аэропорт вылета, аэропорт назначения, дата вылета и авиакомпания. Входной уровень передает эти данные в первый скрытый слой.

Скрытые слои выполняют математические вычисления со входными данными. Одна из задач при создании нейронных сетей — определение количества скрытых слоев и нейронов на каждом слое.

Слово «глубина» в термине «глубокое обучение» означает наличие **более чем одного скрытого слоя.**

Выходной слой выдает результат. В нашем случае это прогноз цены на билет.

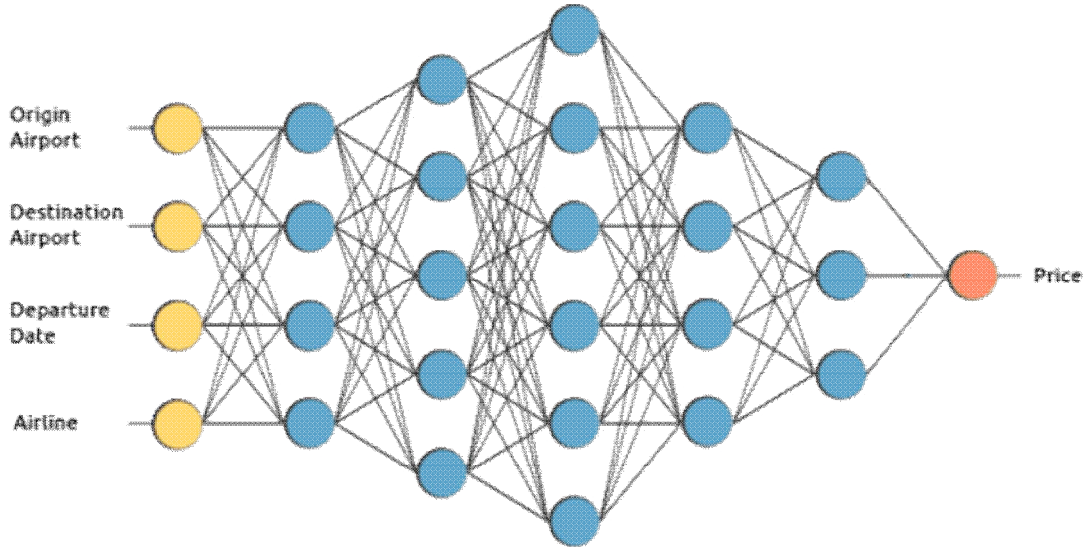

Рисунок 4.13 – Глубокая нейронная сеть вычисления цены билета

Цена вычисляется следующим образом. Нейроны связаны между собой с определенным **весом**. Вес определяет важность элемента входных данных. Исходные веса задаются случайным образом.

При прогнозировании цены на билет дата вылета является одним из наиболее важных факторов. Следовательно, связи нейрона времени вылета будут иметь большой вес.

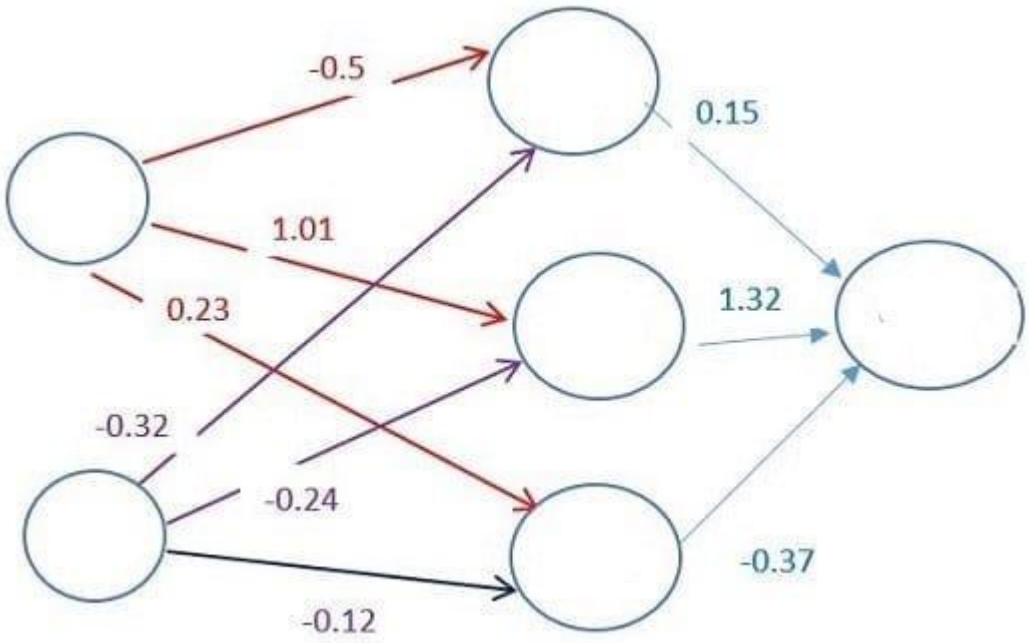

Рисунок 4.14 – Глубокая нейронная сеть с проставленными весами

Каждый нейрон имеет функцию активации. Ее смысл трудно понять без привлечения математических рассуждений. Одной из ее целей является «стандартизация» данных на выходе из нейрона.

После того, как набор входных данных прошел через все слои нейронной сети, функция активации возвращает выходные результаты через выходной уровень.

4.4.3. Обучение глубокой сети

Обучение нейросети – самая сложная часть глубокого обучения, поскольку требует **большого набора данных** и большого количества вычислительной мощности.

Для оценки стоимости билета нужно найти исторические данные о ценах на билеты. Из-за большого количества возможных комбинаций аэропортов и дат вылета нам нужен очень большой список цен на билеты.

Для обучения сети нужно подать в нее подготовленные данные и сравнить сгенерированные ей выходные результаты с результатами из нашего тестового набора данных. Поскольку сеть еще не обучена, результаты будут неверными.

После пропуска всех данных можно определить функцию, которая будет показывать нам, насколько результаты работы алгоритма отличаются от реальных данных. Эта функция называется функцией потерь.

В идеале функция потерь должна быть равна нулю. В этом случае выходные результаты работы сети полностью совпадают с результатами тестового набора данных.

Для уменьшения значения функции потерь необходимо менять веса между нейронами. Можно делать это случайным образом до тех пор, пока функция потерь не станет равной нулю, но это не очень эффективно.

Вместо этого мы будем использовать метод градиентного спуска. **Градиентный спуск** – это метод, который позволяет найти минимум функции. В нашем случае мы ищем минимум функции потерь.

Суть метода состоит в небольшом **изменении весов после каждой итерации**. Вычисляя производную (или градиент) функции потерь при определенном наборе весов, можно определить, в каком направлении находится минимум.

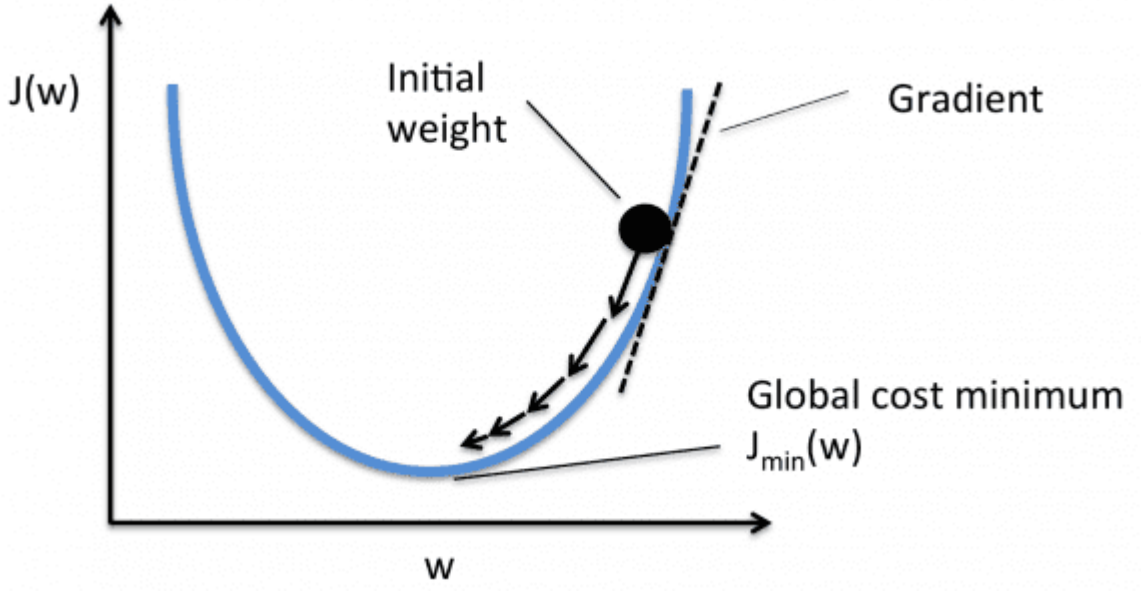

Рисунок 4.15 – Градиентный спуск

Для минимизации функции потерь нужно многократно перебирать данные. Именно поэтому нам требуется большая вычислительная мощность. Уточнение весов с помощью градиентного спуска выполняется **автоматически**. В этом и состоит магия глубокого обучения!

