ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В.Г. Спицын, Ю.Р. Цой

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета

Издательство
Томского политехнического университета
2008

УДК 681.51.012(075.8)
ББК 32.813я73
С73

Спицын В.Г.

С73 Представление знаний в информационных системах: учебное пособие / В.Г. Спицын, Ю.Р. Цой. – Томск: Изд-во Томского политехнического университета, 2008. – 152 с.

ISBN 5-98298-354-3

В учебном пособии рассматриваются современные модели представления знаний в информационных системах и принципы построения экспертных систем; обсуждаются проблемы применения нечеткой логики, генетических алгоритмов и нейронных сетей в интеллектуальных информационных системах; содержатся методические указания и задания для выполнения лабораторных работ.

Разработано в рамках реализации Инновационной образовательной программы ТПУ по направлению «Информационно-коммуникационные системы и технологии» и предназначено для студентов, обучающихся по магистерской программе «Компьютерный анализ и интерпретация данных» направления «Информатика и вычислительная техника».

УДК 681.51.012(075.8)
ББК 32.813я73

Рецензент

Доктор технических наук, профессор ТУСУРа
А.М. Кориков

ISBN 5-98298-354-3

ПРЕДИСЛОВИЕ

В данном учебном пособии приведены современные модели представления знаний, принципы построения экспертных систем (ЭС) и перспективные направления развития систем, основанных на знаниях.

Первая глава посвящена краткому изложению истории возникновения систем искусственного интеллекта, процесса мышления человека, классификации систем, основанных на знаниях, а также методов извлечения знаний из экспертов инженером по знаниям. Во второй главе рассматриваются наиболее распространенные модели представления знаний. Третья глава посвящена изложению архитектуры и технологии разработки ЭС, описанию систем анализа и синтеза входных и выходных сообщений.

В заключительных главах излагаются получившие широкое распространение методы решения неформализованных задач. К этим методам относятся нечеткая логика, генетические алгоритмы и искусственные нейронные сети. Модель представления знаний на основе нечеткой логики излагается в четвертой главе. В настоящее время для настройки и обучения искусственных нейронных сетей все чаще применяются генетические алгоритмы. С их помощью создаются искусственные нейронные сети, адаптированные для решения конкретных задач. В пятой главе рассматривается генетический алгоритм и даются рекомендации для его программной реализации. Нейросетевой модели посвящена шестая глава. Знания в нейросетевой модели представляются неявным образом посредством задаваемой топологии сети, весов связей и типов функции активации.

В Приложениях 1–3 содержатся контрольные вопросы и задания, темы рефератов и индивидуальных заданий, приведены ресурсы сети Интернет.

Таким образом, в учебном пособии авторами рассматриваются следующие проблемы:

- системы, основанные на знаниях, и методы извлечения знаний;
- модели представления знаний;
- архитектура и технология построения ЭС;
- применение нечеткой логики в ЭС;
- генетические алгоритмы и их применение для решения задач оптимизации;

- основные принципы функционирования искусственных нейронных сетей и возможности их применения для решения задач классификации и аппроксимации.

В пособии содержатся задания для выполнения лабораторных работ по программной реализации ЭС, генетических алгоритмов и нейронных сетей.

В результате освоения изложенного материала студент сможет самостоятельно приступить к разработке ЭС в роли инженера по знаниям.

Следует отметить, что авторы данного пособия продолжают исследования в области применения генетических алгоритмов для настройки и обучения искусственных нейронных сетей при решении задач улучшения качества изображений, классификации, аппроксимации, моделирования, адаптивного управления и поведения.

Глава 1
СИСТЕМЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ

1.1. История создания искусственного интеллекта

Родоначальником искусственного интеллекта (ИИ) считается средневековый испанский философ, математик и поэт Раймонд Луллий. Он еще в XIII в. попытался создать механическую машину для решения различных задач на основе разработанной им всеобщей классификации понятий. В XVIII в. Лейбниц и Декарт независимо друг от друга предложили универсальные языки классификации всех наук. Эти работы можно считать первыми теоретическими работами в области искусственного интеллекта.

Появление искусственного интеллекта как научного направления произошло после создания ЭВМ в 40-х гг. XX в. В это время Норберт Винер создал свои основополагающие работы по кибернетике. Вскоре после признания искусственного интеллекта отдельной отраслью науки произошло разделение его на два направления: нейрокибернетика и кибернетика «черного ящика». Эти направления развиваются практически независимо, существенно различаясь как в методологии, так и в технологии. И только в настоящее время появились тенденции к их объединению [1].

Основная идея первого направления заключается в том, что единственный объект, способный мыслить, – это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру. Нейрокибернетика ориентирована на программно-аппаратное моделирование структур, подобных структуре мозга. Как известно из физиологии, человеческий мозг содержит до 10^{11} взаимодействующих между собой нервных клеток – нейронов [1, 2]. В рамках нейрокибернетики создавались нейроноподобные элементы, которые объединялись в функционирующие системы, называемые нейронными сетями.

В основу второго направления ИИ, называемого кибернетикой «черного ящика», был положен альтернативный принцип, суть которого заключается в том, что не имеет значения, как построено «мыслящее» устройство. Требуется, чтобы на заданные входные воздействия оно реагировало так же, как и человеческий мозг.

Это направление ИИ было ориентировано на поиски алгоритмов решения интеллектуальных задач на существующих моделях компью-

теров. В 1963–1970-х гг. решение задач стало основываться на методах математической логики. Робинсоном был предложен метод резолюций, на основе которого автоматически доказывались теоремы при наличии набора исходных аксиом. В это же время русский ученый Ю.С. Маслов предложил метод обратного вывода, решающий ту же задачу другим способом [1, 3]. На основе метода резолюций француз А. Кольмероэ в 1973 г. создал язык логического программирования Пролог.

В США в середине 1970-х г. на смену поискам универсального алгоритма мышления пришла идея моделировать конкретные знания специалистов-экспертов. Появились первые коммерческие системы, основанные на знаниях, или экспертные системы. Начиная с середины 1980-х гг., повсеместно происходит коммерциализация ИИ. Растут капиталовложения и создаются промышленные ЭС. Искусственный интеллект становится одной из наиболее перспективных и престижных областей информатики (computer science).

Следует отметить, что уровень теоретических исследований по ИИ в России сопоставим с мировым. Но, к сожалению, начиная с 1980-х гг., на прикладных работах начинает сказываться отставание в технологии. По оценкам специалистов, отставание в области разработки интеллектуальных промышленных систем составляет 3–5 лет [1].

1.2. Процесс мышления

Процесс мышления, протекающий в человеческом сознании, чрезвычайно сложен. Одна ячейка человеческого глаза способна выполнять за 10 мс обработку, эквивалентную решению системы из 500 нелинейных дифференциальных уравнений. Компьютеру Cray-1 потребовалось бы несколько минут для решения этих уравнений. Поскольку глаз человека насчитывает не менее 10^7 ячеек и каждая из них взаимодействует с другими, то компьютеру Cray-1 необходимо затратить, по меньшей мере, 100 лет, чтобы воспроизвести процессы, происходящие ежесекундно в человеческом глазе [4].

Данные из внешнего мира воспринимаются человеком с помощью одного из пяти органов чувств и затем помещаются в буфер кратковременной памяти для анализа. В другой области памяти (долговременной) хранятся символы и смысловые связи между ними, которые используются для объяснения новой информации, поступающей из кратковременной памяти. В долговременной памяти хранятся не столько факты и данные, сколько объекты и связи между ними, т. е. символьные образы. Большие объемы данных постоянно записываются в кратковременную память, и мы непрерывно анализируем и фильтруем получаемую информацию

для того, чтобы определить степень ее важности и то, как она соотносится с образами, хранящимися в долговременной памяти (рис. 1.1).

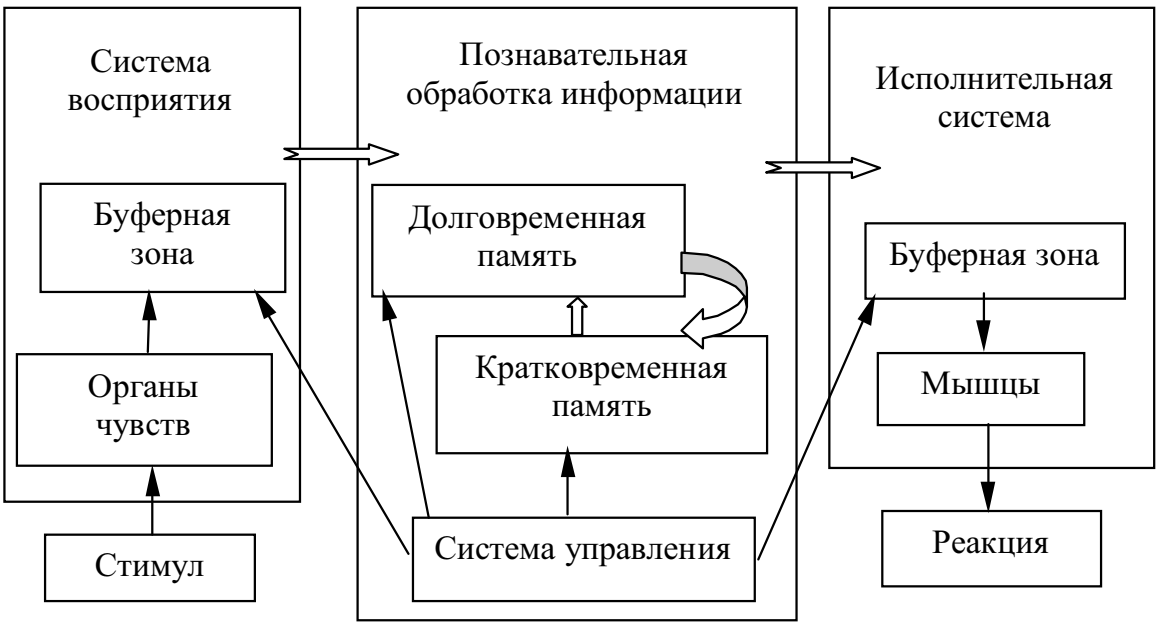


Рис. 1.1. Система обработки информации у человека

Учеными установлено местоположение всех областей памяти в человеческом мозге. Логично предположить, что имеющиеся в мозге области памяти объединяются в единую структуру, в которой и происходит обработка поступающей информации.

Доступ к информации в долговременной памяти осуществляется достаточно эффективно. Практически любой элемент данных может быть извлечен в течение цикла обработки продолжительностью не более 70 мс и затем преобразован. Например, мы инстинктивно отдергиваем руку от горячей печки или резко поворачиваем руль автомобиля при возникновении препятствия на дороге, используя образы, ранее запомненные в долговременной памяти.

Механизм запоминания информации в долговременной памяти представляет собой самостоятельную и обширную тему для исследований. Требуется приблизительно семь секунд для записи одного образа в долговременную память и установления всех связей, необходимых для извлечения этого образа в будущем. Перемещение данных из кратковременной памяти в долговременную память занимает приблизительно 15–20 мин. Если человек в автокатастрофе получил мозговую травму, то долговременная память может восстановиться почти полностью. Однако вся информация, поступившая в течение 15–20 мин до катастрофы, будет потеряна и никогда не восстановится [4].

Можно провести аналогию между кратковременной памятью человека и оперативной памятью ЭВМ, для которой отключение электропитания означает полное уничтожение всех данных. Долговременная память человека похожа на дисковую память ЭВМ. В долговременной памяти образы существуют в виде циркулирующих электрохимических импульсов и физических нейронных взаимодействий. Человек часто полностью выздоравливает после повреждения мозга и уничтожения нейронов в автокатастрофе, если у него не разрушены речевые центры или нейроны, управляющие двигательной системой. Нейроны других типов, даже если они частично повреждаются при травме, могут сохранять работоспособность благодаря зафиксированной ранее информации.

Целесообразно остановиться на организации хранения информации в человеческой памяти. Эта тема представляет наибольший интерес для разработчиков интеллектуальных систем. Способ хранения символьных образов в долговременной памяти во многом схож со способом хранения числовой информации в базах данных сетевого типа. Человеческая память хранит не числовые данные, как, например, номера элементов некоторой совокупности, а образы или символы. В памяти существует система указателей, позволяющая нам быстро извлечь любой нужный символ и все данные, которые с ним связаны. Мозг человека организован более совершенно, чем компьютерные базы данных, т.к. символьные образы в нем объединены в чанки – наборы фактов и связей между ними, запомненные и извлекаемые как единое целое. Чанки хранятся совместно с взаимосвязями между ними. В каждый момент времени человек может обрабатывать и интерпретировать не более четырех-семи чанков.

Чтобы убедиться в этом, прочтите следующую строку, закройте книгу и попытайтесь написать на листе бумаги то, что вы прочли:

Нлонмйкртс зрззйнк внбнм гвссмзл.

Теперь проверьте написанный вами текст и подсчитайте, сколько символов вам удалось воспроизвести правильно. Вероятно, их окажется от четырех до семи.

Повторите эксперимент со следующим предложением:

Экспертные системы почти разумны.

На этот раз вы, видимо, написали все буквы правильно. При этом оба предложения состоят из одинакового количества букв и слов. Почему же так отличаются результаты? Во втором случае вы объединили все объекты (буквы) в четыре чанка и в действительности запомнили только их и связи между ними, что соответствует допустимому для человеческого мозга диапазону. В первом же примере вы должны были запомнить 30 чанков, т. е. выйти за пределы допустимого числа чанков.

Используя ту же стратегию, опытный шахматист может, бегло взглянув на доску, запомнить положение всех фигур. При этом он запоминает не положение каждой фигуры, а их комбинации (чанки), и поэтому безошибочно восстанавливает расположение фигур на доске.

Способность формировать чанки отличает эксперта в данной конкретной области от неэксперта. Эксперт развивает свою способность объединять в чанки большие объемы данных и устанавливать иерархические связи между ними для того, чтобы быстро извлекать эти данные из памяти и с их помощью распознавать новые ситуации по мере поступления информации о них в мозг. Средний специалист в конкретной предметной области помнит от 50 000 до 100 000 чанков, которые могут быть использованы для решения той или иной проблемы. Их накопление в памяти человека и построение указателей для такого объема данных требуют от 10 до 20 лет [4].

1.3. Основные понятия и классификация систем, основанных на знаниях

ИИ ориентирован на создание методов дублирования функций живых интеллектуальных систем искусственными. Технология разработки интеллектуальных программных средств основана на том, что знания о решении задач отделяются от программ и реализуются в виде базы знаний (БЗ), а в программах реализуется алгоритм манипулирования этими знаниями. Этот алгоритм называют механизмом логического вывода. Знания являются явными и доступными, что отличает интеллектуальные системы от большинства традиционных программных средств.

К знаниям относят информацию о логике решения задач, а к данным – информацию, которая должна быть проанализирована в соответствии с этой логикой. Поэтому имеются специфические признаки, отличающие знания от данных:

- внутренняя интерпретируемость, которая означает, что в знаниях находится информация, раскрывающая смысл элементов знаний;
- структурированность знаний, заключающаяся в возможности декомпозиции сложных объектов на более простые и в установлении соответствующих связей между ними;
- связность знаний отражает причинно-следственные и временные отношения между фактами, процессами и явлениями;
- активность знаний, т.к. они могут содержать планы действий и управляющие процедуры.

Знания, которыми обладает специалист в какой-либо области (дисциплине), можно разделить на формализованные (точные) и неформализованные (неточные) знания.

Формализованные знания формулируются в виде общих и строгих суждений (законов, формул, моделей и алгоритмов), отражающих универсальные знания.

Неформализованные знания отличаются конкретностью, субъективностью и приблизительностью. Обычно они являются результатом обобщения многолетнего опыта работы и интуиции специалистов и представляют собой многообразие эмпирических приемов и правил.

В зависимости от того, какие знания преобладают в той или иной предметной области, ее относят к формализованной (если преобладают точные знания) или к неформализованной (если преобладают неточные знания) предметной области. *Задачи*, решаемые на основе точных знаний, называют *формализованными*, а задачи, решаемые с помощью неточных знаний, – *неформализованными* [5].