После обучения можно использовать разработанный нами сервис для прогнозирования цен на авиабилеты.

4.5. СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ И МАШИННОГО ЗРЕНИЯ

В настоящее время эти системы получили широкое распространение на промышленных и строительных предприятиях для автоматизации ввода текстовой и графической информации с бумажных носителей, а также с фотографий, полученных с видеокамер, в том числе в процессе аэрофотосъемки или тахеометрии.

4.5.1. Основные принципы и целостность восприятия

В основе фонтанного преобразования лежит принцип целостности. В соответствии с ним любой воспринимаемый объект рассматривается как целое, состоящее из частей, связанных между собой определенными отношениями. Так, например, печатная страница состоит из статей, статья – из заголовка и колонок, колонка – из абзацев, абзацы – из строк, строки – из слов, слова – из букв. При этом все перечисленные элементы текста связаны между собой определенными пространствами и языковыми отношениями.

Для выделения целого требуется определить его части. Части же, в свою очередь, можно рассматривать только в составе целого. Поэтому целостный процесс восприятия может происходить только в рамках гипотезы о воспринимаемом объекте – целом. После того как выдвинуто предположение о воспринимаемом объекте, выделяются и интерпретируются его части. Затем предпринимается попытка "собрать" из них целое, чтобы проверить правильность исходной гипотезы. Разумеется, воспринимаемый объект может интерпретироваться в рамках более крупного целого.

Так, читая предложение, человек узнает буквы, воспринимает слова, связывает их в синтаксические конструкции и понимает смысл.

В технических системах любое решение при распознавании текста принимается неоднозначно, а путем последовательного выдвижения и проверки гипотез и привлечения как знаний о самом исследуемом объекте, так и общего контекста. Целостное описание класса объектов восприятия отвечает двум условиям: во - первых, все объекты данного класса удовлетворяют этому описанию, а во- вторых, ни один объект другого класса не удовлетворяют ему. Например, класс изображений буквы "К" должен быть описан так, чтобы любое изображение буквы "К" в него попадало, а изображение всех других букв - нет. Такое описание обладает свойством отображаемости, то есть обеспечивает воспроизведение описываемых объектов: эталон буквы для системы OCR позволяет визуально воспроизвести букву, эталон слова для распознавания речи позволяет произнести слово, а описание структуры предложения в синтаксическом анализаторе позволяет синтезировать правильное предложение. С практической точки зрения отображаемость играет огромную роль, поскольку позволяет эффективно контролировать качество описаний.

Существует два вида целостного описания: шаблонное и структурное.

В первом случае описание представляет собой изображение в растровом или векторном представлении, и задан класс преобразований (например, повтор, масштабирование и пр.).

Во втором случае описание представляется в виде графа, узлами которого являются составляющие элементы входного объекта, а дугами - пространственные отношения между ними. В свою очередь элементы могут оказаться сложными (то есть иметь свое описание).

Конечно, шаблонное описание проще в реализации, чем структурное. Однако оно не может использоваться для описания объектов с высокой степенью изменчивости. Шаблонное описание, к примеру, может

приниматься для распознавания только печатных символов, а структурное - еще и для рукописных.

Целостность восприятия предлагает два важных архитектурных решения (рисунок 4.16). Во-первых, все источники знания должны работать по возможности одновременно. Нельзя, например, сначала распознать страницу, а затем подвергнуть ее словарной и контекстной обработке, поскольку в этом случае невозможно будет осуществить обратную связь от контекстной обработки к распознаванию. Во вторых, исследуемый объект должен представляться и обрабатываться по возможности целиком.

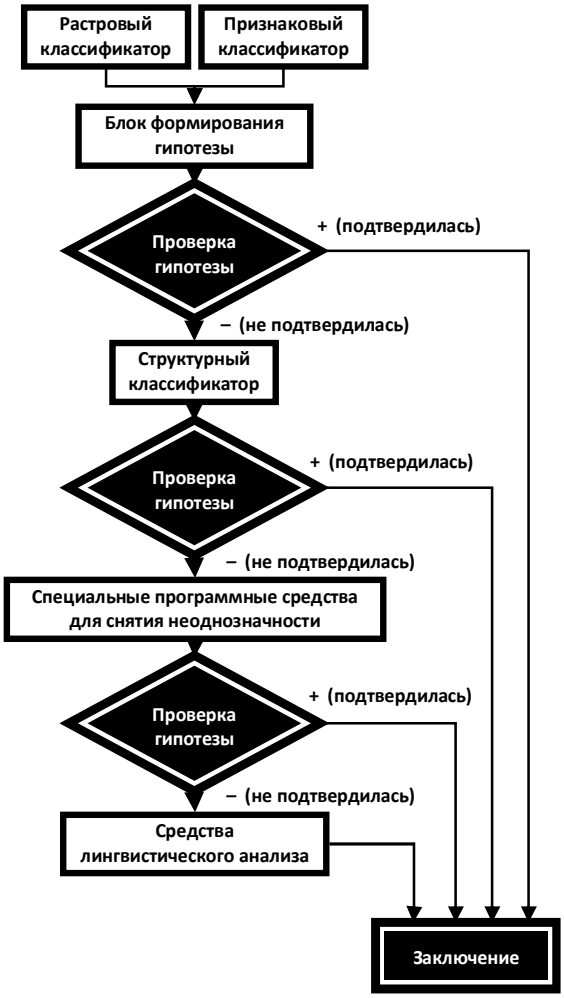

Рисунок 4.16 – Укрупненная схема работы системы распознавания образов

Первый шаг восприятие – это формирование гипотезы о воспринимаемом объекте. Гипотеза может формироваться как на основе априорной модели объекта, контекста и результатов проверки предыдущих гипотез (процесс "сверху - вниз"), так и на основе предварительного анализа объекта ("снизу - вверх"). Второй шаг - уточнение восприятия (проверка гипотезы), при котором производится дополнительный анализ объекта в рамках выдвинутой гипотезы и в полную силу привлекается контекст.

Для удобства восприятия необходимо провести предварительную обработку объекта, не потеряв при этом существенной информации о нем. Обычно предварительная обработка сводится к преобразованию входного объекта в представление, удобное для дальнейшей работы (например, векторизация изображения), или получение всевозможных вариантов сегментации входного объекта, из которого путем выдвижения и проверки гипотез выбирается правильный. Процесс выдвижения и проверки гипотез должен быть явно отражен в архитектуре программы. Каждая гипотеза должна быть объектом, который можно было бы оценить или сравнить с другими. Поэтому обычно гипотезы выдвигаются последовательно, а затем объединяются в список и сортируются на основе предварительной оценке. Для окончательного же выбора гипотезы активно используется контекст и другие дополнительные источники знаний.

Ныне одним из лидеров в области генетического программирования является группа исследователей из Стэндфордского университета (Stanford University), работающая под руководством профессора Джона Коза. Генетическое программирование вдохнуло новую жизнь в хорошенько уже подзабытый язык LISP (List Processing), который создавался группой Джона Маккарти (того самого, кто в 60-е годы ввел в наш обиход термин "искусственный интеллект") как раз для обработки списков и функционального программирования. Кстати, именно этот язык в США

был и остается одним из наиболее распространенных языков программирования для задач искусственного интеллекта.

4.5.2. Распознавание символов

Оптическое распознавание символов (англ. optical character recognition, OCR) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные — последовательность кодов, использующихся для представления символов в компьютере (например, в текстовом редакторе). Распознавание широко используется для конвертации книг и документов в электронный вид, для автоматизации систем учёта в бизнесе или для публикации текста на веб-странице. Оптическое распознавание текста позволяет редактировать текст, осуществлять поиск слова или фразы, хранить его в более компактной форме, демонстрировать или распечатывать материал, не теряя качества, анализировать информацию, а также применять к тексту электронный перевод, форматирование или преобразование в речь. Оптическое распознавание текста является исследуемой проблемой в областях распознавания образов, искусственного интеллекта и компьютерного зрения.

Системы оптического распознавания текста (таблица 4.2) требуют калибровки для работы с конкретным шрифтом; в ранних версиях для программирования было необходимо изображение каждого символа, программа одновременно могла работать только с одним шрифтом.

Таблица 4.2 – Системы распознавания символов

Название	**Операционные системы**	**Заметки**
ExperVision TypeReader & RTK	Windows, Mac OS X, Unix, Linux, OS/2	Получала высокие оценки в начале 1990-х.
ABBYY FineReader	Windows; Linux,	Для работы с различными языками

Название	Операционные системы	Заметки
	Mac OS (не для конечного потребителя)	требуется поддержка соответствующего языка.
OmniPage	Windows, Mac OS	Производство Nuance Communications
Readiris	Windows, Mac OS	Производство бельгийской I.R.I.S. Group. Содержит региональные пакеты для распознавания азиатских языков и языков среднего востока.
Persian Reader	Windows	Специализируется на персидском языке (фарси).
Kirtas Technologies Arabic OCR	Windows	Может распознавать арабские и английские символы на одной странице.
Zonal OCR	Windows	Zonal OCR помогает автоматизировать извлечение данных из компьютерных изображений.
ViewWise компании Computhink	Windows	Система управления документами
CuneiForm	Windows (с GUI), Linux, Mac OS, FreeBSD (CLI)	Промышленная многоязычная система, умеет сохранять форматирование текста и распознаёт запутанные таблицы произвольной структуры
GOCR	Кросс-платформенная	В начальной стадии разработки
Microsoft Office Document Imaging	Windows, Mac OS X	
Microsoft Office OneNote 2007	Windows	
NovoDynamics NovoVerus	Windows	Специализируется на языках среднего востока
Ocrad	Unix-like, OS/2	
Brainware	Windows	Извлечение данных из документов и их обработка — например, счета,

Название	Операционные системы	Заметки
		извещения, накладные и платёжки
HOCR	Linux	Распознавание текстов на иврите
OCRopus	Linux	Расширяемая система распознавания, которая может использовать Tesseract
ReadSoft	Windows	Сканирование, распознавание и классификация деловых бумаг, например, договоров, счетов и платёжных поручений.
RelayFax Network Fax Manager компании Alt-N Technologies	Windows	Многоязычная система, используется для преобразования факсов в доступные для правки форматы документов (doc, pdf и т. д.).
Scantron Cognition	Windows	Для работы с различными языками требуется поддержка соответствующего языка.
SILVERCODERS OCR Server	Linux	Серверная многоязычная система, имеет высокое качество распознавания, может сохранять форматирование текста и распознаёт запутанные таблицы произвольной структуры
SimpleOCR	Windows	
SmartScore	Windows, Mac OS	Для распознавания нотной записи
Tesseract	Windows, Mac OS X, Linux, OS/2	Разрабатывается компанией Google
WeOCR	Интерфейс: Браузер; Сервер: POSIX, Unix	Платформа для браузерных систем распознавания символов. Страница проекта: WeOCR
FreeOCR	Интерфейс: Браузер; Сервер: POSIX, Unix	Платформа для браузерных систем распознавания символов. Использует Tesseract. Большое количество поддерживаемых языков. Страница проекта: FreeOCR
img2txt	Интерфейс:	Online OCR-сервис, позволяет

Название	Операционные системы	Заметки
	Браузер; Сервер: POSIX, Linux	распознать многоязычный текст из сканированного документа или фотографии. Конвертирует результат в редактируемые форматы (RTF, TXT, HTML).
FineReaderOnline.ru	Интерфейс: Браузер	Online OCR-сервис, позволяющий распознать многоязычный текст из отсканированного документа или фотографии. Конвертирует результат в редактируемые форматы (PDF, PDF/A, DOC, RTF, XLS, TXT). На данный момент до 10 страниц в день можно распознавать бесплатно.
OnlineOCR.ru	Интерфейс: Браузер	Online OCR-сервис, позволяет распознать многоязычный текст из сканированного документа или фотографии. Конвертирует результат в редактируемые форматы (PDF, DOC, XLS, TXT, HTML)
NewOCR.com	Интерфейс: Браузер	Online OCR-сервис, позволяет распознать многоязычный текст из сканированного документа или фотографии. Поддерживает 29 языков (болгарский, каталанский, чешский, датский, голландский, английский, финский, французский, немецкий, греческий, венгерский, индонезийский, итальянский, латышский, литовский, норвежский, польский, португальский, румынский, русский, сербский, словацкий, словенский, испанский, шведский, тагалог, турецкий, украинский, вьетнамский) и распознает текст, отформатированный в несколько колонок.

Название	Операционные системы	Заметки
RasterID	Windows	Программа, позволяющая сканировать и обрабатывать растровые изображения, а также автоматизировать регистрацию отсканированных изображений в электронном архиве или системе документооборота. Позволяет распознавать данные из штампа — основной надписи чертежей — и заносить их в базу данных. Страница разработчика: CSoft
LiveOCR	Интерфейс: Браузер	Сервис оптического распознавания символов, позволяющий файлы форматов JPG, JPEG, BMP, PNG, GIF, содержащих текст, конвертировать в файлы формата RTF.
NSOCR	Windows	Современная, развивающаяся система оптического распознавания текста. Разрабатывается компанией Nicomsoft

В настоящее время больше всего распространены так называемые «интеллектуальные» системы, с высокой степенью точности распознающие большинство шрифтов (рисунок 4.17).

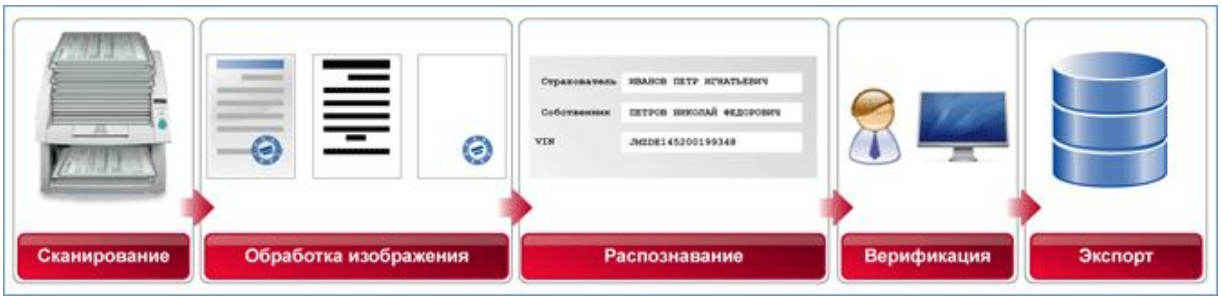

Рисунок 4.17 – Шесть основных стадий обработки документа в системе распознавания

Некоторые системы оптического распознавания текста способны восстанавливать исходное форматирование текста, включая изображения, колонки и другие нетекстовые компоненты.

Сегодня известно три подхода к распознаванию символов - шаблонный, структурный и признаковый. Но принципу целостности отвечает лишь первые два.

Шаблонное описание проще в реализации, однако, в отличие от структурного, оно не позволяет описывать сложные объекты с большим разнообразием форм. Именно поэтому шаблонное описание применяется для распознавания лишь печатных символов, в то время как структурное - для рукописных, имеющих, естественно, гораздо больше вариантов начертания.

Шаблонные системы

Такие системы преобразуют изображение отдельного символа в растровое, сравнивают его со всеми шаблонами, имеющимися в базе и выбирают шаблон с наименьшим количеством точек, отличных от входного изображения. Шаблонные системы довольно устойчивы к дефектам изображения и имеют высокую скорость обработки входных донных, но надежно распознают только те шрифты, шаблоны которых им "известны". И если распознаваемый шрифт хоть немного отличается от эталонного, шаблонные системы могут делать ошибки даже при обработке очень качественных изображений!

Структурные системы

В таких системах объект описывается как граф, узлами которого являются элементы входного объекта, а дугами - пространственные отношения между ними. Система реализующие подобный подход, обычно работают с векторными изображениями. Структурными элементами являются составляющие символ линии. Так, для буквы "р" - это вертикальный отрезок и дуга.

К недостаткам структурных систем следует отнести их высокую чувствительность к дефектам изображения, нарушающим составляющие элементы. Также векторизация может добавить дополнительные дефекты. Кроме того, для этих систем, в отличие от шаблонных и признаковых, до сих пор не созданы эффективные автоматизированные процедуры обучения. Поэтому для Fine Reader структурные описания пришлось создать в ручную.

Признаковые системы

В них усредненное изображение каждого символа представляется как объект в n-мерном пространстве признаков. Здесь выбирается алфавит признаков, значения которых вычисляются при распознавании входного изображения. Полученный n-мерный вектор сравнивается с эталонными, и изображение относится к наиболее подходящему из них. Признаковые системы не отвечают принципу целостности. Необходимое, но недостаточное условие целостности описания класса объектов (в нашем случае это класс изображений, представляющих один символ)состоит в том, что описанию должны удовлетворять все объекты данного класса и ни один из объектов других классов. Но по-скольку при вычислении признаков теряется существенная часть информации, трудно гарантировать, что к данному классу удастся отнести только <родные> объекты.

Структурно-пятенный эталон

В ABBYY FineReader применяется так называемый структурно-пятенный эталон (рисунок 4.18) и его фонтанное (от англ. font – шрифт) представление, которое имеет вид набора пятен с попарными отношениями между ними. Данная разработка фирмы ABBYY первоначально использовалась для распознавания рукописного текста, а затем была успешно применена и для обработки печатных символов. При этом обеспечиваются все достоинства шаблонного и структурного

классификаторов. Также данное представление нечувствительно к различным начертаниям и дефектам символов.