Традиционное программирование в качестве основы для разработки программ использует алгоритм, т. е. формализованное знание. Поэтому до недавнего времени считалось, что ЭВМ не приспособлены для решения неформализованных задач. Расширение сферы использования ЭВМ показало, что неформализованные задачи составляют очень важный класс задач, по-видимому, значительно больший, чем класс формализованных задач. Неумение решать неформализованные задачи сдерживает внедрение ЭВМ в описательные науки. Поэтому исследования в области ЭС имеют большое значение и занимают важное место в информатике.

К неформализованным задачам относятся те, которые обладают одной или несколькими из следующих особенностей:

- алгоритмическое решение задачи неизвестно (хотя, возможно, и существует) или не может быть использовано из-за ограниченности ресурсов ЭВМ (времени, памяти);
- задача не может быть представлена в числовой форме (требуется символьное представление);
- цели задачи не могут быть выражены в терминах точно определенной целевой функции [5].

Как правило, неформализованные задачи обладают неполнотой, ошибочностью, неоднозначностью и (или) противоречивостью знаний (как данных, так и используемых правил преобразования). ЭС отличаются от традиционных программ тем, что ориентированы на решение неформализованных задач. Основными особенностями ЭС являются следующие:

- алгоритм решения неизвестен заранее, а строится самой системой с помощью символических рассуждений, базирующихся на эвристических приемах;

- способность анализа и объяснений своих действий и знаний;
- способность приобретения новых знаний от пользователя – эксперта, не знающего программирования, и изменения в соответствии с ними своего поведения;
- обеспечение «дружественного», как правило, естественно-языкового интерфейса с пользователем.

С точки зрения пользователя, наиболее важными свойствами ЭС являются:

- возможность предоставления подробных объяснений полученных решений;
- возможность постепенного наращивания БЗ без перепрограммирования;
- знания из БЗ обладают самостоятельной ценностью и могут распространяться.

Перечислим основные виды интеллектуальных систем, отличающиеся целью и реализацией знаний:

- интеллектуальные информационные поисковые системы, обеспечивающие в процессе диалога взаимодействие конечных пользователей с базами знаний на профессиональных языках пользователей;
- интеллектуальные пакеты прикладных программ, в которых, в отличие от обычных пакетов, реализуются знания о процессе решения задач предметной области;
- ЭС, обеспечивающие возможность решения задач в полностью неформализованных областях, причем вся информация о решении задач предметной области реализуется в виде БЗ.

ЭС представляют собой класс компьютерных программ, которые выдают советы, проводят анализ, выполняют классификацию и ставят диагноз. Они ориентированы на решение задач, обычно требующих проведения экспертизы человеком-специалистом. В отличие от машинных программ, использующих процедурный анализ, ЭС решают задачи в узкой предметной области на основе дедуктивных суждений.

Классификация ЭС по решаемой задаче [1]:

1. Интерпретация – это анализ исходных данных с целью определения их смысла.
2. Диагностика – это процесс поиска неисправности в системе, который основан на интерпретации данных.
3. Мониторинг (наблюдение) – это задача непрерывной интерпретации сигналов и выдачи оповещений в тех случаях, когда контролируемые параметры выходят за допустимые пределы. Мониторинг является частью диагностической системы, которая работает в реальном масштабе времени.

4. Прогнозирование – это предсказание хода развития системы в будущем на основании ее поведения в прошлом и настоящем. Эти системы содержат блоки обработки статистики, принятия решения на основе неполной информации и генерации альтернативных путей развития системы.
5. Планирование – это определение планов действий, относящихся к объектам, способным выполнять некоторые функции.
6. Проектирование – подготовка спецификаций на создание «объектов» с заранее определенными свойствами. Цель: помочь человеку при нахождении им эвристических решений в процессе творчества или автоматизировать рутинную работу.
7. Обучение – это использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ. Система получает информацию о деятельности некоторого объекта (например студента) и анализирует его поведение. БЗ изменяется в соответствии с поведением объекта. Примером такого обучения может служить компьютерная игра, сложность которой автоматически увеличивается.
8. Контроль и управление – системы, основанные на знаниях, могут применяться в качестве интеллектуальных систем контроля и управления. Они способны принимать решения, анализируя данные, поступающие из разных источников. Такие ЭС уже работают на атомных электростанциях, управляют воздушным движением и осуществляют медицинский контроль.
9. Поддержка принятия решений – это совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающими процесс принятия решения. Эти ЭС помогают специалистам выбрать и сформировать оптимальное решение в конкретной ситуации.

1.4. Экспертные системы как элемент искусственного интеллекта

ИИ как наука появился более 50 лет назад. Задачей этой науки является воссоздание с помощью искусственных устройств (в основном с помощью ЭВМ) разумных рассуждений и действий.

При этом возникают трудности двух типов:

1. В большинстве случаев, выполняя какие-то действия, мы сами не осознаем, как мы это делаем. Нам неизвестен точный способ (или алгоритм) того, как именно происходит понимание текста, узнавание лица, доказательство теоремы, выработка плана действий, решение задач и т. д.

2. ЭВМ априори далеки от человеческого уровня компетентности: до начала любой их работы необходимо составить соответствующие программы. Однако в действительности языки программирования дают возможность выражать только элементарные понятия.

Следовательно, методы ИИ представляют собой экспериментальную научную дисциплину. Под экспериментом в данном случае понимается проверка и уточнение моделей (представляющих собой программы для ЭВМ) на многочисленных примерах-наблюдениях над человеком с целью раскрыть эти модели и лучше понять функционирование человеческого разума.

В течение последних пятнадцати лет в рамках исследований по ИИ сформировалось самостоятельное направление – ЭС или инженерия знаний.

Отличительной чертой компьютерных программ, называемых ЭС, является их способность накапливать знания и опыт наиболее квалифицированных специалистов (экспертов) в какой-либо узкой предметной области. Затем с помощью этих знаний пользователи ЭС, имеющие обычную квалификацию, могут решать свои текущие задачи столь же успешно, как это сделали бы сами эксперты. Такой эффект достигается благодаря тому, что ЭС в своей работе воспроизводит примерно ту же систему рассуждений, которую обычно применяет человек-эксперт при анализе проблемы. Тем самым ЭС позволяет копировать и распространять знания, делая уникальный опыт нескольких высококлассных профессионалов доступным широким кругам рядовых специалистов.

Большой интерес со стороны пользователей к ЭС вызван, по крайней мере, тремя причинами. Во-первых, они ориентированы на решение широкого круга задач в неформализованных областях, на приложения, которые до недавнего времени считались малодоступными для вычислительной техники. Во-вторых, с помощью ЭС специалисты, не знающие программирования, могут самостоятельно разрабатывать интересующие их приложения, что позволяет расширить сферу использования вычислительной техники. В-третьих, ЭС при решении практических задач достигают результатов, не уступающих, а иногда и превосходящих возможности людей-экспертов, не оснащенных ЭС.

В настоящее время технология ЭС получила широкое распространение. Так, на американском и западноевропейском рынке систем ИИ организациям, желающим создать ЭС, фирмы-разработчики предлагают сотни инструментальных средств для их построения. Тысячи прикладных ЭС способны успешно решать специализированные задачи. Это позволяет говорить о том, что ЭС сейчас составляют мощную ветвь в индустрии программных средств.

Вместе с тем следует отметить, что на пути к тому прочному положению, которое ЭС теперь занимают в качестве важного компонента новой информационной технологии, были и спады, и подъемы. ЭС довелось пережить и период безграничной веры во всемогущество, и период, когда высказывались сомнения в их полезности вообще. Неэффективность ЭС проявлялась, как правило, лишь в случаях их некорректного применения или на низко производительной аппаратуре, не соответствующей сложности предметной области, или в задачах, для решения которых они не предназначались.

Разочарование постигало разработчиков ЭС, как правило, тогда, когда они пытались их использовать в качестве инструмента для решения задач, требующих привлечения чисто человеческих приемов мышления, – аналогий, ассоциаций и интуиции. Следует отметить, что перечисленные выше приемы мышления вначале отсутствовали в ЭС даже в зачаточном состоянии.

Компьютерные системы, которые могут лишь повторить логический вывод эксперта, принято относить к ЭС первого поколения.

Однако специалисту, решающему интеллектуально сложную задачу, нужно, чтобы ЭС выступала в роли полноценного помощника и советника. ЭС должна быть способна проводить анализ нечисловых данных, выдвигать и отбрасывать гипотезы, оценивать достоверность фактов, самостоятельно пополнять свои знания, контролировать их непротиворечивость, делать заключения на основе прецедентов и, может быть, даже порождать решение новых, ранее не рассматривавшихся задач. Наличие таких возможностей является характерным для ЭС второго поколения. ЭС, относящиеся ко второму поколению, называют партнерскими системами или усилителями интеллектуальных способностей человека. Их общими отличительными чертами является умение обучаться и развиваться, т. е. эволюционировать.

В настоящее время ЭС применяются в различных областях человеческой деятельности: химии, геологии, медицине и т. д. Наибольшее распространение ЭС получили в проектировании интегральных микросхем, в поиске неисправностей, в военных приложениях и автоматизации программирования.

Применение ЭС позволяет:

- повысить при проектировании интегральных микросхем (по данным фирмы NEC) производительность труда в 3–6 раз, при этом выполнение некоторых операций ускоряется в 10–15 раз;
- ускорить поиск неисправностей в 5–10 раз;
- повысить производительность труда программистов (по данным фирмы Toshiba) в 5 раз;

- при профессиональной подготовке сократить (без потери качества) в 8–12 раз время на индивидуальную работу с обучаемым персоналом [5].

В настоящее время ведутся разработки ЭС для следующих приложений: раннее предупреждение национальных и международных конфликтов и поиск компромиссных решений; принятие решений в кризисных ситуациях; охрана правопорядка; законодательство; образование; планирование распределения ресурсов; системы организационного управления (кабинет министров, муниципалитет, учреждения) и т. д.

Рассмотрим наиболее популярные ЭС. Система PROSPECTOR предназначена для выдачи геологам следующей информации: наличие в анализируемой местности залежей полезных ископаемых; оценка геологических ресурсов района; выбор мест, благоприятных для бурения. Система создана фирмой SRI International. Работы над системой были начаты в 1974 г. и продолжались до 1983 г. Трудозатраты на создание системы оцениваются в 30 человеко-лет, система достигла промышленной стадии. При этом БЗ системы содержат свыше 1 000 правил и включают классификацию более 1 000 геологических понятий.

Система PROSPECTOR решает задачи, не вникая в суть происходящих в проблемной области процессов. Состояние проблемной области описывается в виде утверждений о проблемной области, например: «условия района благоприятны». В системе реализованы объяснительные возможности, выявляющие причины задания вопроса или состояние решения задачи. Во время решения задач не вводится новых утверждений о проблемной области. Все утверждения должны быть априорно перечислены экспертом.

Система MYCIN разработана в Стэндфордском университете и достигла стадии действующего прототипа. Назначение системы – оказание помощи лечащим врачам в постановке диагноза и назначение лечения в сложных случаях инфекционных заболеваний крови. Цель системы – определить организмы, являющиеся причиной заболевания, и назначить антибиотики избирательного действия. Система способна давать объяснения, ПОЧЕМУ потребовалась запрашиваемая информация и КАК получен результат. Система содержит около 800 правил.

Система R1 предназначена для определения конфигурации компонентов ЭВМ VAX 11/780, удовлетворяющей требованиям заказчика. В состав ЭВМ VAX 11/780 входит 420 компонентов, каждый из которых может иметь до 8–10 характеристик. Естественно, что число возможных конфигураций, построенных из этих компонентов, достаточно велико. Система выполняет следующие функции: определяет, не содержит ли заказ пользователя несовместимых компонентов, и выявляет не-

достающие; выдает заказчику конфигурацию ЭВМ VAX 11/780, которая используется при установке системы; учитывает обусловленные заказчиком ограничения. До создания системы решением этих задач на фирме занимались квалифицированные специалисты, тем не менее, часто допускающие ошибки. Система достигла коммерческой стадии существования, объем ее БЗ 3 000 правил языка OPS-5. Система работает в интерактивном режиме. Средства объяснения реализованы в виде заранее записанных текстов. Эффективность решения задач вызвана тем, что каждый шаг выполняется тогда, когда для этого имеется достаточно информации, чтобы его можно было сделать точно и в дальнейшем никогда не пересматривать. Поэтому процесс проектирования сводится к постоянному расширению частичной конфигурации. Усовершенствованная со временем система R1 получила название XCON.

К отечественным ЭС на ПЭВМ относится МОДИС-2, предназначенная для диагностики различных форм симптоматической гипертонии. Гипертонией страдают 10 % населения Земли. Причины повышения артериального давления очень разнообразны – более 30 основных заболеваний. Сложность диагностики этих заболеваний состоит в том, что они могут относиться к компетенции специалистов из различных областей медицины: нефрологии, ангиологии, урологии и т. п.

ЭС МОДИС-2 имеет важную особенность. Ее использование предполагается как в поликлиниках общего профиля, так и в специализированных учреждениях. Уровень доступной информации о больном в этих заведениях может быть разным. В поликлинике доступна информация общего характера: жалобы пациента, данные внешнего осмотра, история болезни и результаты общих анализов. В этом случае ЭС должна дать рекомендации, к каким специалистам следует обратиться, направить на специальные исследования. Таким образом, на основе имеющейся информации общего характера ЭС должна сузить круг подозреваемых заболеваний, выбрать наиболее вероятные. При использовании ЭС в специализированном учреждении может быть доступна более детальная информация о больном – данные специальных исследований. В этом случае ЭС должна ставить по возможности точный диагноз.

Представляет интерес остановиться на современном состоянии разработок в области ИИ. Начиная с 1950-х гг. ИИ развивается по следующим направлениям: разработка символических, биологических и квантово-механических методов. Первое направление включает в себя логику, фреймы и сценарии, а также ЭС. Второе направление содержит искусственные нейронные сети, эволюционные методы и биоинформатику. Третье направление рассматривает проблему создания компьютеров на основе применения законов квантовой механики [6].

Следует отметить большой успех ИИ в создании видеоигр, что привело к возможности получения университетского диплома специалиста по видеоиграм. Развитие систем ИИ и ЭС привело к появлению специальности *инженера по онтологии.* Под *онтологией* в данном случае понимается явная формальная спецификация терминов проблемной области и отношений между ними [6, 7].

Другими словами, онтологию можно представить как стандартный, согласованный набор терминов, применяемых для описания определенной прикладной области. Потребность в специалистах по онтологии увеличивается в связи с необходимостью классификации возрастающих объемов знаний, в частности при создании ЭС [6].

1.5. Теоретические аспекты извлечения знаний

Можно выделить три основных аспекта извлечения знаний: психологический, лингвистический и гносеологический.

Из перечисленных аспектов извлечения знаний психологический аспект является ведущим, т.к. он определяет эффективность взаимодействия инженера по знаниям и эксперта. Общение или коммуникация (лат. communicatio – связь) является междисциплинарным понятием, обозначающим все формы непосредственных контактов между людьми – от дружеских до деловых. Указанное понятие широко исследуется в психологии, философии, социологии, лингвистике и других науках [1].

Установлено, что общение не сводится к однонаправленному процессу передачи сообщений или двухтактному обмену порциями сведений. Скорее, общение можно охарактеризовать в виде нерасчлененного процесса циркуляции информации, целью которого является совместный поиск истины.

Таким образом, в результате общения вырабатывается новая информация, общая для всех участвующих в общении людей. Следует отметить, что культурой и наукой общения на профессиональном уровне владеют единицы.

Выделяются четыре основных уровня общения:

1. Уровень манипулирования, когда один субъект рассматривает другого как средство или помеху по отношению к проекту своей деятельности.
2. Уровень «рефлексивной игры», когда человек учитывает контрпроект другого субъекта, но не признает его ценность и стремится к реализации своего проекта.
3. Уровень правового общения, когда субъекты признают право на существование проектов деятельности друг друга и пытаются согласовать их хотя бы внешне.

4. Уровень нравственного общения, когда субъекты внутренне принимают общий проект взаимной деятельности.

Для достижения поставленной цели инженер по знаниям должен уметь общаться с экспертами на высшем четвертом уровне.

Как известно, потери информации при разговорном общении велики (рис. 1.2) [1]. В связи с этим представляет интерес увеличение информативности общения аналитика и эксперта за счет применения психологических знаний.

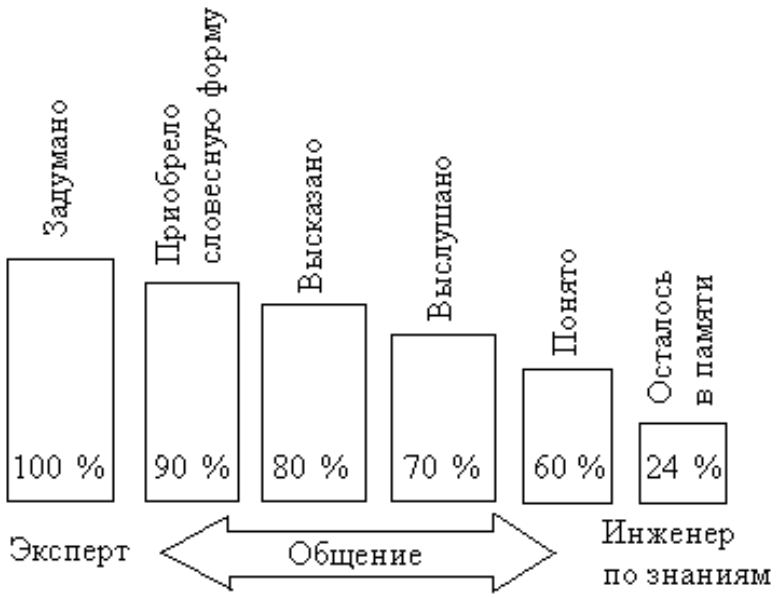


Рис. 1.2. Потери информации при разговорном общении

Психологи отмечают, что на коллективный процесс влияет атмосфера, возникающая в группе участников. Результаты экспериментов показывают, что зачастую дружеская атмосфера в коллективе больше влияет на результат, чем индивидуальные особенности участников. При этом важным является то, чтобы в коллективе складывались кооперативные, а не конкурентные отношения. Кооперация характеризуется атмосферой сотрудничества и взаимопомощи, что соответствует уровню нравственного общения. Отношения конкурентного типа характеризуются атмосферой индивидуализма и соперничества.

Можно выделить ряд особенностей личности, оказывающих влияние на эффективность процесса извлечения знаний. Под личностью обычно понимается система психологических черт, характеризующая индивидуальность человека. Рекомендуемыми компонентами личности инженера по знаниям являются: доброжелательность, хорошая память, внимание, наблюдательность, воображение, собранность, настойчивость, общительность, находчивость.

Как известно, флегматики и меланхолики медленнее усваивают информацию. Поэтому для обеспечения психологического контакта с ними не нужно задавать беседе быстрый темп и торопить их с ответом.

На эффективность коллективного решения задач влияет мотивация (стремление к успеху). Если некоторые эксперты охотно делятся своим опытом, то другие неохотно открывают свои профессиональные секреты. Поэтому инженер по знаниям должен изыскивать стимулы для экспертов. В некоторых случаях бывает полезно вызвать у эксперта стремление к соперничеству и конкуренции. При этом, конечно, нужно постараться не нарушить обстановку сотрудничества и взаимопомощи в коллективе.

Беседу с экспертом лучше проводить наедине, поскольку посторонние нарушают доверительность беседы. Атмосфера замкнутости пространства и уединенности способствует эффективности беседы. Считается, что для делового общения наиболее благоприятной является дистанция от 1,2 до 3 м. Минимальное комфортное расстояние составляет 0,7 м. Продолжительность сеанса не должна превышать 2 ч. Через 20−25 мин беседы обычно происходит взаимная утомляемость партнеров. Поэтому в сеансе извлечения знаний необходимо предусмотреть паузы [1].

Для увеличения эффективности процесса извлечения знаний применяется наглядный материал. Людей, занимающихся интеллектуальной деятельностью, можно отнести к художественному или мыслительному типу. Индивидуумы, относящиеся к первому типу, лучше воспринимают зрительную информацию в форме рисунков и графиков. Дело в том, что эта информация воспринимается через первую сигнальную систему. Эксперты мыслительного типа лучше понимают язык формул и текстовую информацию. Существенным является то, что большую часть информации человек получает через зрительное восприятие. Поэтому в процедуре извлечения знаний целесообразно активно использовать наглядный материал.

Наиболее распространенным способом протоколирования результатов является их запись на бумагу аналитиком в ходе беседы с экспертом. Однако здесь существует опасность потери знаний, т. к. запись ответа по сути уже является интерпретацией. Интерпретация зависит от степени понимания предмета аналитиком.

Аналитику необходимо учитывать индивидуальный темп и стиль участия в беседе эксперта. Отрицательный результат дает навязывание аналитиком собственного темпа и стиля.

На успех беседы влияет длина фраз. Установлено, что человек лучше всего воспринимает предложения длиной 7 ± 2 слова (число Ингве–Миллера). Опытные лекторы, используя короткие фразы в лекции, сводят потерю информации до 3 %. Несоблюдение этого принципа может приводить к потере до 20–30 % информации [1].

Лингвистический аспект извлечения знаний относится к исследованиям возникающих при этом языковых проблем. Различие языков, на которых говорят эксперт и аналитик, обусловливает возникновение языкового барьера между ними. Эти два языка являются отражением «внутренней речи» эксперта и аналитика. Психологи и эксперты полагают, что язык является основным средством мышления.

Можно предположить, что бытовой язык у эксперта и аналитика приблизительно совпадает. Наиболее существенное отличие заключается в знании общенаучной и специальной терминологии, принятой в предметной области. Поэтому перед партнерами появляется задача в выработке общего языка, который необходим для успешного взаимодействия.

Гносеологический аспект извлечения знаний связан с теорией отражения действительности в сознании человека. При этом вначале действительность отображается в сознании эксперта. Затем опыт эксперта интерпретируется сознанием инженера по знаниям. Далее происходит построение третьей интерпретации в виде поля знаний экспертной системы [1].

Ранее нами предполагалось, что взаимодействие инженера по знаниям и эксперта осуществляется в форме непосредственного живого общения. Однако это далеко не единственная форма извлечения знаний. На рис. 1.3 приведена классификация методов извлечения знаний [1].

Рис. 1.3. Классификация методов извлечения знаний

Коммуникативные методы извлечения знаний охватывают методы и процедуры контактов инженера по знаниям с непосредственным источником знаний – экспертом, а текстологические включают методы извлечения знаний из документов (методик, пособий, руководств) и специальной литературы (статей, монографий, учебников).

1.6. Коммуникативные методы извлечения знаний

Коммуникативные методы извлечения знаний можно разделить на *активные* и *пассивные*.

Пассивные методы подразумевают, что ведущую роль в процедуре извлечения знаний играет эксперт, а инженер по знаниям только протоколирует высказывания эксперта во время его работы. В *активных методах* инициативой владеет инженер по знаниям, который контактирует с экспертом различными способами: в играх, диалогах, беседах за круглым столом. Следует отметить, что активные и пассивные методы могут чередоваться даже в рамках одного сеанса извлечения знаний.

Активные методы можно разделить на две группы, в зависимости от числа экспертов, отдающих свои знания. Если их число больше одного, то целесообразно, кроме индивидуальных контактов с каждым, применять и методы групповых обсуждений предметной области. Групповые методы обычно активизируют мышление участников дискуссий и позволяют выявлять различные аспекты их знаний. И тем не менее, индивидуальные методы извлечения знаний являются наиболее продуктивными, поскольку деликатная процедура передачи знаний не терпит лишних свидетелей.

Игровые методы широко используются в социологии, экономике, менеджменте и педагогике для подготовки руководителей, учителей, врачей и других специалистов. В игре человек раскрепощается и чувствует себя намного свободнее, чем в обычной трудовой деятельности.

На выбор метода извлечения знаний влияют три фактора:

- личностные особенности инженера по знаниям;
- личностные особенности эксперта;
- характеристика предметной области.

Существует три типа классификации людей по психологическим характеристикам: мыслитель (познавательный тип), собеседник (эмоционально-коммуникативный тип) и практик (практический тип) [1]. Мыслители ориентированы на интеллектуальную работу, теоретические обобщения. Собеседники – это общительные, открытые люди, готовые к сотрудничеству. Практики предпочитают действия разговорам, хорошо реализуют замыслы других.

Предметные области можно классифицировать следующим образом: хорошо, средне и слабо документированные.

Остановимся подробнее на пассивных методах извлечения знаний. В *пассивных методах* ведущая роль при извлечении знаний передается эксперту, а инженер по знаниям только фиксирует рассуждения эксперта во время принятия решений [1].

Пассивные методы извлечения знаний реализуются в виде наблюдений, анализа протоколов «мыслей вслух», лекций.

Во время *наблюдений* инженер по знаниям находится рядом с экспертом во время его профессиональной деятельности или имитации этой деятельности и фиксирует все действия эксперта, его реплики и объяснения. Полезна видеозапись всего процесса в реальном масштабе времени. Следует отметить, что именно этот метод извлечения знаний является наиболее «чистым» методом, исключающим навязывание инженером по знаниям своих представлений эксперту.

В случае *протоколирования «мыслей вслух»* эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено, представить всю цепочку своих рассуждений. Во время рассуждений эксперта все его слова, паузы и междометия протоколируются инженером по знаниям. Использование магнитофона не всегда возможно, т.к. его применение в некоторых случаях негативно действует на эксперта и разрушает атмосферу доверительности. Основной проблемой в этом методе является принципиальная сложность для любого человека объяснить, как он думает. Существует экспериментальное психологическое доказательство того, что люди не всегда в состоянии достоверно описывать мыслительные процессы. Автор теории фреймов М. Минский считает, что только как исключение, а не как правило, человек может объяснить то, что он знает [1].

Лекция является самым испытанным способом передачи знаний. Необходимо сформулировать эксперту тему и задачу лекции. Искусство ведения конспекта заключается в выделении главного, фиксировании фрагментов, записи только осмысленных предложений и помехоустойчивости инженера по знаниям. Рекомендуемая стандартная продолжительность лекции – 40–50 мин, и после перерыва (5–10 мин) вновь проводится лекция такой же продолжительности. Курс обычно содержит от 2 до 5 лекций [1].

Активные индивидуальные методы извлечения знаний являются самыми распространенными. К ним относятся анкетирование, интервью, свободный диалог, игры с экспертом.

В активных методах инженер по знаниям пишет сценарий и режиссирует сеансы извлечения знаний.

Анкетирование является наиболее стандартизованным методом. В этом случае инженер по знаниям составляет анкету, размножает ее и использует для опроса нескольких экспертов. Это основное преиму-

щество анкетирования. На основе опыта работы с анкетами, накопленного в социологии и психологии, выработаны следующие рекомендации для составителей анкет:

- анкета не должна быть монотонной и однообразной, чтобы не вызывать скуку и усталость; необходимо варьировать формы вопросов, менять тематику вопросов, вставлять вопросы-шутки и игровые вопросы;
- анкета должна быть приспособлена к языку экспертов.

Следует учитывать, что вопросы влияют друг на друга и поэтому последовательность вопросов должна быть хорошо продумана. Должен соблюдаться принцип оптимальной избыточности; как правило, в анкете необходимыми являются лишь часть вопросов, называемых контрольными, остальные должны быть минимизированы.

Под *интервью* понимается специфическая форма общения инженера по знаниям и эксперта. При этом инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области. Основное отличие интервью от анкетирования состоит в том, что оно позволяет аналитику опускать ряд вопросов в зависимости от ситуации, генерировать новые вопросы, изменять темп и разнообразить ситуацию общения.

Полезно включать в интервью следующие вопросы:

- контактные (ломающие лед между аналитиком и экспертом);
- буферные (для разграничения различных тем интервью);
- оживляющие память экспертов (для реконструкции отдельных случаев из практики);
- «провоцирующие» (для получения спонтанных, неподготовленных ответов) [1].

Основные характеристики вопроса, от которых зависит качество интервью, заключаются в следующем:

- язык вопроса (понятность, лаконичность, терминология);
- порядок вопросов (логическая последовательность и немонотонность);
- уместность вопросов (этика, вежливость).

Вопрос в интервью – это не только средство общения, но и способ передачи мысли и позиции инженера по знаниям. Вопрос представляет собой форму движения мысли, в нем ярко выражен момент перехода от незнания к знанию, от неполного, неточного знания к более полному и более точному [1]. Поэтому необходимо фиксировать в протоколах не только ответы, но и вопросы.

Свободный диалог – это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в котором нет жестко регламентирован-

ного плана и вопросника. Квалифицированная подготовка к диалогу помогает инженеру по знаниям стать сценаристом будущих сеансов и запланировать плавную процедуру извлечения знаний: от приятного впечатления в начале беседы переходят к профессиональному контакту через пробуждение интереса и завоевание доверия эксперта. Чтобы разговорить эксперта, желательно вначале рассказать ему о себе, о своей работе.

Следует отметить, что для определения профессиональной пригодности инженера по знаниям необходимо проведение предварительного психологического тестирования. Образец портрета инженера по знаниям перед серией свободных диалогов выглядит так: он должен выглядеть здоровым, спокойным, уверенным, внушать доверие, быть искренним, веселым, проявлять интерес к беседе, быть опрятно одетым и ухоженным [1].

Важно правильно выбрать ритм беседы: без больших пауз, чтобы эксперт не отвлекался, но и без гонки, чтобы оба участника не утомлялись. Умение чередовать темп беседы, напряжение и разрядку в беседе существенно влияет на результат.

К *активным групповым методам* извлечения знаний относятся ролевые игры, дискуссии за «круглым столом» с участием нескольких экспертов и «мозговые штурмы». Преимущество групповых методов состоит в одновременном извлечении знаний от нескольких экспертов и возможности генерации экспертами новых идей в процессе взаимодействия друг с другом.

Метод *круглого стола* предусматривает обсуждение какой-либо проблемы из выбранной предметной области, в котором принимают участие с равными правами несколько экспертов. Как правило, участники высказываются в определенном порядке, а затем переходят к живой свободной дискуссии. Число участников дискуссии составляет 3–7 человек.

Следует отметить, что поведение человека в группе иное, чем наедине с инженером по знаниям. Желание произвести впечатление на коллег будет приводить к изменению высказываний экспертов. Так, зачастую члены диссертационного совета на защите спрашивают не то, что им действительно интересно, а то, что демонстрирует их собственную компетентность.

Перед началом дискуссии ведущему необходимо:

- убедиться, что все правильно понимают задачу (сеанс извлечения знаний);
- установить регламент;
- четко сформулировать тему.

Дискуссии полезны и для самих экспертов, особенно для тех, кто знает меньше. Это отмечалось еще Эпикуром: «При философской дискуссии больше выигрывает побежденный – в том отношении, что он умножает знания» [1].

Активные групповые методы обычно применяют как дополнительные к традиционным индивидуальным методам для активизации мышления и поведения экспертов.

«Мозговой штурм» является одним из наиболее распространенных методов активизации творческого мышления. Впервые этот метод был использован в 1939 г. А. Осборном в США как способ получения новых идей в условиях запрещения критики. Установлено, что критика мешает творческому мышлению, поэтому основная идея штурма – это отделение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей [1].

Как правило, процесс штурма длится около 40 мин. Участникам (до 10 человек) предлагается высказывать любые идеи на заданную тему, при этом критика запрещена. Обычно высказывается более 50 идей. Регламент составляет до 2 мин на выступление. Самым интересным моментом штурма является достижение максимума числа гипотез, непроизвольно генерируемых участниками. Этот пик имеет теоретическое обоснование в работе З. Фрейда о бессознательном. При дальнейшем анализе всего лишь 10–15 % идей оказываются разумными и среди них встречаются оригинальные. Оценивает результаты обычно группа экспертов, не участвовавших в генерации гипотез.