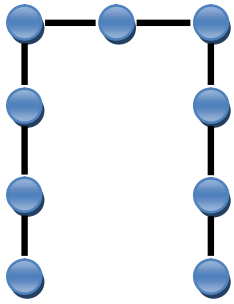

Рисунок 4.18 - Структурно-пятенный эталон

Фонтанное преобразование совмещает в себе достоинства шаблонной и структурной систем и, по нашему мнению, позволяет избежать недостатков, присущих каждой из них по отдельности. В основе этой технологии лежит использование структурно-пятенного эталона. Он позволяет представить изображения в виде набора пятен, связанных между собой n-арными отношениями, задающими структуру символа. Эти отношения (то есть расположение пятен друг относительно друга) образуют структурные элементы, составляющие символ. Так, например, отрезок - это один тип n-арных отношений между пятнами, эллипс - другой, дуга - третий. Другие отношения задают пространственное расположение образующих символ элементов.

В эталоне задаются:

- имя;
- обязательные, запрещающие и необязательные структурные элементы;
- отношения между структурными элементами;
- отношения, связывающие структурные элементы с описывающим прямоугольником символа;
- атрибуты, используемые для выделения структурных элементов;

– атрибуты, используемые для проверки отношений между элементами;
– атрибуты, используемые для оценки качества элементов и отношений;
– позиция, с которой начинается выделение элемента (отношения локализации элементов).

Структурные элементы, выделяемые для класса изображений, могут быть исходными и составными. Исходные структурные элементы - это пятна, составные - отрезок, дуга, кольцо, точка. В качестве составных структурных элементов, в принципе, могут быть взяты любые объекты, описанные в эталоне. Кроме того, они могут быть описаны как через исходные, так и через другие составные структурные элементы.

Например, для распознавания корейских иероглифов (слоговое письмо) составными элементами для описания слога являются описания отдельных букв (но не отдельные элементы букв). В итоге, использование составных структурных элементов позволяет строить иерархические описания классов распознаваемых объектов.

В качестве отношений используются связи между структурными элементами, которые определяются либо метрическими характеристиками этих элементов (например, <длина больше>), либо их взаимным расположением на изображении (например, <правее>, <соприкасается>).

При задании структурных элементов и отношений используются конкретизирующие параметры, позволяющие доопределить структурный элемент или отношение при использовании этого элемента в эталоне конкретного класса. Для структурных элементов конкретизирующими могут являться, например, параметры, задающие диапазон допустимой ориентации отрезка, а для отношений - параметры, задающие предельное допустимое расстояние между характерными точками структурных элементов в отношении <соприкасается>.

Конкретизирующие параметры используются также для вычисления качества конкретного структурного элемента изображения и качества выполнения данного отношения.

Построение и тестирование структурно-пятенных эталонов для классов распознаваемых объектов - процесс сложный и трудоемкий. База изображений, которая используется для отладки описаний, должна содержать примеры хороших и плохих (предельно допустимых) изображений для каждой графемы, а изображения базы разделяются на обучающее и контрольное множества.

Разработчик описания предварительно задает набор структурных элементов (разбиение на пятна) и отношения между ними. Система обучения по базе изображений автоматически вычисляет параметры элементов и отношений. Полученный эталон проверяется и корректируется по контрольной выборке изображений данной графемы. По контрольной же выборке проверяется результат распознавания, то есть оценивается качество подтверждения гипотез.

Распознавание с использованием структурно-пятенного эталона происходит следующим образом. Эталон накладывается на изображение, и отношения между выделенными на изображении пятнами сравниваются с отношениями пятен в эталоне. Если выделенные на изображении пятна и отношения между ними удовлетворяют эталону некоторого символа, то данный символ добавляется в список гипотез о результате распознавания входного изображения.

4.5.3. Распознавание рукописных текстов

Проблема распознавания рукописного текста значительно сложнее, чем в случае с текстом печатным. Если в последнем случае мы имеем дело с ограниченным числом вариаций изображений шрифтов (шаблонов), то в случае рукописного текста число шаблонов неизмеримо больше.

Дополнительные сложности вносят также иные соотношения линейных размеров элементов изображений и т.п.

И все же сегодня мы можем признать, что основные этапы разработки технологии распознавания рукописных (отдельные символы, написанные от руки) символов уже пройдены. В настоящее время существуют технологии распознавания всех основных типов текстов: стилизованных цифр, печатных символов и рукопечатных символов. Но технологии ввода рукопечатных символов потребуется еще пройти стадию адаптации, после чего можно будет заявить, что инструментарий для потокового ввода документов в архивы действительно реализован полностью.

Динамичное развитие новых компьютерных технологий (сетевые технологии, технологии клиент-сервер, и т. д.) нашли свое отражение и в состоянии сектора электронного документооборота. Если раньше в продвижении технологий бесклавиатурного ввода делался упор на преимущества их персонального использования, то сегодня на первый план выходят преимущества коллективного и рационального использования технологий ввода и обработки документов. Иметь одну, обособленную систему распознавания сегодня уже явно недостаточно. С распознанными текстовыми файлами (как бы хорошо они распознаны ни были) нужно что-то делать: хранить в базе данных, осуществлять их поиск, передавать по локальной сети, и т. д. Словом, требуется взаимодействие с архивной или иной системой работы с документами. Таким образом, система распознавания превращается в утилиту для архивных и иных систем работы с документами.

ЗАКЛЮЧЕНИЕ

Создание систем искусственного интеллекта, сопряжено с решением множества проблем. На пути создания ИИ это и ограниченность ресурсов, и недостаточные знания в этой области, а также проблема осуществимости это сделать, и многие другие технические проблемы.

Искусственный интеллект в настоящее время остается научным направлением, связанным с компьютерным моделированием человеческих интеллектуальных функций.

Системы искусственного интеллекта обычно используются для обозначения способности вычислительной системы выполнять задачи, свойственные интеллекту человека, например, задачи логического вывода и обучения.

Любая задача, алгоритм решения которой заранее не известен или же данные неполные может быть отнесена к задачам области ИИ.

Системы, программы, выполняющие действия по решению задачи можно отнести к ИИ, если результат их деятельности аналогичен результату человека при решении той же задачи. Поэтому к ИИ можно отнести целый ряд программных средств: системы распознавания текста, автоматизированного проектирования, самообучающиеся программы и др. Но не только поэтому, а еще и потому, что они работают по сходным принципам с человеком.

Есть два основных перспективных направления в исследовании ИИ. Первое заключается в приближении систем ИИ к принципам человеческого мышления. Второе заключается в создании ИИ, представляющего интеграцию уже созданных систем ИИ в единую систему, способную решать проблемы человечества.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Суркова Н.Е. Методы проектирования информационных систем / Н.Е. Суркова, А.В. Остроух – М.: РосНОУ, 2004. – 144 с. – ISBN 5-89789-021-8.
2. Остроух А.В. Основы построения систем искусственного интеллекта для промышленных и строительных предприятий: монография / А.В. Остроух. – М.: ООО «Техполиграфцентр», 2008. – 280 с. – ISBN 978-5-94385-033-2.
3. Остроух А.В. Автоматизация транспортировки продукции / А.В. Остроух, Н.Г. Куфтинова – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2011. – 146 p. – ISBN 978-3-8454-1089-0.
4. Остроух А.В. Рефакторинг баз данных. Автоматизация технологических процессов рефакторинга баз данных промышленных предприятий / А.В. Остроух, Д.А. Пшеничный, О.Б. Рогова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 133 p. – ISBN 978-3-659-38753-1.
5. Остроух А.В. Автоматизация управления производством. Повышение эффективности автоматизированных аналитических систем предприятий автомобильной промышленности / А.В. Остроух, Э.А. Чернов, Д.Т. Нгуен. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 285 p. – ISBN 978-3-659-34762-7.
6. Остроух А.В. Системы планирования перевозок. Программно-технологические решения по разработке системы планирования заданий для заказных пассажирских перевозок / А.В. Остроух, А.Б. Львова, А.Р. Исмаилов. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 121 p. – ISBN 978-3-659-43619-2.