Ведущий мозгового штурма (инженер по знаниям) должен свободно владеть аудиторией, не зажимать плохие идеи, т.к. они могут быть катализаторами хороших идей. Основной девиз штурма – «чем больше идей, тем лучше». Ход сеанса протоколируется или записывается на магнитофон.

Представляет интерес остановиться еще на одном методе извлечения знаний – *экспертных играх*. Игрой называется такой вид человеческой деятельности, который отражает (воссоздает) другие ее виды [1]. Понятие экспертной игры, или игры с экспертами в целях извлечения знаний, формировалось на основе трех разновидностей игр:

- деловых игр, широко используемых при подготовке специалистов и моделировании;
- диагностических игр;
- компьютерных игр, применяемых в обучении.

Следует отметить, что, в зависимости от количества участников, экспертные игры могут быть как групповыми, так и индивидуальными.

Под *деловой игрой* понимается эксперимент, где участникам предлагается производственная ситуация и они на основании своего жизненного опыта и специальных знаний и представлений принимают решения. На основе анализа решений вскрываются закономерности мышления участников эксперимента, что является полезным для получения знаний. *Деловая игра* превращается в *экспертную игру*, если ее участниками становятся эксперты.

Диагностическая игра – это деловая игра, применяемая для диагностики методов принятия решения в медицине. Эта игра возникла при исследовании способов передачи знаний от опытных врачей новичкам. Таким образом, в диагностических играх до сих пор исследовалась лишь одна предметная область – медицина [1].

Компьютерные игры классифицируются следующим образом:

1. Игры-действия (Action/Arcade games) требуют хорошего глазомера и быстрой реакции.
2. Трехмерные игры (3D Action games) – то же, что и предыдущее, но с активным использованием трехмерной графики.
3. Игры-симуляторы (Simulation games) базируются на моделировании реальной действительности и отработке практических навыков в вождении автомобиля, самолета, поезда, в выполнении функций авиадиспетчера и т. д.
4. Стратегические игры (Strategy games) требуют стратегического планирования и ответственности при принятии решений, например: развитие цивилизации, экономическая борьба. Особым классом стратегических игр являются военные игры (wargames).
5. Компьютерные реализации различных логических игр (Puzzles).
6. Приключенческие игры (Adventure/Quest) обладают разветвленным сценарием, красивой графикой и звуком. Управляя одним или несколькими персонажами, игрок должен правильно вести диалоги, разгадывать множество загадок и головоломок.
7. Ролевые игры (Role-playing games, RPG) являются распространенным жанром, берущим начало в английских настольных играх. Существует один или несколько персонажей, обладающих индивидуальными способностями и характеристиками. Им приходится сражаться с врагами, решать загадки. По мере выполнения этих задач у героев накапливается опыт и их способности и характеристики улучшаются.

Плодотворность моделирования реальных ситуаций в играх подтверждается сегодня практически во всех областях науки и техники. Они развивают логическое мышление, умение быстро принимать решение и вызывают интерес у экспертов.

1.7. Текстологические методы извлечения знаний

Группа *текстологических методов* объединяет методы извлечения знаний, основанные на изучении специальных текстов из учебников, монографий, статей, методик и других носителей профессиональных знаний. Задачу извлечения знаний из текстов можно сформулировать как задачу понимания и выделения смысла текста.

При извлечении знаний аналитику, интерпретирующему текст, приходится решать задачу декомпозиции этого текста на компоненты для выделения истинно значимых для реализации БЗ фрагментов. К таким компонентам можно отнести следующие: наблюдения, научные понятия, субъективные взгляды, общие места, заимствования.

Сложность интерпретации научных и специальных текстов заключается еще и в том, что любой текст приобретает смысл только в контексте. Под контекстом понимается окружение, в которое «погружен» текст. Различают микроконтекст и макроконтекст. *Микроконтекст* – это ближайшее окружение текста. Так, предложение получает смысл в контексте абзаца, абзац – в контексте главы и т. д. *Макроконтекст* – это вся система знаний, связанная с предметной областью (т. е. знания об особенностях и свойствах, явно не указанных в тексте).

На языке современного языкознания понимание – это формирование второго текста, т. е. семантической структуры.

Основными моментами процесса понимания текста являются:

- выдвижение предварительной гипотезы о смысле всего текста;
- определение значений непонятных слов (т. е. специальной терминологии);
- возникновение общей гипотезы о содержании текста;
- уточнение значения терминов и интерпретация отдельных фрагментов текста под влиянием общей гипотезы (от целого к частям);
- формирование смысловой структуры текста за счет установления внутренних связей между отдельными ключевыми словами и фрагментами, а также за счет образования абстрактных понятий, обобщающих конкретные фрагменты знаний;
- корректировка общей гипотезы относительно содержащихся в тексте фрагментов знаний (от частей к целому);
- принятие основной гипотезы.

При этом существенным является наличие как дедуктивной (от целого к частям), так и индуктивной (от частей к целому) составляющей процесса понимания. Благодаря этому удается при понимании текста учесть основные признаки текста: связность, цельность и законченность.

Центральным моментом процесса понимания является выделение опорных, ключевых слов, или смысловых вех, в тексте и дальнейшее их связывание в единую семантическую структуру [1].

При анализе текста выделяют два вида связей – *эксплицитные* (явные связи) и *имплицитные* (скрытые связи). Эксплицитные связи выражаются во внешнем дроблении текста, они делят текст на параграфы с помощью перечисления компонентов, вводных слов типа «во-

первых», «во-вторых», «однако» и т. д. Имплицитные связи между смысловыми вехами вызывают основное затруднение при понимании.

Семантическая структура текста образуется в сознании познающего субъекта с помощью знаний о языке, о мире, общих знаний о предметной области, которой посвящен текст. Таким образом, для адекватного понимания текста необходима предварительная подготовка.

Подготовкой к прочтению специальных текстов является выбор совместно с экспертами базового списка литературы, который постепенно введет аналитика в предметную область. В этом списке, как правило, содержатся учебники, фрагменты из монографий, популярные издания. После ознакомления с указанным списком целесообразно приступать к чтению специальных текстов.

Следует подчеркнуть, что процедура разбивки текста на части (смысловые группы), а затем сгущение, сжатие содержимого каждого смыслового блока в смысловую веху является основой для любого процесса понимания. Представление текста в виде набора ключевых слов, передающих основное содержание текста, является методологической основой для проведения текстологических процедур извлечения знаний.

В качестве ключевого слова может служить любая часть речи (существительное, глагол, прилагательное и т. д.) или их сочетание. Набор ключевых слов – это набор опорных точек, по которым развертывается текст при кодировании в память и осознается при декодировании.

Алгоритм извлечения знаний из текста можно представить в следующем виде:

1. Составление базового списка литературы для ознакомления с предметной областью и чтение по списку.
2. Выбор текста для извлечения знаний.
3. Первое знакомство с текстом (беглое прочтение); для определения значения незнакомых слов – консультации со специалистами или привлечение справочной литературы.
4. Формирование первой гипотезы о макроструктуре текста.
5. Внимательное прочтение текста с выписыванием ключевых слов и выражений, т. е. выделение смысловых вех (компрессия текста).
6. Определение связей между ключевыми словами, разработка макроструктуры текста в форме графа или сжатого текста (реферата).
7. Формирование поля знаний на основании макроструктуры текста.

ГЛАВА 2
МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

2.1. Представление знаний и выводы в экспертных системах

Характерная особенность ЭС, отличающая их от традиционных систем обработки информации, заключается в оперировании знаниями. Формализм описания такого рода информации определяется как представление знаний. Компонент, который использует для решения проблем знания экспертов, описанные в заранее выбранной для них форме представления, является механизмом вывода. В системах с БЗ, в том числе и ЭС, представление знаний является фундаментальным понятием. Решение о выборе способа представления знаний оказывает существенное влияние на любую их составную часть.

Можно сказать, что представлением знаний определяются возможности системы БЗ. И, наоборот, чтобы система обработки знаний отвечала определенным прикладным потребностям, должно быть создано соответствующее представление знаний. Поскольку представление знаний является средством описания знаний человека, то желательно, чтобы его описательные возможности были как можно выше. С другой стороны, если форма представления становится излишне сложной, то усложняется и механизм выводов, при этом не только затрудняется проектирование ЭС, но и возникает опасность потери достоверности выполняемых ею действий. В конечном итоге проектирование представления знаний предусматривает выработку всех этих условий, а затем и выбор решения на основе некоторого компромисса между ними.

Наиболее распространенными моделями представления знаний в экспертных системах являются:

- модель представления знаний средствами логики предикатов первого порядка;
- продукционная модель;
- фреймовая модель;
- модель представления знаний в виде семантической сети;
- модель представления знаний в виде доски объявлений;
- модель представления знаний в виде сценария;
- модель представления знаний на основе нечеткой логики;
- нейросетевая модель представления знаний [1–6, 8–16].

2.2. Модель представления знаний средствами логики предикатов первого порядка

Одним из наиболее важных способов представления знаний является представление знаний средствами логики предикатов первого порядка. Этот способ является основой языка Пролог [4, 9, 10].

В основе такого представления лежит язык математической логики, позволяющий формально описывать понятия математической логики и связи между ними. В естественном языке существуют грамматические правила, которые задают его синтаксис. Эти правила не связаны со значением слов, т. е. с семантикой языка. Основными компонентами естественного языка являются слова (существительные, прилагательные, глаголы, наречия, предлоги), предложения и контексты. Правила языка задают порядок следования слов в предложениях.

Язык, предназначенный для формализации знаний, должен иметь собственный синтаксис и располагать средствами для выражения связей между объектами реального мира. Указанному требованию удовлетворяет язык исчисления предикатов или логики первого порядка. Логика предикатов рассматривает отношения между утверждениями и объектами.

Предикатом называется функция, принимающая только два значения – *истина* и *ложь* – и предназначенная для выражения свойств объектов или связей между ними. Выражение, в котором утверждается или отрицается наличие каких-либо свойств у объекта, называется *высказыванием. Константы* служат для именования объектов предметной области. Логические предложения или высказывания образуют *атомарные формулы*.

Логический вывод осуществляется с помощью силлогизма (если из *A* следует *B*, а из *B* следует *C*, то из *A* следует *C*).

Логика предикатов является основой языка программирования Пролог, разработанного А. Колмероэ в Марсельском университете (Франция) в 1973 г. [9, 10].

Представляет интерес более детально остановиться на основных понятиях и определениях, применяемых в логике предикатов первого порядка [3].

Определение формальной системы. *Формальной системой* называют множество абстрактных объектов, в котором определены правила манипулирования множеством символов, обработанных синтаксическим образом, т. е. без учета смысла (семантики). Формальную систему составляют:

1) конечный алфавит (конечное множество символов);
2) процедура образования формул (или слов) формальной системы;
3) множество формул, называемых аксиомами;
4) конечное множество правил вывода, позволяющих получать конечное множество формул [3].

Эти правила могут быть представлены в виде U_1 и U_2 и … и $U_m \to$ W_1 и W_2 и … и W_m. Здесь U_i и W_j – формулы формальной системы, а стрелка читается как «влечет» или «следует».

Формальную систему иногда называют аксиоматической, формальной теорией или просто множеством формул. Алфавит предполагается конечным и его иногда называют словарем. Он содержит константы, переменные и операторы.

Процедура образования формул определяет синтаксическое и грамматическое построение символов из уже сформулированных символов. Она отличается от правил вывода.

Формальное доказательство (или просто доказательство) определяется как конечное множество формул M_1, M_2 ... M_n, таких, что каждая формула M_i либо является аксиомой, либо выводится с помощью правил вывода из предшествующих формул.

Формула *t* называется *теоремой,* если существует доказательство, в котором она является последней, т. е. $M_n = t$. В частности, всякая аксиома является теоремой. Тот факт, что формула является теоремой, обозначается как *t.* Правила вывода позволяют определить, является ли данная формула теоремой данной формальной системы. Различают два типа правил вывода. Правила первого типа применяются к формулам, рассчитываемым как единое целое (в этом случае их называют *продукциями, правилами продукции* или *продукционными правилами).* Правила второго типа могут применяться к любой отдельной части формулы, причем сами эти части являются формулами. Такие правила называют *правилами переписывания.*

Так, в математике правило вывода «$X < Y$ и $Y < Z$ *влечет* $X < Z$» применяется к формуле как к целому. Это продукция с двумя посылками. Слово «влечет» часто заменяется стрелкой $\to$. В отличие от предыдущего правила, $x - x = 0$ истинно при любом подвыражении *x*. Оно является правилом переписывания, и в этом случае для обозначения слова «влечет» используется специальная стрелка $\mapsto$.

Пример:

1. Алфавит: *T, S, A*.
2. Формула: любая последовательность символов алфавита.
3. Аксиома: *TA*.
4. Правила вывода:
   a) $tA \to tAS$ (продукция),
   b) $Tt \to Ttt$ (продукция),
   c) $AAA \mapsto S$ (переписывание),
   d) $SS \mapsto$ (переписывание).

Правило *a* является продукцией и применяется только тогда, когда последняя буква теоремы есть *A*. Таким образом, из теоремы «*TASTA»* можно вывести *«TASTAS».* Следует отметить, что символ *t* не принадлежит формальной системе и играет роль некоторого слова. Так, правило *b* позволяет из теоремы *«TSA»* вывести теорему *«TSASA».* Правило *c* позволяет переход, например, от *«TSAAAST»* к *«TSSST».* Правило *d* заменяет в *«TASSSSV»* каждые два *S* пробелом и в результате получаем *«TAV».*

Формальные системы предназначены для получения умозаключений без рассмотрения смысла обрабатываемых заключений, т. е. элементы, на основании которых делается умозаключение, могут быть произвольным образом заменены другими по смыслу элементами.

Например, в абстрактной модели:

«Любой *X* есть *Y.*

Если *Z* есть *X,*

то *Z* есть *Y*.»

используются переменные *X, Y, Z.* Слова-связи «если», «то», «или» и другие называются *операторами*.

Синтаксис языка предикатов первого порядка. *Предикатом,* или *логической функцией,* называется функция от любого числа аргументов, принимающая истинностные значения: *И* (истина – 1) и *Л* (ложь – 0). Аргументы принимают значения из произвольного, конечного или бесконечного множества *D,* называемого предметной областью. Предикат от *n* аргументов называют *n*-местным предикатом [3].

Предикат *F(x),* определенный на предметной области *D*, задает определенное свойство элементам множества *D* и интерпретируется как высказывание «*x* обладает свойством *F*», причем *F* принимает значение *И,* если это высказывание истинно, и значение *Л*, если оно ложно. Предикат $F(x_1, x_2 ... x_N)$ задает отношение между элементами x_1, $x_2 \ldots x_N$ и интерпретируется как высказывание «x_1, $x_2 \ldots x_N$ находятся между собой в отношении *F*». Пусть, например, *D* – множество натуральных чисел. Тогда предикат *F(x)* может обозначать, что «*x* – четное число» или «*x* – нечетное число», а предикат *G(x, y)* – «*x* больше *y*» или «*x* делится на *y*» и т. д.

Алфавит языка предикатов первого порядка включает множество следующих символов:

- разделители – запятая, открывающая и закрывающая скобки;
- константы, обозначаемые строчными буквами или соединением таких букв, например «друг»;
- переменные, обозначаемые прописными буквами, например: *X*, *АДРЕС*;

- предикаты, обозначаемые прописными буквами, например: *P, Q, БОЛЬШЕ*;
- функции, устанавливающие зависимость и отображающие значения одной предметной области в значения другой (или той же), *n*-местные функции могут служить аргументами предиката. Функции будем обозначать строчными буквами *f*, *g*;
- логические операции:
  1) «$\neg$» (отрицание или дополнение). Высказывание «$\neg A$» читается «не *A*». Оно истинно *(И),* если высказывание *A* – ложно (*Л*);
  2) «$\wedge$» (конъюнкция). Высказывание «$A \wedge B$» читается «*A* и *B*». Оно истинно в том случае, когда истинно как *A,* так и *B*;
  3) «$\vee$» (дизъюнкция). Высказывание «$A \vee B$» читается «*A* или *B*». Оно истинно, если истинно хотя бы одно из высказываний;
  4) «$\rightarrow$» (импликация). Высказывание «$A \rightarrow B$» читается «если *A,* то *B*». Оно ложно в том и только в том случае, если *A* истинно, а *B* ложно;
  5) «$\leftrightarrow$» (эквивалентность). Высказывание «$A \leftrightarrow B$» читается «*A* тогда и только тогда, когда *B*». Оно истинно тогда и только тогда, когда *A* и *B* имеют одно и то же истинностное значение;
- кванторы:
  1) «$\exists$» (квантор существования). Высказывание $\exists A$ читается «существует *A*»*;*
  2) «$\forall$» (квантор общности). Высказывание $\forall A$ читается «для любого *A*».

Пропозициональной формой, или *формулой алгебры логики,* называют всякое высказывание, составленное из некоторых исходных высказываний посредством логических операций. Другими словами, если *F* и *G* – пропозициональные формы, то $\neg F$, $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$ и $(F \leftrightarrow G)$ – пропозициональные формы.

Определение формулы как основного объекта в логике предикатов включает понятие «терм».

Терм – выражение, включающее константы, переменные или *n*-местные функции $f(t_1 \ldots t_N)$, где $t_1 \ldots t_N$ – термы. Например, $f(X, Y)$, вес (b) – термы.

Атом, или *элементарная (атомная) формула,* – это выражение, включающее константы, переменные, функции и предикаты. Таким образом, если *P* – *N*-местный предикат, а $t_1 \ldots t_N$ – термы, то $P(t_1 \ldots t_N)$ – атом. Например, выражения *P*(*X*, зеленый), *ABC* являются атомами.

Формула, или *правильно построенная формула* (ППФ), определяется следующим образом:

всякий атом есть ППФ;

если *G* и *H* – ППФ, а *X* – переменная, тогда

$(\neg H)$, $(G \vee H)$, $(G \wedge H)$, $(\exists X)G$, $(\forall X)H$ – ППФ.

Примерами ППФ являются:

$$P(a, X, f(Y, a, b)), G(X, Y) \to F(a, b), (P(a) \to (G(b) \wedge \neg F(a))).$$

Выражение «*первого порядка*» во фразе «*исчисление предикатов первого порядка*» связано с определением ППФ, в которых запрещается квантифицировать символы предикатов и функций. Например, $(\forall P)P(a)$ не является ППФ логики предикатов первого порядка.

На практике ППФ используется для представления знаний. Например, ППФ $(\forall X)(M(X) \to F(X))$ может выражать «все матери есть женщины», условившись, что *M(X)* означает: «*X* есть мать» и что *F(X)* означает: «*X* есть женщина».

Правилом вывода называют процедуру, которая из одной или нескольких ППФ производит другие ППФ. Например, правило вывода: G и $(G \to H)$ производит одну ППФ *H;* из ППФ $(\forall X)G(X)$ и любой константы «*a*» получают ППФ $G(a)$, при этом значения X в G заменяются на «*a*». Исходные ППФ называют *аксиомами,* а ППФ, полученные из правил вывода, называют *теоремами.*

Семантика языка предикатов первого порядка. *Интерпретация формул.* Формула имеет определенный смысл, т. е. обозначает некоторое высказывание, если существует какая-либо интерпретация. *Интерпретировать формулу* – это значит связать с ней определенное непустое множество *D,* т. е. конкретизировать предметную область, называемую также *областью интерпретации* [3] и указать:

- для каждой константы в формуле – конкретный элемент из *D*;
- для каждой *n*-местной функциональной буквы в формуле – конкретную *n*-местную функцию на *D*;
- для каждой n-местной предикатной буквы в формуле – конкретное отношение между *n* элементами из *D.*

Пример:

Рассмотрим атом $G(f(a, b), g(a, b))$ и следующую интерпретацию:

D – множество действительных чисел;

$$a = 2, b = 3;$$

f – функция сложения $f(a, b) = a + b$;

g – функция умножения $g(a, b) = a \cdot b$;

G – отношение «не меньше».

При такой интерпретации приведенная ниже формула обозначает высказывание «сумма $2 + 3$ не меньше произведения $2 \cdot 3$». Это утвер-

ждение неверно и поэтому $G(f(a,b),g(a,b)) = Л$. Если видоизменить интерпретацию, приняв $b = 1$ или $b = 2$, то $G(f(a,b),g(a,b)) = И$.

Свойства правильно построенных формул. При заданной интерпретации значения истинности ППФ можно вычислить по правилам, объединенным в таблице истинности.

Если F и G – любые две ППФ, то значения истинности составного выражения, построенного из этих ППФ, даются таблицей истинности (табл. 2.1) [3].

Таблица 2.1

Таблица истинности

F	G	$\neg F$	$F \vee G$	$F \wedge G$	$F \rightarrow G$	$F \leftrightarrow G$
И	И	Л	И	И	И	И
Л	И	И	И	Л	И	Л
И	Л	Л	И	Л	Л	Л
Л	Л	И	Л	Л	И	И

Принцип резолюций. Аппарат логики предикатов используется для представления задачи в виде теоремы: формула H логически следует из формулы G, т. е. $G \rightarrow H$. Доказательство этой теоремы состоит в том, чтобы показать, что каждая интерпретация, удовлетворяющая G, удовлетворяет и H. С целью упрощения доказательства теоремы все формулы представляются в виде дизъюнкции литералов.

Литералом называется атом или его отрицание. Формулу, представляющую собой дизъюнкцию литералов, называют *предложением* (или *дизъюнктом*). Любую ППФ исчисления предикатов первого порядка можно преобразовать во множество предложений [3, 9].

К достоинствам логических моделей относятся единственность теоретического обоснования и возможность реализации системы формально точных определений и выводов. Но попытки представления неформализованных знаний эксперта в системе строгой логики не всегда удается осуществить. Это объясняется нечеткой структурой человеческой логики [2, 3].

2.3. Представление знаний продукционными правилами

Продукционная модель, или модель, основанная на правилах, позволяет представить знания в виде предложений типа «**Если** (условие), **то** (действие)». Под «условием» понимается некоторое предложение-образец, по которому осуществляется поиск в БЗ, а под «действием» – действие, выполняемое при успешном исходе поиска. Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения – к данным). *Данные* – это ис-

ходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, перебирающий правила из продукционной базы знаний [1].

Простота и наглядность этого способа обусловили его применение во многих системах. Системы обработки знаний, использующие представление знаний продукционными правилами, получили название *продукционных систем* [11]. В состав продукционной системы входят база правил, база данных и интерпретатор правил (рис. 2.1).

База правил – это область памяти, которая содержит базу знаний – совокупность знаний, представленных в форме правил вида ЕСЛИ…ТО; *база данных* (БД) – это область памяти, содержащая фактические данные (факты), которые описывают вводимые данные и состояния системы.

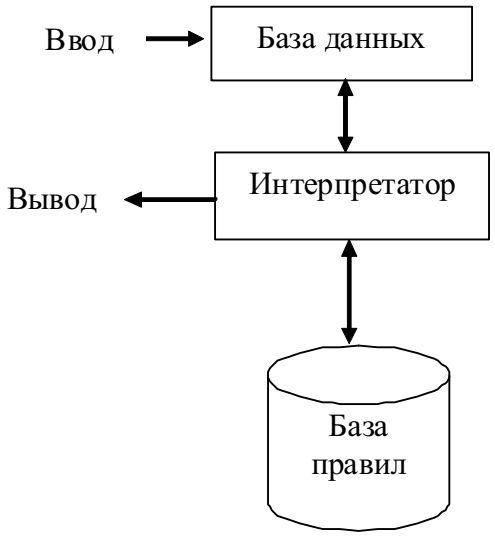


Рис. 2.1. Структура продукционной системы

БД у различных систем имеют различную форму, однако все они могут быть описаны как группа данных, содержащих имя данных, атрибуты и значения атрибутов. Интерпретатор представляет собой механизм вывода, и он является тем компонентом системы, который формирует заключения, используя базу правил и базу данных.

Рассмотрим выводы, основанные на продукционных правилах. Механизм, реализованный сегодня как средство выводов в продукционной системе, в принципе несложный. Он имеет функции поиска в БЗ, последовательного выполнения операций над знаниями и получения заключений. Причем существует два способа проведения таких заключений – прямые выводы и обратные выводы. В прямых выводах осуществляется продвижение к поставленной цели с последовательным применением правил к данным (фактам), которые принимаются за отправную точку.

В прямых выводах выбирается один из элементов данных, содержащихся в БД, и если при сопоставлении этот элемент согласуется с посылкой правила, то из правила выводится соответствующее заключение и помещается в БД, или же исполняется действие, определяемое правилом, и соответствующим образом изменяется содержимое базы данных. Зачастую такие выводы называют выводами, управляемыми данными, или восходящими выводами, когда последовательно выводятся новые результаты, начиная с уже известных данных.

Выводы, при которых процесс движется в направлении от поставленной цели к отправной точке, являются обратными. Они называются также нисходящими, или выводами, ориентированными на цель. Процесс нисходящих выводов начинается от поставленной цели. Если эта цель согласовывается с заключением правила, то посылка правила принимается за подцель или гипотезу. Этот процесс продолжается до тех пор, пока не будет получено совпадение подцели или гипотезы с полученными данными.

Нельзя категорически ответить на вопрос, какой же из этих способов лучше, поскольку это зависит от той проблемы, для которой они используются. У систем, к которым предъявляется требование высокой универсальности, необходимо наличие обоих способов вывода.

Обычно в знаниях, представленных продукционными правилами, одновременно присутствуют несколько условий в посылке, однако имеется одно единственное заключение. Требования к механизму выводов определяется из условия согласования знаний, используемых для выводов, причем обработка станет особенно легкой, если вывод обходится однократным согласованием. В этом смысле нисходящие выводы являются простыми выводами, однако, существуют задачи, для которых не так просто дать четкое описание цели. Например, в системах медицинской диагностики, когда диагноз ставится по симптомам, трудно получать нисходящий вывод, если неизвестна цель.

Продукционная модель чаще всего применяется в промышленных ЭС. Она характеризуется наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Продукционные системы принципиально весьма просты, и успехи самых первых ЭС DENDRAL и MYCIN, использующих продукционные правила, внесли огромный вклад в становление инженерии знаний.

ЭС DENDRAL (1965) – родоначальник инженерии знаний. Непосредственной целью ее создания явилась оценка структуры неизвестных химических соединений, исходя из экспериментальных результатов масс-спектрометрии (позднее сюда вошли также результаты ядерного магнитного резонанса). DENDRAL реализует процедуру генерации всех хими-

ческих структур-кандидатов, которые потенциально согласуются с полученными исходными данными, и осуществляет выбор из них структуры, точно согласуемой с этими данными. В таком процессе и используются знания экспертов, позволяющие исключать из рассмотрения неподходящие структуры и быстро отыскивать правильные результаты. При этом поставленная цель может быть достигнута тем эффективнее, чем значительнее по объему будут знания, которыми располагает система. Система DENDRAL была высоко оценена с точки зрения становления инженерии знаний, а принятый в ней метод одновременного выполнения операций генерации – тестирования идентичен алгоритмам, реализуемым человеком в процессе проектирования, и весьма важен как способ создания систем, целью которых является использование знаний.

В числе первых ЭС преобладали системы для медицинской диагностики. Это объясняется тем, что знания клинической диагностики в значительной степени определяются суждениями практического опыта врачей, и хотя эти знания являются процедурными, они не описываются формальными методами и для них необходим способ представления, приближающийся к описанию на естественном языке. С учетом всего этого, медицинская диагностика рассматривалась как весьма подходящий объект при разработке ЭС.

MYCIN представляет собой характерный пример системы, использующей продукционные правила. Представление знаний в MYCIN простое по форме и легко понимаемое, однако, описание всех языковых выражений в форме ЕСЛИ…ТО не всегда простое. В системе MYCIN выводы осуществлялись в форме, зависящей от содержания знаний предметной области. Появление ЭС MYCIN инициировало создание ряда систем, например PUFF (для диагностики легочных заболеваний).

В настоящее время имеется большое число программных средств, реализующих продукционный подход (язык OPS-5; «оболочки» ЭС – EXSYS Professional, Kappa, ЭКСПЕРТ, ЭКО, инструментальные системы ПИЭС (Хорошевский В.Ф., 1993) и СПЭИС (Ковригин О.В., Перфильев К.Г., 1988), а также промышленных ЭС на его основе (например, ЭС, созданных средствами G2 (Попов Э.В., 1996)) [1].

2.4. Модель представления знаний в виде фреймов

Продукционные системы, базы знаний которых насчитывают сотни правил, отнюдь не считаются чем-то необычным. При такой сложности системы для инженера знаний процесс обновления состава правил и контроль связей между ними становится весьма затруднительным, поскольку добавляемые правила могут дублировать имеющиеся или вступать с ними в противоречие. Для выявления подобных фактов можно использовать про-

граммные средства, но включение их в работу системы приводит к еще более тяжелым последствиям – потере ею работоспособности, т. к. в этом случае инженер знаний теряет представление о том, как взаимодействуют правила. При этом количество опосредованных родовидовых связей между понятиями резко возрастает, и инженеру знаний трудно их контролировать.

Представление знаний, основанное на фреймах, является альтернативным по отношению к системам продукции: оно дает возможность хранить родовидовую иерархию понятий в БЗ в явной форме.

Термин «фрейм» (от английского frame – «каркас» или «рамка») был предложен Марвином Минским в 1979 г. для обозначения структуры данных при восприятии пространственных сцен. Эта модель имеет психологическое обоснование. *Фрейм* – это абстрактный образ для представления некоего стереотипа восприятия [1].

Во фреймовой системе единицей представления знания является объект, называемый фреймом. Он является формой представления некоторой ситуации, которую можно (или целесообразно) описывать некоторой совокупностью понятий и сущностей. В качестве идентификатора фрейму присваивается имя. Это имя должно быть единственным во всей фреймовой системе. Фрейм имеет определенную внутреннюю структуру, состоящую из множества элементов, называемых слотами, которым также присваиваются имена. Каждый слот, в свою очередь, представляется определенной структурой данных. Иногда слот включает компонент, называемый фасетом, который задает диапазон или перечень его возможных значений. Фасет указывает также граничные значения заполнителя слота (например, максимально допустимое число братьев).

Помимо конкретного значения, в слоте могут храниться процедуры и правила, которые вызываются при необходимости вычисления этого значения.

Совокупность фреймов, моделирующая какую-либо предметную область, представляет собой иерархическую структуру, в которую фреймы соединяются с помощью родовидовых связей (рис. 2.2). На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов.

Фреймы обладают способностью наследовать значения характеристик своих родителей, находящихся на более высоком уровне иерархии. Так, фрейм АФРИКАНСКИЙ СЛОН наследует от фрейма СЛОН значение СЕРЫЙ характеристики ЦВЕТ.

Значения характеристик фреймов могут передаваться по умолчанию фреймам, находящимся ниже их в иерархии, но если последние содержат собственные значения данных характеристик, то в качестве истинных применяются именно они [4]. Это обстоятельство позволяет

легко учитывать во фреймовых системах различного рода исключения. В частности, во фрейме АЗИАТСКИЙ СЛОН значением слота ЦВЕТ будет КОРИЧНЕВЫЙ, а не СЕРЫЙ, которое могло бы в нем находиться, если бы предпочтение при выборе отдавалось не собственному значению, а наследуемому от фрейма СЛОН (рис. 2.3).

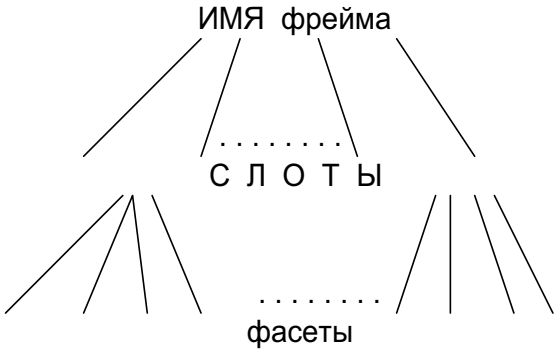


Рис. 2.2. Иерархическая структура фрейма

Различают статические и динамические системы фреймов. В системах первого типа фреймы не могут быть изменены в процессе решения задачи, в системах второго типа это допустимо.