7. Остроух А.В. Интеллектуальные системы в науке и производстве / А.В. Остроух, А.Б. Николаев. – Saarbrucken, Germany: Palmarium Academic Publishing, 2012. – 312 p. – ISBN 978-3-659-98006-0.
8. Остроух А.В. Мультимедиа-технологии / А.В. Остроух, А.М. Васьковский, А.Б. Николаев. – Saarbrucken, Germany: Palmarium Academic Publishing, 2012. – 228 p. – ISBN 978-3-659-98030-5.
9. Остроух А.В. Ввод и обработка цифровой информации: учебник для нач. проф. образования / А.В. Остроух. – М.: Издательский центр «Академия», 2012. – 288 с. – ISBN 978-5-7695-9457-1.
10. Остроух А.В. Оперативный контроль транспортно-экспедиционной деятельности. Процессный подход к агрегированию системы показателей деятельности транспортно-экспедиционного предприятия / А.В. Остроух, А.М. Ивахненко, Н.А. Крупенский. – Saarbrucken, Germany: Palmarium Academic Publishing, 2013. – 221 p. – ISBN 978-3-659-98329-0.
11. Исмоилов М.И. Подготовка и переподготовка персонала предприятий промышленного и транспортного комплексов с применением мобильных технологий: монография / М.И. Исмоилов, А.Б. Николаев, А.В. Остроух. – Saint-Louis, MO, USA: Publishing House Science and Innovation Center, 2013. – 166 с. – ISBN 978-0-615-67111-6.
12. Николаев А.Б. Информационные технологии в менеджменте и транспортной логистике: учебное пособие / А.Б. Николаев, А.В. Остроух. – Saint-Louis, MO, USA: Publishing House Science and Innovation Center, 2013. – 254 с. – ISBN 978-0-615-67110-9.
13. Остроух А.В. Системы искусственного интеллекта в промышленности, робототехнике и транспортном комплексе: монография / А.В. Остроух – Красноярск: Научно-инновационный центр, 2013. – 326 с. – ISBN 978-5-906314-10-9.

14. Остроух А.В. Основы информационных технологий: учебник для сред. проф. образования / А.В. Остроух. – М.: Издательский центр «Академия», 2014. – 208 с. – ISBN 978-5-4468-0588-4.
15. Суркова Н.Е. Методология структурного проектирования информационных систем: монография / Н.Е. Суркова, А.В. Остроух. – Красноярск: Научно-инновационный центр, 2014. – 190 с. – ISBN 978-5-906314-16-1.
16. Покровский А.К. Системный анализ и компьютерные технологии при управлении транспортным обслуживанием предприятий: монография / А.К. Покровский, А.В. Остроух, А.М. Ивахненко. – М.: Известия, 2014. – 261 с. – ISBN 978-5-94280-314-7.
17. Демидов Д.Г. Программные и аппаратные средства систем мультимедиа. Часть 1. Аппаратные средства: учебное пособие / Д.Г. Демидов, А.М. Васьковский, А.Б. Николаев, А.В. Остроух, П.И. Лукащук, В.А. Виноградов. – М.: МГУП имени Ивана Федорова, 2014. – 78 с.
18. Демидов Д.Г. Программные и аппаратные средства систем мультимедиа. Часть 2. Программные средства: учебное пособие / Д.Г. Демидов, А.М. Васьковский, А.Б. Николаев, А.В. Остроух, П.И. Лукащук, В.А. Виноградов. – М.: МГУП имени Ивана Федорова, 2014. – 70 с.
19. Остроух А.В. Проектирование системы распределенных баз данных / А.В. Остроух, А.В. Помазанов. – Saarbrucken, Germany: Palmarium Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-60041-8.
20. Остроух А.В. Автоматизация управления автотранспортными предприятиями. Новый подход на основе интеллектуальных мультиагентных систем / А.В. Остроух, А.В. Воробьева, Н.Е. Суркова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-47576-4.

21. Суркова Н.Е. Профессиональные информационные системы и базы данных: методические указания к лабораторным работам / Н.Е. Суркова, А.В. Остроух, Т.И. Еремина. – Красноярск: Научно-инновационный центр, 2015. – 49 с. – ISBN 978-5-906314-23-9. DOI: 10.12731/asu.madi.ru/PISDB.2015.49.
22. Остроух А.В. Об одном походе к построению автоматизированной навигационной системы мониторинга и диспетчерского контроля и учета работы транспорта нефтедобывающих предприятий / А.В. Остроух, Д.Б. Ефименко, В.Н. Жучков // Инженер-нефтяник. – 2011. – №2. – С. 43-47.
23. Остроух А.В. Разработка информационно-аналитической системы мониторинга технологических процессов предприятия автомобильной промышленности / А.В. Остроух, Ю. Тянь // В мире научных открытий. – Красноярск: «Научно-инновационный центр», 2013. – № 8.2 (44). – С. 191-205.
24. Остроух А.В. Интеграция компонентов системы мониторинга / А.В. Остроух, Юань Тянь // Молодой ученый. – 2013. – №10. – С. 182-185.
25. Остроух А.В. Исследование начального периода моделирования на точность среднеинтегральной оценки имитационных моделей / А.В. Остроух, А.А. Солнцев, Н.В. Солдатов, К.А. Новицкий, П.С. Якунин // Вестник МАДИ. – 2010. – Вып. 2(21). – С. 61-65.
26. Остроух А.В. Математическая модель связей в системе диагностики электрооборудования автомобилей / А.В. Остроух, А.А. Солнцев, О.Ф. Калухов, Г.Г. Ягудаев // Вестник МАДИ. – 2010. – Вып. 2(21). – С. 66-70.
27. Антонов П.Д. User Is A Great Obstacle For Security Systems / П.Д. Антонов, А.В. Остроух // Молодой ученый. – 2011. – №4. Т.3. – С. 62-63.

28. Краснянский М.Н., Карпушкин С.В., Обухов А.Д., Молоткова Н.В., Галыгина И.В., Остроух А.В. Структура системы электронного документооборота для управления научно-образовательной деятельностью высшего учебного заведения // Промышленные АСУ и контроллеры. – 2014. – №8. – С. 23-31.
29. M.N. Krasnyanskiy, A.V. Ostroukh, S.V. Karpushkin, A.D. Obukhov. Information Technology for the Development of Automated Control System for University Projects // Математические методы в технике и технологиях – ММТТ-27: сб. трудов XXVII Междунар. науч. конф.: в 12 т. Т.4. Секции 10, 11 / под общ. ред. А.А. Большакова. – Тамбов: Тамбовск. гос. техн. ун-т, 2014. – С. 42-45. – ISBN 978-5-7433-2386-9.
30. Краснянский М.Н., Остроух А.В., Карпушкин С.В., Обухов А.Д. Разработка комплексной системы управления научно - инновационной деятельностью: взаимодействие базы данных изделий и архива документации // Приборы и системы. Управление, контроль, диагностика. – 2014. – №12. – С. 19-23.
31. Сальный А.Г., Остроух А.В. Исследование производительности файловых систем ядра Linux // Автоматизация и управление в технических системах. – 2014. – №4 (12). – С. 158-167. DOI: 10.12731/2306-1561-2014-4-16.
32. Обухов П.А., Николаев А.Б., Остроух А.В. Исследование эффективности работы сетевых серверов в среде имитационного моделирования AnyLogic // Международный журнал экспериментального образования. – 2015. – № 3-3. – С. 338-342.
33. Пастухов Д.А., Юрчик П.Ф., Остроух А.В. Сравнительный анализ гипервизоров // Международный журнал экспериментального образования. – 2015. – № 3-3. – С. 346-350.

34. Сальный А.Г., Остроух А.В. Исследование эффективности структур хранения данных ядра Linux // Международный журнал экспериментального образования. – 2015. – № 3 – С. 366-367.
35. Чувиков Д.А., Феоктистов В.П., Остроух А.В. Исследование 3D форматов хранения данных в интеллектуальных системах виртуальной реальности // Международный журнал экспериментального образования. – 2015. – № 3-3. – С. 416-420.
36. Терентьев Д.И., Николаев А.Б., Остроух А.В. Исследование дисковых массивов RAID по параметрам надежности и быстродействия // Международный журнал экспериментального образования. – 2015. – № 3-3. – С. 423-427.
37. Круглов А.М. Актуализация сведений о данных информационной системы средствами активного словаря-справочника данных / А.М. Круглов, А.В. Будихин, Д.А. Буров, А.В. Остроух // Научный вестник МГТУ ГА. Серия Аэромеханика и прочность. – 2007. – №119 (9). – С. 166-171.
38. Николаев А.В. Принципы организации динамических интерфейсов доступа к данным с использованием словарей-справочников данных / А.В. Николаев, А.В. Будихин, Д.А. Буров, А.В. Остроух // Научный вестник МГТУ ГА. Серия Аэромеханика и прочность. – 2007. – №119 (9). – С. 172-178.
39. Остроух А.В., Баринов А.П., Николаев А.В., Будихин С.А. Использование словаря-справочника данных для реализации пользовательских средств обработки информации // Приборы и системы. Управление, контроль, диагностика. – 2008. – №3. – С. 13-17.
40. Пшеничный Д.А. Анализ параметров и сравнение СУБД для реализации информационного обеспечения промышленного

предприятия / Д.А. Пшеничный, А.В. Будихин, А.В. Остроух // Промышленные АСУ и контроллеры. – 2010. – №12. – С. 7-11.

41. Помазанов А.В. Методика оптимизации баз данных / А.В. Помазанов, А.И. Белоусова, А.О. Васильева, А.В. Остроух // В мире научных открытий. Серия «Проблемы науки и образования». – 2012. – №12. – С.49-54.

42. Ostroukh A.V., Krasnyanskiy M.N., Karpushkin S.V., Obukhov A.D. Development of Automated Control System for University Research Projects // Middle East Journal of Scientific Research. 2014. Vol. 20 (12). pp. 1780-1784. DOI: 10.5829/idosi.mejsr.2014.20.12.21091.