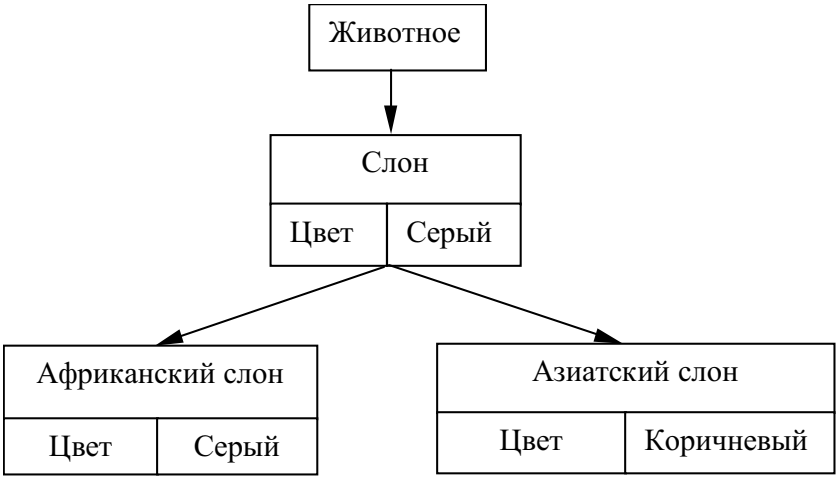


Рис. 2.3. Пример наследования свойств

О системах программирования, основанных на фреймах, говорят, что они являются объектно-ориентированными системами. Каждый фрейм соответствует некоторому объекту предметной области, а слоты содержат данные, описывающие этот объект, т. е. в слотах находятся значения признаков объекта. Фрейм может быть представлен в виде списка свойств, а если использовать средства БД, то в виде записи (кортежа).

Выводы во фреймовой системе исполняются обменами сообщений между фреймами.

Наиболее ярко достоинства фреймовых систем представления знаний проявляются в том случае, если родовидовые связи изменяются не часто и предметная область насчитывает немного исключений. Во фреймовых системах данные о родовидовых связях хранятся явно, т. е. так же, как и знания всех других типов. Значения слотов представляются в системе в единственном экземпляре, поскольку включаются только в один фрейм, описывающий наиболее общее понятие из всех тех, которые содержат слот с данным именем. Такое свойство систем фреймов дает возможность уменьшить объем памяти, необходимый для их размещения в компьютере. Еще одно достоинство фреймов состоит в том, что значение любого слота при необходимости может быть вычислено с помощью соответствующих процедур или найдено эвристическими методами. Для этого инженер знаний должен заранее разработать все требуемые процедуры и эвристические методы, чтобы включить их в систему на этапе ее проектирования.

Как недостаток фреймовых систем, следует отметить их относительно высокую сложность, что проявляется в снижении скорости работы механизма вывода и в увеличении трудоемкости внесения изменений в родовидовую иерархию.

В настоящее время имеется лишь небольшое число ЭС, где для представления знаний служат фреймы. Самая популярная среди них – KEE, созданная компанией Intel licorp. Система KEE имеет развитые встроенные графические средства объяснения и тестирования как динамических процессов, происходящих в ходе решения задачи, так и статического состояния БЗ.

В качестве средств приобретения знаний используется интеллектуальный редактор БЗ. Система KEE является экспертной оболочкой, эффективным средством автоматизации проектирования ЭС, ориентированным на использование в различных приложениях. Программное обеспечение этой системы выполнено на языке ЛИСП и требует применения специальной аппаратуры. Стоимость самих программ приблизительно 60 000 $, а к ней еще нужно добавить стоимость ЛИСП-машины, на которой они выполняются, т. е. еще 50 000–100 000 $. Стоимость же полного комплекта системы составляет не менее 250 000 $ [4].

2.5. Представление знаний в виде семантической сети

Семантические сети, по мнению специалистов, реализуют наиболее общий способ представления знаний. *Семантическая сеть* (semantic network) представляет знания в виде графа, узлы которого соответствуют фактам или понятиям, а дуги – отношениям между понятиями. Как узлы, так и дуги обычно имеют метки. *Граф* представляет собой множество вершин и множество дуг, соединяющих некоторые пары вершин.

Размеченный граф для каждой вершины содержит дескрипторы (метки), благодаря которым вершины графа отличаются между собой. Для графа пространства состояний дескрипторы идентифицируют состояния в процессе решения задачи. Метки дуг в семантических сетях применяются для задания именованных отношений [9].

В *ориентированном графе* для каждой дуги приписано определенное направление, указанное стрелкой. *Путь на графе* – это последовательность дуг, соединяющая соседние вершины. Две вершины называются связными, если существует путь, содержащий эти вершины. Если путь включает некоторую вершину более одного раза, то он содержит *петлю* или *цикл*.

Корневой граф содержит одну выделенную вершину (*корень*), от которой существует путь к любой вершине графа. Корень обычно располагается в верхней части рисунка над остальными вершинами. Корень графа не имеет родителей. Вершина, не имеющая потомков, называется *концевой вершиной*. Примером такого графа является генеалогическое дерево.

Деревом является граф, в котором существует единственный путь между любыми двумя вершинами. Отношения между вершинами для корневого дерева описываются понятиями *родителя*, *потомка* и *вершин-братьев* (имеющих общих родителей). Вершина называется *предком* всех вершин, расположенных после нее и *потомком* всех вершин, расположенных на пути к ней.

Следует отметить, что графы использовались в психологии Сэлзом (1913) для представления иерархии понятий и наследования свойств. Первые компьютерные реализации семантических сетей, предназначенные для понимания естественных языков, появились в начале 1960-х гг. в системах автоматического перевода текстов [9].

Шенком и Ригером была создана теория концептуальной зависимости для моделирования семантических структур естественного языка. В их теории рассматриваются четыре типа примитивов, на основе которых определяется смысл выражений [9]:

ACT – действия (actions);

PP – объекты (pictures producers);

AA – модификаторы действий (action aiders);

PA – модификаторы объектов (picture aiders).

При этом предполагается, что каждое из действий должно приводить к созданию одного или нескольких примитивов ACT. Далее приведены примитивы, выбранные в качестве основных компонентов действия:

- ATRANS – передавать отношения (давать);
- PTRANS – передавать физическое расположение объекта (идти);
- PROPEL – применять физическую силу к объекту (толкать);

- MOVE – перемещать часть тела (владельцем);
- GRASP – захватывать объект (исполнителем);
- INGEST – поглощать объект (есть);
- EXPEL – издавать звуки (кричать);
- MTRANS – передавать ментальную информацию (сказать);
- MBUILD – создавать новую ментальную информацию (решать);
- CONC – осмысливать идею (думать);
- SPEAK – производить звук (говорить);
- ATTEND – слушать.

Указанные примитивы используются для определения отношений концептуальной зависимости, которые описывают смысловые структуры (например, связи объектов и значений). Под отношениями концептуальной зависимости имеются в виду концептуальные синтаксические правила, которые выражают семантические связи в соответствии с грамматикой языка. Эти соотношения могут использоваться для конструирования внутреннего представления предложения на естественном языке. Ниже приведен пример некоторых концептуальных зависимостей [9]:

$PP \Leftrightarrow ACT$ – указывает, что исполнитель действует;

$PP \leftrightarrow PA$ – указывает, что объект имеет определенный признак;

$ACT \leftarrow PP$ – означает объект действия.

На основе приведенных зависимостей можно построить следующие предложения:

$PP \Leftrightarrow ACT$ $\quad$ $John \Leftrightarrow PTRANS$ – Джон бежал;

$PP \leftrightarrow PA$ $\quad$ $John \leftrightarrow height\ (> average)$ – Джон высокий;

$ACT \leftarrow PP$ $\quad$ $John \Leftrightarrow PROPEL \leftarrow cart$ – Джон двигал тележку.

В теории концептуальной зависимости проводится построение *канонической формы* смысла выражений. При этом все выражения, имеющие один и тот же смысл, будут представлены *синтаксически идентичными*, а не только семантически эквивалентными графами. Каноническое представление является средством упрощения выводов, требуемых для понимания. Например, на основе сопоставления графов концептуальной зависимости можно сделать вывод о том, что два выражения означают одну и ту же сущность.

К недостаткам рассматриваемой теории относятся трудности, связанные с алгоритмизацией процесса преобразования выражений в каноническую форму. Тем не менее, можно сделать вывод, что теория концептуальной зависимости является завершенной моделью семантики естественного языка и широко применяется в настоящее время.

Следует отметить, что, как и в системе, основанной на фреймах, в семантической сети могут быть представлены родовидовые отноше-

ния, которые позволяют реализовать наследование свойств от объектов родителей. Это обстоятельство приводит к тому, что в этом случае семантические сети приобретают все достоинства и недостатки представления знаний в виде фреймов [4].

2.6. Модель доски объявлений

В ряде случаев реализуются ситуации, когда решаемая проблема конструируется из нескольких частных проблем с различными свойствами. Эти частные проблемы могут быть зависимыми. Например, разделить частные проблемы невозможно при наличии между ними слабой связи, когда результатом решения одной частной проблемы инициируется решение следующей.

Одной из подобных проблем является проблема распознавания речи. Она заключается в понимании разговора и состоит из нескольких, различающихся по свойствам частных проблем, начиная с этапа обнаружения сигналов с детекторов и вплоть до понимания на высоком уровне абстракции, когда последовательно проводится вычленение фонем, распознавание слов, построение предложения и понимание предложения.

Существует система HEARSAY–II, которая разработана в американском Университете Карнеги–Меллона и ориентирована на решение подобного рода проблем. Характерной особенностью ее построения является использование *модели доски объявлений* [11].

В этой модели для каждой отдельной проблемы имеется соответствующее множество знаний. При этом все множества знаний через общую рабочую область памяти, называемую доской объявлений, управляются так, что все знания используются согласованно, как единое целое. Такие отдельные множества знаний называются обычно источником знаний (ИЗ). Каждый ИЗ сам по себе строится как продукционная система. В качестве примера модели доски объявлений на рис. 2.4 показана структура системы HEARSAY–II [11].

Здесь каждая частная проблема связана только со смежными проблемами. Правила преобразования частных проблем представлены продукционными правилами в форме ЕСЛИ…ТО.

В системе HEARSAY–II доска объявлений состоит из ряда иерархических уровней, и данные на каждом уровне сохраняются до тех пор, пока обработка не достигнет этого уровня. Данные обрабатываются последовательно, начиная с обработки акустических параметров на самом нижнем уровне, далее они проходят уровень вычлененных фонем, находящийся над ним уровень слогов, построенных из фонем, затем более высокий уровень отдельных слов и уровень синтаксических единиц предложений.

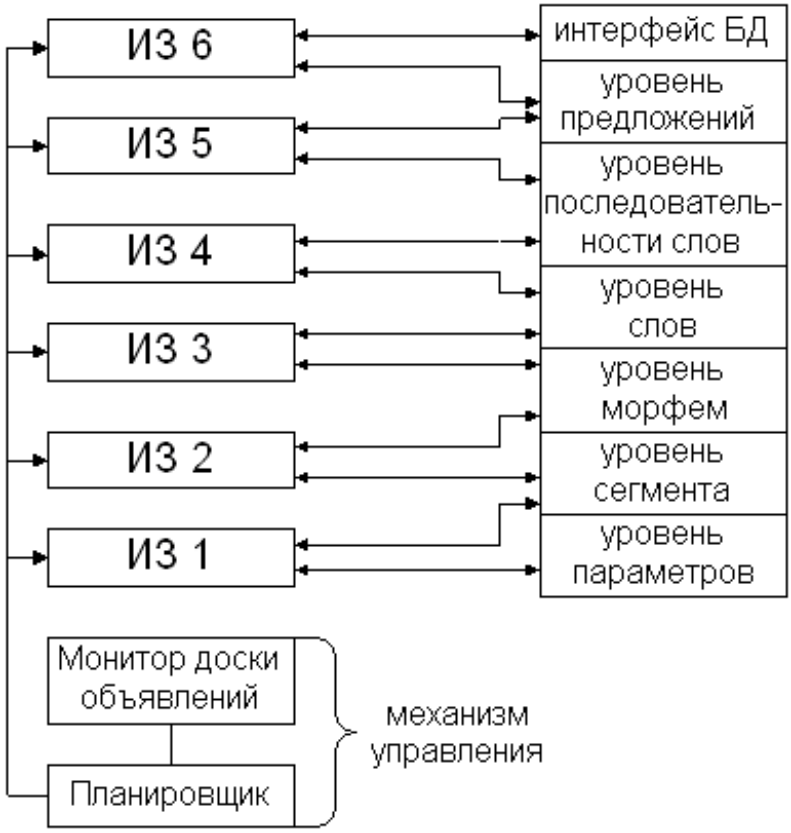


Рис. 2.4. Модель доски объявлений на примере системы HEARSAY–II

Результаты обработки на каждом уровне показывают только одну из возможностей понимания, и в этом смысле они называются гипотезами. Когда, используя определенную гипотезу, конструируется гипотеза для более высокого уровня, то она переносится на уровень, на который переходит и обработка данных. Например, если удается из фонем образовать отдельное слово, имеющее смысл, то эта гипотеза считается в дальнейшей обработке достоверной. Некоторый источник знаний, имеющий отношение к гипотезе соответствующего уровня, принимает гипотезу нижнего уровня за посылку и с помощью механизма восходящих выводов генерирует гипотезу для более высокого уровня. Приняв гипотезу верхнего уровня как заключение, этот ИЗ осуществляет нисходящий вывод и проводит проверку результата обработки, передавая его как обратную связь на более низкий уровень.

Следует отметить, что все ИЗ могут работать независимо, асинхронно и параллельно. Для эффективной реализации процесса обработки знаний в целом система содержит механизм управления запуском ИЗ в соответствии с текущей ситуацией.

В рассматриваемой модели действия механизма управления чрезвычайно важны: он анализирует сложившуюся на доске объявлений ситуацию, выносит решение о том, какой уровень обработки должен об-

ладать наивысшим приоритетом. Механизм управления осуществляет планирование запуска источников знаний. Считается, что в системе HEARSAY–II управляющая часть соответствует интеллектуальным возможностям и методам мышления человека.

Представленная на рис. 2.4 структура, которую имеет система HEARSAY–II, соответствует сложности решаемых проблем. К таким проблемам можно отнести не только понимание речи, но и автоматическое программирование, автоматическое планирование, планирование эксперимента, понимание текстов, обработку изображений.

Все эти проблемы непросты. Они требуют более универсальной по функциям системы. Для доски объявлений необходимо использовать структуру данных общего вида, не зависящих от проблемы. Это сделает еще более сложным все управление, которое в результате будет полностью возложено на самого пользователя. Такие системы, как AGE, ART, ESHELL, унаследовавшие идеи HEARSAY–II, являются еще более универсальными системами, чем она [11].

2.7. Модель представления знаний в виде сценария

Особую роль в системах представления знаний играют стереотипные знания, описывающие стандартные ситуации реального мира. Такие знания позволяют восстанавливать информацию, пропущенную в описании ситуации, предсказывать появление новых фактов, которые можно ожидать в данной ситуации, устанавливать смысл происхождения с точки зрения более ситуативного контекста.

Для описания стереотипного знания используются различные модели. Среди них распространенными являются сценарии. *Сценарием* называется формализованное описание стандартной последовательности взаимосвязанных фактов, определяющих типичную ситуацию предметной области. Это могут быть последовательности действий или процедур, описывающие способы достижения целей действующих лиц сценария (например, обед в ресторане, командировка, полет самолета, поступление в вуз). В интеллектуальных системах сценарии используются в процедурах понимания естественно-языковых текстов, планирования поведения, обучения, принятия решений, управления изменениями среды и др.