43. Ostroukh A., Pomazanov A. Realtime Development and Testing of Distributed Data Processing System for Industrial Company // Middle East Journal of Scientific Research. 2014. Vol. 20 (12). pp. 2184-2193. DOI: 10.5829/idosi.mejsr.2014.20.12.21106.

44. Ostroukh A.V., Belousova A.I., Pavlov D.A., Yurchik P.F. Problems of organization and search the knowledge base in the CRM-systems // IOSR Journal of Engineering (IOSRJEN). 2014. Vol. 04. Issue 02. V3. pp. 18-23. DOI: 10.9790/3021-04231823. ANED: 0.4/3021-04231823.

45. Krasnyanskiy M.N., Karpushkin S.V., Obukhov A.D., Ostroukh A.V. Automated control system for university research projects // International Journal of Advanced Studies (iJAS). 2014. Vol. 4, Issue 1, pp. 22-26. DOI: 10.12731/2227-930X-2014-1-4.

46. Mikhail Nikolaevich Krasnyanskiy, Andrey Vladimirovich Ostroukh, Sergey Viktorovich Karpushkin, Artyom Dmitrievich Obukhov, Nataliya Vyacheslavovna Molotkova and Irnina Vladimirovna Galygina. Electronic Document Management Systems Structure for University Research and Education // Journal of Engineering and Applied Sciences. 2014. Vol 9. Issue 5. pp. 182-189. DOI: 10.3923/jeasci.2014.182.189.

47. Помазанов А.В., Остроух А.В. Создание и тестирование распределённой системы работы с удалёнными узлами // Автоматизация и современные технологии. – 2014. – №7. – С. 17-23.
48. Помазанов А.В., Остроух А.В. Новый подход к разработке прототипа распределенной системы баз данных промышленного предприятия // Промышленные АСУ и контроллеры. – 2014. – №9. – С. 11-20.
49. Суркова Н.Е., Остроух А.В. Методика обучения технологии баз данных для студентов непрофильных направлений подготовки в технических ВУЗах // ИНЖЕНЕРНАЯ ПЕДАГОГИКА. – М: МАДИ, 2015. – Вып. 17. Т.3. – С. 146-156.
50. Суркова Н.Е., Остроух А.В., Еремина Т.И. ЭЛЕКТРОННОЕ УЧЕБНОЕ ИЗДАНИЕ «ПРОФЕССИОНАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ: МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ» // Хроники объединенного фонда электронных ресурсов Наука и образование. 2015. № 3 (70). С. 10.
51. Krasnyanskiy M.N., Karpushkin S.V., Obukhov A.D., Ostroukh A.V. Model of Documents Management for Academic and Research Universities on Basis Set Theory// American-Eurasian Journal of Agricultural & Environmental Sciences. 2015. Vol. 15. No. 5. pp. 824-831, DOI: 10.5829/idosi.aejaes.2015.15.5.12639.
52. Краснянский М.Н., Карпушкин С.В., Обухов А.Д., Остроух А.В. Математическая модель электронного документооборота на основе теории множеств // Промышленные АСУ и контроллеры. – 2015. – №5. – С. 52-58.
53. Краснянский М.Н., Обухов А.Д., Карпушкин С.В., Остроух А.В. Разработка информационной системы электронного документооборота управления фундаментальных и прикладных

исследований // Вестник Тамбовского государственного технического университета. 2015. Т. 21. № 2. С. 216-230.

54. Ostroukh A.V., Mezencev K.N., Krasnyanskiy M.N., Punam J., Surkova N.E. SIMULATED OPERATION OF THE COMPUTER NETWORK // SGEM2015 Conference Proceedings, ISBN 978-619-7105-34-6 / ISSN 1314-2704, June 18-24, 2015, Book2 Vol. 1, 299-306 pp. DOI: 10.5593/SGEM2015/B21/S7.037.

55. Ostroukh A.V., Krasnyanskiy M.N., Obukhov A.D., Karpov S.V., Dedov D.L. DEVELOPMENT OF AUTOMATED INFORMATION SYSTEMS FOR MONITORING OF INTELLECTUAL ACTIVITY RESULTS // SGEM2015 Conference Proceedings, ISBN 978-619-7105-34-6 / ISSN 1314-2704, June 18-24, 2015, Book2 Vol. 1, 101-108 pp. DOI: 10.5593/SGEM2015/B21/S7.014.

56. Баринов К.А. Опыт разработки и использования ролевых игр для подготовки и переподготовки специалистов предприятий промышленности и транспортного комплекса / К.А. Баринов, Д.А. Буров, А.В. Бугаев, А.В. Остроух // Научный вестник МГТУ ГА. – 2009. – №141. – С. 189-197.

57. Николаев А.Б., Остроух А.В., Краснянский М.Н. Разработка информационно-образовательной системы для повышения квалификации персонала предприятий химической промышленности и транспортного комплекса // ЭЛЕКТРОННОЕ ОБУЧЕНИЕ И ДИСТАНЦИОННЫЕ ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ. – 2013. – № 1; URL: eodot.esrae.ru/1-10 (дата обращения: 11.12.2013).

58. Лукащук П.И. Адаптивная методика прогнозирования пассажиропотоков в АСУ пассажирского автотранспортного предприятия / П.И. Лукащук, С. Бенгедаш, А.Г. Николаев, А.В. Остроух // Приборы и системы. Управление, контроль, диагностика. – 2006. – №11. – С. 7-11.

59. Кузнецов И.А. Особенности реализации автоматизированной информационно-аналитической системы центра планирования перевозок строительных грузов / И.А. Кузнецов, А.В. Остроух // Вестник МАДИ(ГТУ). – М.: МАДИ (ГТУ), 2008. – Вып. 1(12). – С. 92-96.
60. Куфтинова Н.Г. Процессно-ориентированный подход к автоматизации планирования и управления транспортировкой продукции предприятий промышленности / Н.Г. Куфтинова, А.В. Остроух // Вестник МАДИ – 2010. – Вып. 4(23). – С. 62-66.
61. Остроух А.В., Куфтинова Н.Г. Имитационное моделирование управления транспортными потоками в мегаполисе // Автотранспортное предприятие. – 2010. – №12. – С. 41-42.
62. Остроух А.В., Алтунина А.В. Система автоматизации и контроля за продажами и производством автомобилей // Автотранспортное предприятие. – 2011. – №3. – С. 41-43.
63. Польгун М.Б. Анализ моделей оперативного диспетчерского управления городским пассажирским транспортом / М.Б. Польгун, А.В. Воробьева, А.В. Остроух // Молодой ученый. – 2011. – №4. Т.3. – С. 9-13.
64. Ефименко Д.Б. Автоматизированная навигационная система диспетчерского контроля и учета работы транспорта нефтедобывающих предприятий / А.И. Губанов, Д.Б. Ефименко, А.Б. Николаев, А.В. Остроух // Молодой ученый. – 2011. – №4. Т.3. – С. 18-21.
65. Соломатин А.В. Об одном подходе к решению задачи регулирования дорожного движения: автоматизация системы управления светофорами / А.В. Остроух, А.В. Соломатин, О.О. Варламов // Автотранспортное предприятие. – 2011. – №6. – С. 43-45.

66. Ефименко Д.Б., Остроух А.В., Филатов С.А. Развитие навигационной системы диспетчерского управления грузовым транспортом (на примере нефтедобывающих предприятий) // Автотранспортное предприятие. – 2011. – №11. – С. 32-34.
67. Ефименко Д.Б. Концепция автоматизированной навигационной системы диспетчерского контроля и учета работы транспорта нефтедобывающих и нефтеперерабатывающих предприятий / Д.Б. Ефименко, А.В. Остроух, А.Б. Николаев, А.И. Губанов // Автоматизация, телемеханизация и связь в нефтяной промышленности. – 2011. – №11. – С.12-14.
68. Остроух А.В. Научный подход к разработке автоматизированной навигационной системы диспетчерского контроля и учета работы транспорта нефтедобывающих и нефтеперерабатывающих предприятий / А.В. Остроух, А.Б. Николаев, Д.Б. Ефименко, С.В. Жанказиев // Автоматизированные системы управления и приборы автоматики. – Харьков: ХНУРЭ, 2011. - Вып. 157. - С. 48-58.
69. Ostroukh A.V., Kuftinova N.G. Automation of Planning and Management of the Transportation of Production for Food Processing Industry Enterprises // Automatic Control and Computer Sciences. 2012. Vol. 46. No. 1. pp. 41-48. DOI: 10.3103/S0146411612010063.
70. Остроух А.В. Автоматизация планирования и управления транспортировкой продукции предприятий пищевой промышленности / А.В. Остроух, Н.Г. Куфтинова // Автоматика и вычислительная техника. – Рига: ИЭ и ВТ Латвийского университета, 2012. − №1. − С.57-67.
71. Остроух А.В. Розробка автоматизованої навігаційної системи диспетчерського контролю та обліку роботи транспорту нафтовидобувних підприємств / А.В. Остроух, А.Б. Николаев, Д.Б.