Впервые понятие сценария было введено Шенком и Абельсоном при разработке новых средств понимания истории. Сценарии в их системе понимания представлялись в виде фреймоподобных структур и использовались для связывания событий истории. Каждый сценарий состоял из набора слотов и их значений, описывающих роли, причины и последовательности сцен, которые, в свою очередь, являлись последовательностью определенных действий.

Слот «роль» задавал исполнителей сценария, слот «цель» – мотивировку предпринимаемых действий. Каждая последовательность действий в сценах обладает свойством каузальных цепочек: всякое предшествующее действие создает условия для совершения последующего действия.

Сценарии используются для пополнения знания о ситуации. Так называется процедура обогащения входной информации сведениями, хранящимися в памяти системы. Сценарий представляется некоторой сетью, вершинам которой соответствуют факты, а дугам – связи, описывающие отношения специального типа. Например: «причина – следствие», «цель – подцель», «часть – целое» (рис. 2.5).

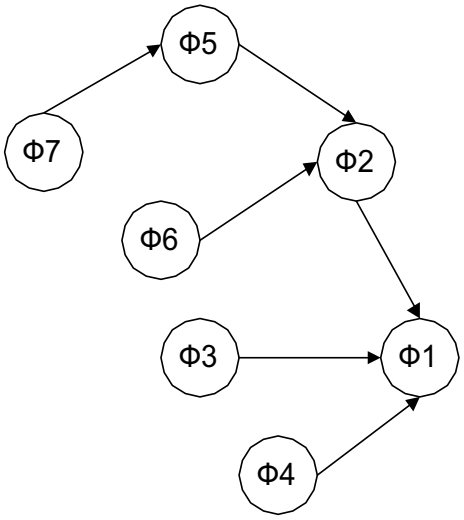


Рис. 2.5. Пример сценария, в котором в качестве связей между вершинами сети выступает причинно-следственное отношение

Вершины сети задают следующие факты:

Ф1 – станок простаивает;

Ф2 – на рабочем месте нет рабочего;

Ф3 – станок неисправен;

Ф4 – в цехе нет заготовок;

Ф5 – обеденный перерыв;

Ф6 – рабочий покинул станок;

Ф7 – рабочий находится в столовой.

Каузальные сценарии разработаны для представления проблемно-зависимых каузальных знаний о событиях, действиях и процедурах. Каузальный сценарий задает типичную последовательность действий в заданной предметной области и описывается в виде фрейма, состоящего из слотов. Имена слотов отражают следующие понятия:

- деятель и участники сценария;
- цели и мотивы деятеля и участников;

- время, место, средства реализации сценария;
- ключ, посылки, следствия, побочные действия, закономерности;
- системное имя сценария.

Слот «ключ» задает основное событие, определяющее тип ситуации. Реализация ключевого события обеспечивает достижение цели деятеля и участников сценария. Слот «посылки» описывает необходимые условия реализации сценария. В посылках содержится последовательность действий, которые надо выполнить, чтобы создать необходимые условия для осуществления ключевого события. Слот «следствия» задает результаты его выполнения. Слот «побочные действия» описывает действия, реализующиеся параллельно с выполнением действий в посылках сценария. Сценарий считается завершенным, если произошло ключевое событие и реализована цель деятеля.

Кроме каузальных сценариев, встречаются и иные типы сценариев. Наиболее распространенными типами являются сценарии в виде дерева целей и классифицирующие сценарии.

В сценариях первого типа описывается, как некоторая цель может быть декомпозирована в систему подцелей. Такие сценарии применяются в случае планирования решений. Классифицирующие сценарии используются при обобщении знаний и представляют собой сети, между вершинами которых имеются отношения типа «часть – целое», «элемент – класс» или «род – вид».

ГЛАВА 3
АРХИТЕКТУРА И ТЕХНОЛОГИЯ РАЗРАБОТКИ ЭКСПЕРТНЫХ СИСТЕМ

3.1. Основные положения

ЭС представляют собой класс компьютерных программ, которые выдают советы, проводят анализ, выполняют классификацию и ставят диагноз. Они ориентированы на решение задач, обычно требующих проведения экспертизы человеком-специалистом. В отличие от машинных программ, использующих процедурный анализ, ЭС решают задачи в узкой предметной области (конкретной области экспертизы) на основе дедуктивных суждений.

Все ЭС состоят, по крайней мере, из трех основных элементов: базы знаний, машины вывода и интерфейса пользователя (рис. 3.1).

Группа экспертов или иной источник экспертизы обеспечивает загрузку в БЗ фактов, наблюдений и способов анализа ситуаций. Пользователь запрашивает систему о конкретных проблемах через интерфейс, который допускает общение с использованием обычных выражений. В мощных интеллектуальных системах существует интерфейс на естественном языке, который позволяет задавать вопросы и получать ответы на обычном английском или русском языках. В случае обычных интеллектуальных систем пользователю предоставляется не столь изысканный, но тем не менее «дружественный» интерфейс. Информация, содержащаяся в БЗ, обрабатывается с помощью машины вывода, которая использует эмпирические ассоциации или правила «ЕСЛИ…ТО» для формирования и проверки возможных решений. Интерфейс пользователя в доступной форме передает полученные результаты оператору.

БЗ содержит известные факты, выраженные в виде объектов, атрибутов и условий. Помимо описательных представлений о действительности, она включает выражения неопределенности, представляющие собой ограничения на достоверность фактов. В этом отношении БЗ отличается от традиционной БД. При обработке информации в БД пользуются заранее определенными логическими правилами. Соответственно БЗ, представляющая более высокий уровень абстракции, имеет дело с классами объектов, а не с самими объектами.

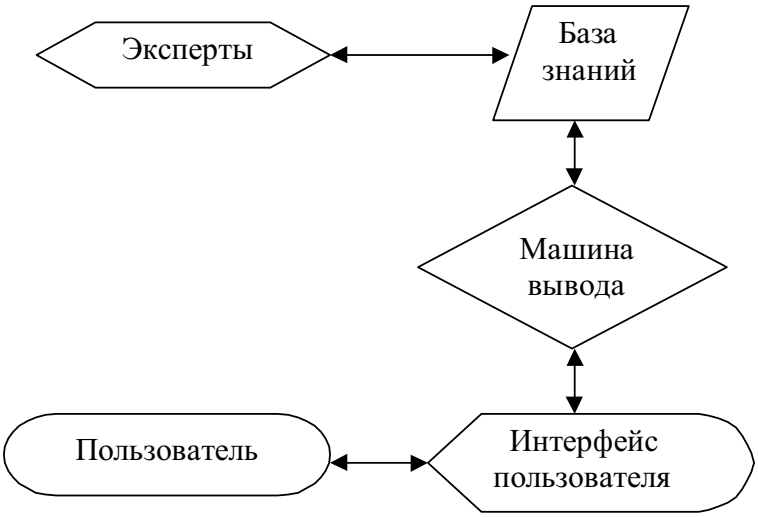


Рис. 3.1. Структура ЭС

БЗ создается консультантами, авторами учебников, исследователями, либо самими экспертами, либо на основе их работы. В поисках источника для наполнения БЗ жизненный опыт важнее, чем высокий интеллект. Эксперт, который исходит из продолжительных наблюдений за событиями в некоторой конкретной области знания, скорее всего, создаст более полезную БЗ, чем гениальный аналитик, который больше полагается на интуитивное проникновение в сущность явлений. Предположим, например, вы строите ЭС выбора компьютера для различных применений. Блестящий инженер-конструктор, отлично владеющий методами разработки интерфейсов, на практике будет менее полезен, чем консультант по применениям, который наблюдал большое количество установленных систем, выполняющих предназначенные для них задачи.

Главным в ЭС является механизм, осуществляющий поиск в БЗ по правилам рациональной логики, для получения решений. Этот механизм, называемый машиной вывода, приводится в действие при получении запроса пользователя и выполняет следующие задачи:

- сравнивает информацию, содержащуюся в запросе пользователя, с информацией базы знаний;
- ищет определенные цели или причинные связи;
- оценивает относительную определенность фактов, основываясь на соответствующих коэффициентах доверия, связанных с каждым фактом.

Как следует из ее названия, машина вывода предназначена для построения заключений. Ее действие аналогично рассуждениям эксперта-человека, который оценивает проблему и предлагает гипотетические решения. В поиске целей на основе предложенных правил машина вывода обращается к БЗ до тех пор, пока не найдет вероятный путь к получению приемлемого результата. Например, программа медицинской

диагностики сначала пытается выделить болезнетворный орган или организм, анализируя список на первый взгляд не связанных симптомов, а затем определяет курс эффективной терапии.

Задача интерфейса пользователя состоит в организации обмена информацией между оператором и машиной вывода. Интерфейс с использованием естественного языка создает видимость произвольной беседы, применяя повседневные выражения в правильно построенных предложениях. Очевидно, что чем более естественен такой интерфейс, тем выше требования к внешней и оперативной памяти. Следовательно, системы, предоставляющие пользователю максимум удобств, расходуют больше ресурсов основной машины, доступных с удаленных рабочих станций.

Инструментальные средства, предназначенные для работы на персональных компьютерах, имеющих ограниченные возможности, неизбежно приносят «дружественность» интерфейса в жертву эффективности ЭС в целом.

Интерфейс прямого ввода должен уметь распознавать язык или, по крайней мере, достаточное количество ключевых слов и фраз, чтобы улавливать их связь с рассматриваемой проблемой и ее предполагаемыми решениями.

Более мощные ЭС объясняют, почему задан тот или иной вопрос и как был сделан соответствующий вывод. Эти объяснения получаются на основе эвристических правил, с помощью которых происходит управление взаимодействием машины вывода с БЗ.

3.2. Технология разработки экспертной системы

Разработка (проектирование) ЭС существенно отличается от разработки обычного программного продукта. Опыт разработки первых ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату. Объясняется это как неформализованностью решаемых задач, так и отсутствием завершенной теории ЭС.

Перед тем как приступить к разработке ЭС, инженер по знаниям должен ответить на вопрос, следует ли разрабатывать ЭС для данного приложения. В общем виде ответ может быть таким: использовать ЭС следует тогда, когда ее разработка возможно оправдана и методы инженерии знаний соответствуют решаемой задаче.

Чтобы разработка ЭС была возможной (для данного приложения), необходимо одновременное выполнение, по крайней мере, следующих требований:

- эксперты в данной области должны решать задачу значительно лучше, чем начинающие специалисты;

- эксперты должны сходиться в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;
- эксперты должны уметь выразить на естественном языке и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что их знания будут «извлечены» и вложены в ЭС;
- задача, возложенная на ЭС, требует только рассуждений, а не действий (если требуются действия, то необходимо объединять ЭС с роботами);
- задача не должна быть слишком трудной, ее решение должно занимать у эксперта несколько дней, а не месяцев;
- 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;
- 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), т. к. подобные знания пока не удается (в достаточном количестве) вложить в системы ИИ [5].

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, использование ЭС для поиска полезных ископаемых может принести доход в сотни миллионов рублей в случае успеха;
- использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
- при передаче информации к эксперту происходит недопустимая потеря времени или информации;
- необходимость решать задачу в окружении, враждебном для человека.

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

- может быть естественным образом решена посредством манипуляции с символами (т. е. с помощью символьных рассуждений), а не с числами;
- должна иметь эвристическую (не алгоритмическую) природу, т. е. ее решение должно сводиться к применению эвристических правил;

- должна быть достаточно сложной, чтобы оправдать затраты на разработку ЭС, однако не должна быть чрезмерно сложной (решение занимает у эксперта дни, а не месяцы), чтобы ЭС могла ее решить;
- должна быть достаточно узкой, чтобы решаться методами инженерии знаний, и практически значимой.

При разработке ЭС используется концепция «быстрого прототипа». Суть ее состоит в том, что разработчики не пытаются сразу создать конечный продукт. На начальном этапе они создают прототип ЭС, который должен удовлетворять двум противоречивым требованиям. Прототип должен решать типичные задачи конкретного приложения, при этом время и трудоемкость его разработки должны быть весьма незначительными. Это необходимо для того, чтобы можно было параллельно осуществлять процесс накопления и отладки знаний (реализуемый экспертом) с процессом выбора (разработки) программных средств (реализуемым инженером по знаниям и программистом). Для удовлетворения указанных требований при создании прототипа, как правило, используются разнообразные инструментальные 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 инженерии знаний для данного приложения. По мере увеличения знаний прототип может достичь такого состояния, когда он решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как повышение быстродействия ЭС, так и уменьшение требуемой памяти.

В ходе работ по созданию ЭС сложилась определенная технология их разработки [5], включающая ряд этапов: идентификацию, получение знаний, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию (рис. 3.2).

На этапе идентификации определяются задачи, которые подлежат решению, выявляются цели разработки, ресурсы, эксперты и категории пользователей.

На этапе получения знаний выделяют три стратегии: приобретение знаний, извлечение знаний и обнаружение знаний.

Под *приобретением знаний* понимается способ автоматизированного наполнения базы знаний посредством диалога эксперта и специальной программы. *Извлечением знаний* называют процедуру взаимодействия инженера по знаниям с источником знаний (экспер-

том, специальной литературой и т. д.) без использования вычислительной техники.

Термин *обнаружение знаний* (knowledge discovery, data mining) связывают с созданием компьютерных систем, реализующих методы автоматического получения знаний. Сейчас это направление является наиболее перспективным. При этом предполагается, что система сама сможет раскрыть закономерности предметной области и сформулировать необходимые знания на основе имеющегося эмпирического материала [8].

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

Этап формализации определяет способы представления всех видов знаний, формализует основные понятия, определяет способы интерпретации знаний, моделирует работу системы. На этом этапе оценивается адекватность целям системы зафиксированных понятий, методов решения, средств представления и манипулирования знаниями.

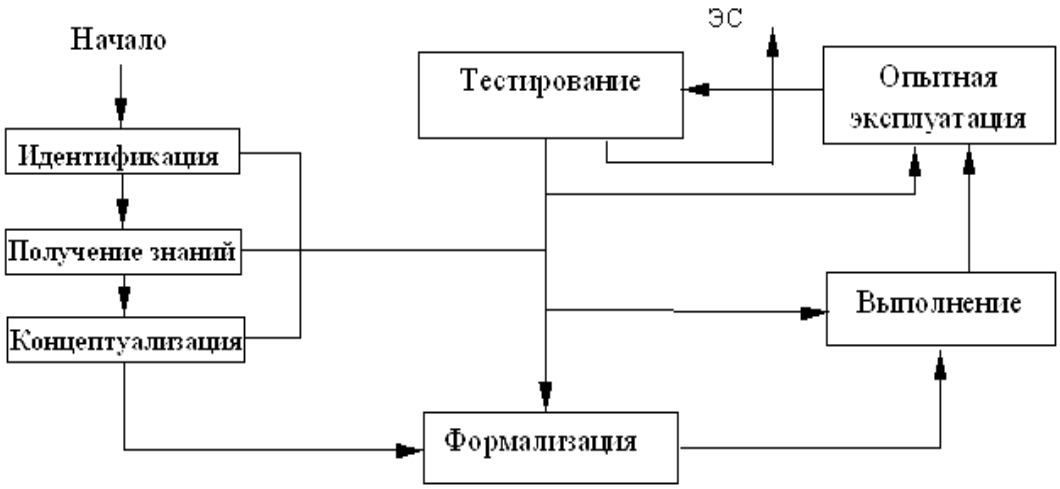


Рис. 3.2. Этапы технологии создания ЭС

На этапе выполнения осуществляется наполнение БЗ системы. На этапе тестирования эксперт (и инженер по знаниям) в интерактивном режиме, используя диалоговые и объяснительные средства, проверяет компетентность ЭС. Процесс тестирования продолжается до тех пор, пока эксперт не решит, что система достигла требуемого уровня компетентности.

На этапе опытной эксплуатации проверяется пригодность ЭС для конечных пользователей. По результатам этого этапа, так же как и этапа тестирования, может потребоваться существенная модификация ЭС.

Процесс создания ЭС не сводится к соблюдению строгой последовательности перечисленных выше этапов. В ходе разработки приходит-

ся неоднократно возвращаться на более ранние этапы и пересматривать принятые там решения [5].