Ефименко, С.В. Жанказиев // Нафтова і газова промисловість. – Киев: Укргазпром, 2012. – № 3. – С. 48-52.

72. Данчук К.А. Автоматизированные информационные системы на автотранспортном предприятии / К.А. Данчук, А.Б. Львова, С.А. Порфирьева, А.В. Остроух, П.С. Якунин // В мире научных открытий. – 2012. – №2.6 (27). – С. 34-38.

73. Ефименко Д.Б. Использование программного обеспечения радионавигационных диспетчерских систем для транспортного обслуживания специальных объектов нефтедобывающих компаний / Д.Б. Ефименко, А.В. Остроух, А.Б. Николаев, А.Р. Исмаилов // Автотранспортное предприятие. – 2012. – №2. – С. 42-44.

74. Волков Н.Н. Автоматизация процессов начальной установки бинокулярных астровизиров в бортовых автономных информационно-вычислительных системах ориентации и навигации / А.В. Остроух, Н.Н. Волков, Е.А. Прохорова // Авиакосмическое приборостроение. – 2012. – №10. – С. 8-15.

75. Куфтинова Н.Г. Разработка информационно-логической модели транспортной сети мегаполиса / Н.Г. Куфтинова, А.В. Остроух // Бюллетень транспортной информации. – 2013. – №1 (211). – С. 23-26.

76. Польгун М.Б. Автоматизация процессов диспетчерского управления городским пассажирским транспортом / М.Б. Польгун, А.В. Остроух, А.Б. Николаев, Д.Б. Ефименко // Промышленные АСУ и контроллеры. – 2013. – №5. – С. 10-16.

77. Исмаилов А.Р. Разработка архитектуры подсистемы планирования организации заказных перевозок клиентских групп при проведении XXII зимних Олимпийских Игр / А.В. Остроух, А.Р. Исмаилов, А.Б. Львова // Бюллетень транспортной информации. – 2013. – №12 (222). – С. 3-10.

78. Исмаилов А.Р. Автоматизированная система планирования заданий для заказных перевозок пассажиров при проведении Олимпийских Игр / А.Р. Исмаилов, А.В. Остроух, А.Б. Львова, А.Б. Николаев, П.Ю. Збавитель // Транспорт Российской Федерации. - 2013. - №6 (49). - С. 46-51.
79. Исмаилов А.Р. Программно-технологические решения по разработке подсистем планирования заданий для заказных перевозок пассажиров при проведении Олимпийских Игр / А.Р. Исмаилов, А.Б. Львова, А.В. Остроух // Приборы и системы. Управление, контроль, диагностика. – 2013. – №11. – С. 74-85.
80. Ismailov A., Lvova A., Nikolaev A., Ostroukh A. Organization and Management of Transport Provided for the Guests and Participants of the Olympic Games // Middle East Journal of Scientific Research. 2013. Vol. 17 (8). pp. 1098-1104. DOI: 10.5829/idosi.mejsr.2013.17.08.12300.
81. Башмаков И.А., Остроух А.В. Процессная модель технологии транспортировки бетонных смесей автомобильным транспортом // Автоматизация и управление в технических системах. – 2013. – № 4.1. – С. 75-81. DOI: 10.12731/2306-1561-2013-4-14.
82. Джха П., Остроух А.В. Автоматизированная система управления складом железобетонных изделий // Автоматизация и управление в технических системах. – 2013. – № 4.1. – С. 51-56. DOI: 10.12731/2306-1561-2013-4-9.
83. Остроух А.В., Синха Б.Р. Исследование информационных систем управления взаимоотношениями с поставщиками // Автоматизация и управление в технических системах. – 2013. – № 4.1. – С. 56-62. DOI: 10.12731/2306-1561-2013-4-10.
84. Башмаков И.А., Польгун М.Б., Джха П., Остроух А.В. Обзор технологий транспортировки бетонных смесей автомобильным

транспортом // Автоматизация и управление в технических системах. – 2013. – № 4.2. – С. 178-189. DOI: 10.12731/2306-1561-2013-4-38.

85. Башмаков И.А., Польгун М.Б., Остроух А.В. Оптимизация параметров процессов автотранспортного обслуживания потребителей бетонных смесей // Автоматизация и управление в технических системах. – 2013. – № 4.2. – С. 189-198. DOI: 10.12731/2306-1561-2013-4-39.

86. Башмаков И.А., Остроух А.В. Минимизация производственных рисков при автотранспортном обслуживании потребителей бетонных смесей // Автоматизация и управление в технических системах. – 2013. – № 4.2. – С. 83-90. DOI: 10.12731/2306-1561-2013-4-40.

87. Ostroukh A.V., Polgun M.B. Automation of processes supervisory control urban passenger transport // International Journal of Advanced Studies (iJAS). 2013. Vol. 3. No 3. pp. 3-9. DOI: 10.12731/2227-930X-2013-3-1.

88. Ostroukh A.V., Polgun M.B. New approaches to development of automated supervisory systems of industrial enterprises transport // International Journal of Advanced Studies (iJAS). 2013. Vol. 3. No 4. pp. 3-9. DOI: 10.12731/2227-930X-2013-4-1.

89. Остроух А.В., Исмаилов А.Р., Львова А.Б. Проектирование программно-технологических решений подсистемы планирования заданий для заказных перевозок пассажиров при проведении XXII Зимних Олимпийских Игр // Автоматизация и управление в технических системах. – 2014. – № 1.1 (8). – С. 98-111. DOI: 10.12731/2306-1561-2014-1-11.

90. Остроух А.В. Применение графов дорожной сети в автоматизированной системе управления транспортировкой дорожно-строительных материалов / А.В. Остроух, М.Б. Польгун, В.В. Тихоцкий // Приборы и системы. Управление, контроль, диагностика. – 2014. – №1. – С. 12-17.

91. Ostroukh A.V., Bashmakov I.A., Surkova N.E. Process Model of the Technology of Concrete Mixtures Transportation by Road // World Applied Sciences Journal (WASJ). 2014. Vol. 31, No 4. pp. 500-507. DOI: 10.5829/idosi.wasj.2014.31.04.333.
92. Остроух А.В., Исмаилов А.Р., Львова А.Б. Разработка логической модели данных подсистемы планирования заказных перевозок при проведении XXII Зимних Олимпийских Игр // Автоматизация и управление в технических системах. – 2014. – № 1.2 (9). – С. 111-122. DOI: 10.12731/2306-1561-2014-1-25.
93. Krupensky N.A., Ivakhnenko A.M., Ostroukh A.V. Process-oriented aggregation scorecards operational control forwarding activities // Автоматизация и управление в технических системах. – 2014. – № 1.2 (9). – С. 123-142. DOI: 10.12731/2306-1561-2014-1-26.
94. Ostroukh A.V., Bashmakov I.A., Polgun M.B. Process-Functional Model of Transportation Mix Concrete // Journal of Transportation Technologies (JTT). 2014. Vol. 4, Issue 2, pp.157-163. DOI: 10.4236/jtts.2014.42016.
95. Башмаков И.А., Остроух А.В. Процессная модель технологии транспортировки бетонных смесей автомобильным транспортом // Промышленные АСУ и контроллеры. – 2014. – №3. – С. 10-18.
96. Andrey Ostroukh, Andrey Ivakhnenko, Nikita Krupensky. Development of Process-Oriented System For Operational Control of Freight Forwarding Activity // Journal of Applied Sciences (JAS). 2014. Vol. 14. No 20. pp. 2601-2607. DOI: 10.3923/jas.2014.2601.2607.
97. Andrey Ostroukh, Andrey Ismailov, Anna Lvova, Andrey Nikolaev. Architecture of Automated Navigation System of Passenger Transportation at Winter Olympic Games // Trends in Applied Sciences Research (TASR). 2014. Vol. 9. No 8. 425-437. DOI: 10.3923/tasr.2014.425.437.

98. Куфтинова Н.Г., Остроух А.В. Разработка автоматизированной системы обследования пассажиропотоков // Автоматизация и управление в технических системах. – 2014. – № 3 (11). – С. 23-33. DOI: 10.12731/2306-1561-2014-3-3.
99. Ostroukh A.V., Elhadi H. COMPARATIVE STUDY OF ROUTING PROTOCOLS IN VEHICULAR AD-HOC NETWORKS (VANETS) // International Journal of Advanced Studies (iJAS). 2014. Vol. 4. No 2. pp. 9-14. DOI: 10.12731/2227-930X-2014-2-2.
100. Kuftinova N.Ya.G.E., Ostroukh A.V. Development of an automated system of survey passenger traffics // International Journal of Advanced Studies (iJAS). 2014. Vol. 4. No 4. pp. 3-9. DOI: 10.12731/2227-930X-2014-4-2.
101. Kuftinova N.G., Ostroukh A.V., Vorobieva A.V. Automated Control System For Survey Passenger Traffics // International Journal of Applied Engineering Research. 2015. Vol. 10. No 7. pp. 16419-16427.
102. Гусеница Д.О., Юрчик П.Ф., Остроух А.В. Применение облачного хранения данных в автоматизированной системе диспетчерского управления транспортом // Промышленные АСУ и контроллеры. – 2015. – №5. – С. 59-66.
103. Gusenitsa D.O., Yurchik P.F., Ostroukh A.V. Cloud Computing Application on Transport Dispatching Informational Support Systems // International Journal of Advanced Studies (iJAS). 2015. Vol. 5. No 1. pp. 22-27. DOI: 10.12731/2227-930X-2015-1-6.
104. Ostroukh A.V., Surkova N.E., Polgun M.B., Vorobieva A.V. AUTOMATED SUPERVISORY CONTROL SYSTEM OF URBAN PASSENGER TRANSPORT // ARPN Journal of Engineering and Applied Sciences. 2015. Vol. 10. No 10. pp. 4334-4340.
105. Остроух А.В., Суркова Н.Е., Воробьева А.В., Салих Х.С. Математическая модель системы дистанционной диагностики

неисправностей автомобилей // В мире научных открытий. – 2015. – №6. – С. 63-70. DOI: 10.12731/WSD-2015-6-6.

106. Васюгова С.А. Исследование перспектив и проблем интеграции человека с компьютером: искусственный интеллект, робототехника, технологическая сингулярность и виртуальная реальность / С.А. Васюгова, А.В. Остроух, М.Н. Краснянский, А. Самаратунга // Перспективы науки. – Тамбов: «ТМБПринт», 2011. – № 4(19). – С. 109-112.

107. Белоусова А.И. Подход к формированию многоуровневой модели мультиагентной системы с использованием миваров / А.И. Белоусова, О.О. Варламов, М.Н. Краснянский, А.В. Остроух // Перспективы науки. – Тамбов: «ТМБПринт», 2011. – № 5(20). – С. 57-61.

108. Варламов О.О. Анализ возможностей миварного подхода для систем искусственного интеллекта и современной робототехники / О.О. Варламов, А.В. Остроух, М.Н. Краснянский, Т.Л. Давыдова // Вестник ТГТУ. – 2011. – Т.17. – № 3. – С.687-694.

109. Ostroukh A., Nikonov V., Ivanova I., Morozova T., Strakhov V. Distributed System of Real Time Head Gesture Recognition in Development of Contactless Interfaces // Middle East Journal of Scientific Research. 2014. Vol. 20 (12). pp. 2177-2183. DOI: 10.5829/idosi.mejsr.2014.20.12.21105.

110. Ostroukh A., Nikonov V., Ivanova I., Morozova T., Sumkin K., Akimov D. Development of Contactless Integrated Interface of Complex Production Lines // Journal of Artificial Intelligence (JAI). 2014. Vol. 7, No 1. pp. 1-12. DOI: 10.3923/jai.2014.1.12.

111. Morozova T., Sumkin K., Akimov D., Ostroukh A. Contactless integrated interface of production lines // International Journal of Advanced Studies

(iJAS). 2014. Vol. 4, Issue 1, pp. 32-38. DOI: 10.12731/2227-930X-2014-1-6.

112. Омар М., Омар Ф., Исмоилов М.И., Остроух А.В. Применение систем распознавания образов в различных предметных областях // Автоматизация и управление в технических системах. – 2014. – №4 (12). – С. 32-47. DOI: 10.12731/2306-1561-2014-4-4.

113. Омар М., Омар Ф., Исмоилов М.И., Остроух А.В. Анализ современного состояния развития интеллектуальных роботов // Автоматизация и управление в технических системах. – 2014. – №4 (12). – С. 48-54. DOI: 10.12731/2306-1561-2014-4-5.

114. Ле К.Х., Суркова Н.Е., Остроух А.В. Генетические алгоритмы в задачах рациональной организации информационно-вычислительных процессов сетей // Автоматизация и управление в технических системах. – 2014. – №4 (12). – С. 82-99. DOI: 10.12731/2306-1561-2014-4-9.

115. Valeriy Ivchenko, Petr Krug, Tatiana Morozova, Andrey Ostroukh, Sergey Pavelyev. The Remotely Reconfigurable Intelligence of the Space-Based Mobile Robot // Journal of Engineering and Applied Sciences. 2014. Vol. 9. Issue 10. pp. 389-395. DOI: 10.3923/jeasci.2014.389.395.

116. Ostroukh A.V., Gusenitsa D.O., Golubkova V.B., Yurchik P.F. Integration of PDM and ERP systems within a unified information space of an enterprise // IOSR Journal of Computer Engineering (IOSR-JCE). 2014. Vol. 16. Issue 02. V6. pp. 31-33. DOI: 10.9790/0661-16263133. ANED: 11.0661/iosr-jce-E016263133.

117. Akimov D.A., Ignatov A.S., Ostroukh A.V., Pavelyev S.A., Tsunikova T.G. Modeling of Temporal Stability to Critical States for Predicting Operational Safety of Turbogenerators // International Journal of Signal System Control and Engineering Application. 2018. Vol. 11. No. 1. pp. 13-19. DOI: 10.3923/ijssceapp.2018.13.19

118. Ostroukh A.V., Kolbasin A.M., Moroz D.G., Vorobiev A.I., Gavriluk M.V., Akhterov A.V. Remote Monitoring Cement Silo Stock Data // Research Journal of Applied Sciences. 2018. Vol. 13. No 2. pp. 146-149. DOI: 10.3923/rjasci.2018.146.149.
119. Ostroukh A., Surkova N., Varlamov O., Chernenky V., Baldin A. Automated process control system of mobile crushing and screening plant // Journal of Applied Engineering Science. 2018. Vol. 16. No. 3. pp 343-348. DOI: 10.5937/jaes16-15586.
120. Остроух А.В., Воробьева А.В., Колбасин А.М. Цифровая система удаленной диагностики автомобильного транспорта // Промышленные АСУ и контроллеры. – 2018. – № 11. – С. 3-10.
121. Arbuzov V., Ivchenko V., Matiukhina E., Pavelyev S., Ostroukh A. APPLICATION OF NEURAL NETWORK TECHNOLOGIES FOR DIAGNOSTICS OF THE TECHNICAL STATE OF POWER PLANT TURBO GENERATORS BASED ON SPECTROGRAMS OF THE VIBRATION MEASUREMENTS // ARPN Journal of Engineering and Applied Sciences. 2018. Vol. 13. No 7. pp. 2549-2555.
122. Остроух А.В., Листов Н.О. Система информационной поддержки адаптивного машинного обучения искусственных нейронных сетей с обратной связью // Промышленные АСУ и контроллеры. – 2018. – № 12. – С. 13-22.
123. Остроух А.В., Колбасин А.М. Концепция цифрового бетонного завода башенного типа // Промышленные АСУ и контроллеры. – 2020. – № 4. – С. 39-50.
124. Остроух, А.В. Системы искусственного интеллекта [Электронный ресурс] : монография / А.В. Остроух, Н.Е. Суркова. — Электрон. дан. — Санкт-Петербург : Лань, 2019. — 228 с. — Режим доступа: https://e.lanbook.com/book/113401. — Загл. с экрана.

125. Остроух, А.В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А.В. Остроух, А.Б. Николаев. — Электрон. дан. — Санкт-Петербург : Лань, 2019. — 308 с. — Режим доступа: https://e.lanbook.com/book/115518. — Загл. с экрана.
126. Pronin C., Ostroukh A., Pronin B., Vasiliev, Y., Kotliarskiy, E. Development of a quantum algorithm based on quantum parallelism for finding the shortest path in a graph // ARPN Journal of Engineering and Applied Sciences. 2019. Vol. 14. No 4. pp. 848-851.
127. Ostroukh A., Vakhrushev O., Maikov K., Kolbasin A. Combined identification method by reconstruction and analysis of face 3D structure // ARPN Journal of Engineering and Applied Sciences. 2019. Vol. 14. No 7. pp. 1385-1388.
128. Ostroukh A., Vasiliev, Y., Kotliarskiy, E., Sarychev, I. Connected Quarry Machines Digital Systems // ARPN Journal of Engineering and Applied Sciences. 2019. Vol. 14. No 1. pp. 135-140.

Научное издание

Андрей Владимирович Остроух

ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Монография

Компьютерный набор и верста: Максимов Я.А.

Корректор: Доценко Д.В.

Дата выхода в свет: 31.03.2020.
http://nkras.ru/arhiv/2020/ostroukh1.pdf

По вопросам приобретения и издания литературы обращаться по адресу:
Издательство «Научно-инновационный центр»
ул. 9 Мая, 5/192, г. Красноярск, 660127 Россия
тел. +7 (923) 358-10-20
Электронная почта: monography@nkras.ru
Дополнительная информация на сайте: www.nkras.ru