3.3. Механизм вывода (интерпретатор правил)

Идеальная ЭС должна содержать следующие основные подсистемы: интерфейс с пользователем, систему логического вывода (механизм вывода), БЗ, составляющих ядро любой ЭС, а также модуль приобретения знаний, модуль отображения и объяснения решений (рис. 3.3) [4].

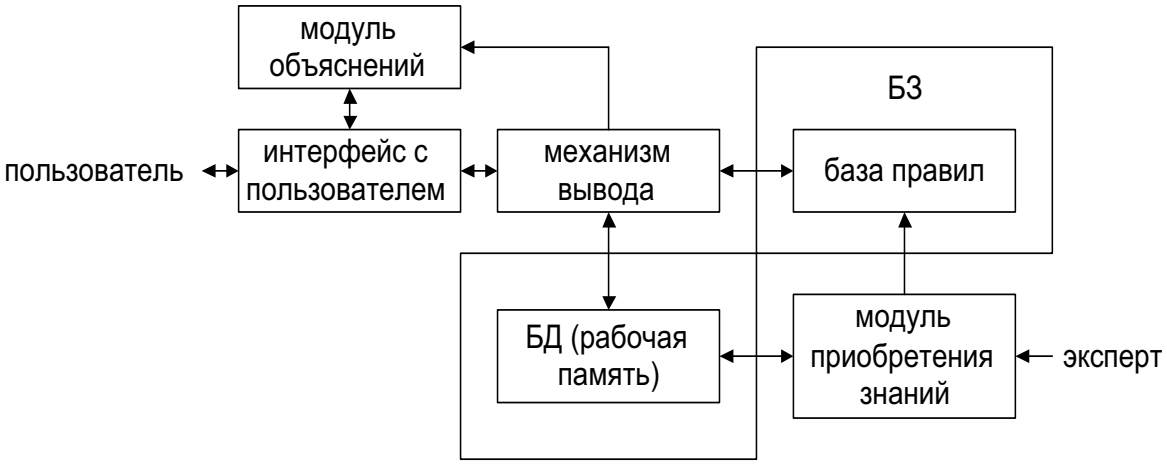


Рис. 3.3. Структура связей между подсистемами ЭС

Механизм вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр существующих фактов из рабочей памяти (БД) и правил из БЗ, добавление (по мере возможности) в рабочую память новых фактов и, во-вторых, определение порядка применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для очередного правила в рабочей памяти оказывается недостаточно данных [4].

В некоторых системах принят прямой порядок вывода: от фактов, которые находятся в рабочей памяти, к заключению. В других системах вывод осуществляется в обратном порядке: заключения просматриваются последовательно до тех пор, пока не будут обнаружены в рабочей памяти или получены от пользователя факты, подтверждающие одно из них. В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой небольшую по объему программу. Основную же часть памяти компьютера занимают правила.

Механизм вывода включает в себя два компонента: один из них реализует собственно вывод, другой управляет этим процессом. Компонент вывода выполняет первую задачу, просматривая имеющиеся правила и факты из рабочей памяти, и добавляет в нее новые факты при

срабатывании какого-нибудь правила. Управляющий компонент определяет порядок применения правил. Представляет интерес рассмотреть каждый из этих компонентов более подробно.

Компонент вывода. Его действие основано на применении правила вывода, суть которого состоит в следующем. Пусть известно, что истинно утверждение A и существует правило вида «ЕСЛИ A, ТО B», тогда утверждение B также истинно. Правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.

Хотя в принципе на первый взгляд кажется, что такой вывод легко может быть реализован на компьютере, тем не менее на практике человеческий мозг все равно оказывается более эффективным при решении задач. Например: «Мэри искала ключ». Для слова «ключ» допустимы, как минимум, два значения: «родник» и «ключ от квартиры». В следующих же двух предложениях одно и то же слово имеет совершенно разные значения: «Мы заблудились в чаще»; «Нужно чаще ходить в театры».

Понять факты становится еще сложнее, если они являются составными частями продукций. Например:

ЕСЛИ Белый автомобиль легко заметить ночью
И Автомобиль Джека белый,
ТО Автомобиль Джека легко заметить ночью.

Это заключение легко выведет даже ребенок, но оно оказывается не под силу ЭС.

Подводя итоги, можно сказать, что человек способен вывести большое число заключений с помощью очень большой БЗ, которая хранится в его памяти; а ЭС могут вывести только сравнительно небольшое число заключений, используя заданное множество правил.

Механизм вывода должен быть способен продолжить рассуждения и со временем найти решения даже при недостатке информации. Это решение может и не быть точным, однако система ни в коем случае не должна останавливаться из-за отсутствия какой-либо части входной информации.

Управляющий компонент. Управляющий компонент определяет порядок применения правил, а также устанавливает, имеются ли еще факты, которые могут быть изменены в случае продолжения консультации. Управляющий компонент выполняет четыре функции:

1. Сопоставление. В этом случае образец правила сопоставляется с имеющимися фактами.
2. Выбор. Если в конкретной ситуации могут быть применены сразу несколько правил, то в этом случае из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

3. Срабатывание. Если образец правила при сопоставлении совпадает с какими-либо фактами из рабочей памяти, то правило срабатывает.
4. Действие. Рабочая память подвергается изменению путем добавления в нее заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется (как, например, в системах обеспечения безопасности информации).

Интерпретатор правил работает циклически. В каждом цикле он просматривает все правила, чтобы выявить среди них те, посылки которых совпадают с известными на данный момент фактами из рабочей памяти. Интерпретатор определяет также порядок применения правил. После выбора правило срабатывает, его заключение заносится в рабочую память, и затем цикл повторяется сначала.

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то интерпретатор производит выбор по определенному критерию единственного правила, которое и сработает в данном цикле. Цикл работы интерпретатора схематически представлен на рис. 3.4.

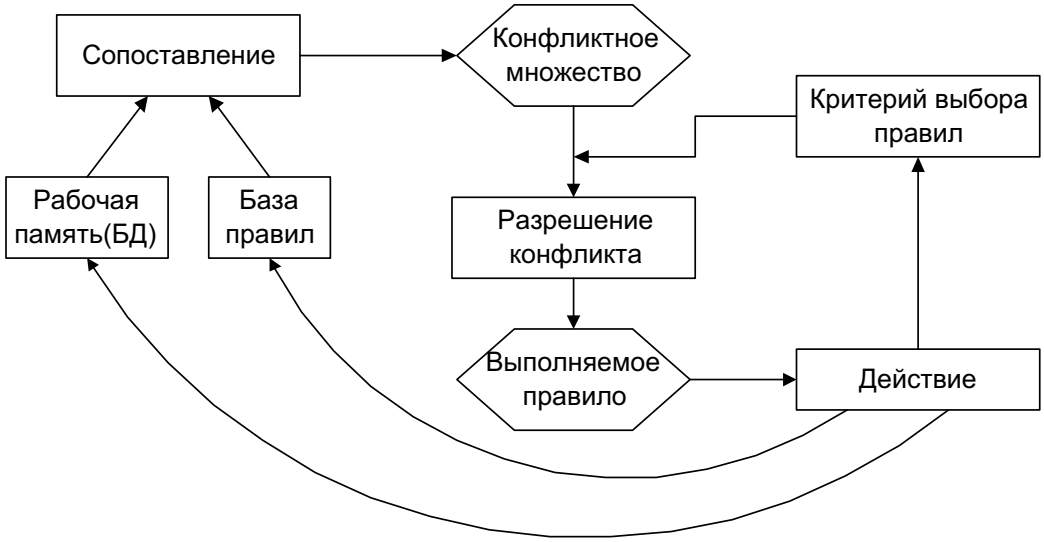


Рис. 3.4. Цикл работы интерпретатора правил

Информация из рабочей памяти последовательно сопоставляется с правилами для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое конфликтное множество. Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он выбирает единственное правило, после чего он срабатывает. Это выражается в занесении фактов, образующих заключение правила, в рабочую память или в изменении критерия выбора конфликтующих правил. Если же в заключение правила входит какое-либо действие, то оно выполняется.

Новые данные, введенные в систему сработавшим правилом, в свою очередь, могут изменить критерий выбора правила. В действительности ЭС не располагают процедурами, которые могли бы построить в пространстве состояний сразу весь путь решения задачи. Более того, зачастую даже не удается определить, имеется ли вообще какое-нибудь решение задачи. Тем не менее, поиск решения выполняется, поскольку движением в пространстве состояний управляют скрытые или виртуальные процедуры. Такие управляющие процедуры получили название *недетерминированных процедур*. Это означает, что траектория поиска решения в пространстве состояний полностью определяется данными. Следует отметить, что при разработке управляющего компонента механизма вывода существенным является определение критерия выбора правила, которое будет применено в конкретном цикле.

3.4. Взаимодействие пользователей с экспертной системой

Взаимодействие с ЭС осуществляют различные типы пользователей: пользователи-неспециалисты, пользователи-специалисты, эксперты, пользователи-студенты, инженеры по знаниям [12, 13].

Задача пользователей, не являющихся специалистами в области экспертизы, состоит в получении от ЭС решения некоторой задачи. Задача пользователей-специалистов в области экспертизы заключается в использовании ЭС для сокращения трудоемкости получения результата или повышения его качества. Задача пользователей-студентов состоит в обучении с помощью ЭС методам решения задач из области экспертизы. Задача экспертов, т. е. высококвалифицированных специалистов, заключается в обнаружении недостающих знаний и вводе их в систему, т. е. осуществлении отладки знаний. Задача инженеров по знаниям заключается в отладке управляющего механизма, анализе и модификации ЭС.

Каждый из перечисленных типов пользователей предъявляет свои специфические требования к общению, но всех их объединяет следующее:

- язык общения представляет собой ограниченный естественный язык, а не формальный язык программирования;
- процесс взаимодействия пользователей любого типа с ЭС не сводится к обмену изолированными парами предложений («запрос–ответ»), а представляет собой разветвленный диалог, в котором инициатива переходит от одного участника к другому.

Сложность и обособленность задач, решаемых в процессе общения пользователей с ЭС, приводит к необходимости выделения в структуре ЭС компоненты взаимодействия.

Назначение компоненты взаимодействия состоит в следующем:

- организовать диалог «пользователь – ЭС», т. е. распределить функции участников общения в ходе кооперативного решения задачи и отслеживать состояние диалога как функцию текущих целей участников и фазы решения задачи;
- осуществить обработку отдельного сообщения с учетом текущего состояния диалога, т. е. осуществить преобразование сообщения из естественно-языковой формы в форму внутреннего представления или обратное преобразование.

Общая схема компоненты взаимодействия, состоящей из диалоговой подсистемы и подсистемы анализа и синтеза, приведена на рис. 3.5.

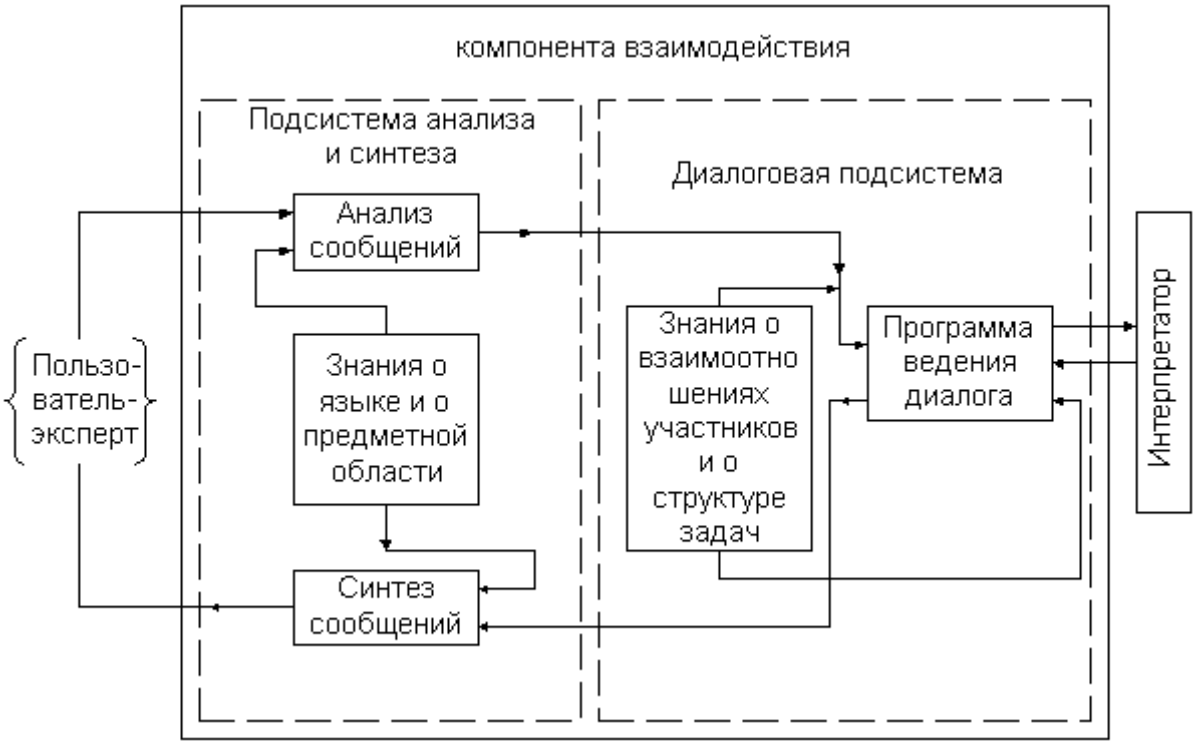


Рис. 3.5. Схема компоненты взаимодействия

В настоящее время не существует полной общепринятой модели диалога. Диалог рассматривается как процесс непротиворечивого взаимодействия участников, в котором они преследуют свои цели с помощью обмена сообщениями на установленном языке общения [12].

Диалог возможен благодаря согласованности целей участников. При этом цели известны участникам заранее и дополняют друг друга, например диалог-покупка (покупатель/продавец) и т. д. Именно пара целей определяет тип диалога, т. е. множество диалогов, преследующих данные цели, вне зависимости от конкретных участников общения и решаемой ими задачи.

Информация о типе диалога может быть охарактеризована следующими компонентами: множеством параметров, описаниями параметров и макроструктурой диалога.

Множество параметров несет информацию о том, в каких пределах может варьироваться данный тип диалога. Обычно достаточно охарактеризовать любой диалог тремя параметрами: двумя параметрами, определяющими участников (их роли), и предметом (темой) диалога.

Описания параметров содержат множество утверждений о параметрах. В первую очередь эти утверждения определяют цели и состояния участников, которые должны соблюдаться в ходе ведения диалога данного типа. Описания характеризуют те аспекты диалога, которые остаются неизменными на протяжении всего диалога данного типа. Аспекты диалога, изменяемые систематическим образом, представляются в виде общей структуры (макроструктуры) диалога. Макроструктура задается в виде множества подцелей участников, частично упорядоченных во времени.

Элементарную единицу диалога называют *шагом диалога*. Последовательность нескольких шагов диалога называют *диалоговой последовательностью*. Шаг диалога состоит из действия первого участника (инициатора действия) и следующей за ним реакции второго участника. Действие всегда составляет первую часть шага, а реакция – вторую. Термин «реакция» является более предпочтительным, чем термин «ответ», т. к. реакция по форме может быть не ответом (в смысле ответа на вопрос), а, например, вопросом. Действие состоит из подготовки и осуществления сообщения инициатором. Реакция состоит из подготовки и осуществления сообщения вторым участником.

При рассмотрении последовательности шагов диалога для определения того, от кого исходит действие, а от кого – реакция, необходимо учитывать смысл. Участники могут перехватывать инициативу, т. е. вместо реакции в ответ на действие первого участника второй участник может совершить действие. Например, вместо ответа на поставленный вопрос задается встречный вопрос. Перехват инициативы необходим при возникновении непонимания, несогласия или недоверия к действиям собеседника.

Шаг диалога характеризуется следующими параметрами:

- инициатор и тип инициирования;
- способ и форма влияния действия на реакцию;
- способ спецификации шага (подзадачи).

Инициатором шага диалога может быть пользователь или система. При этом действия пользователя всегда обозначают выбор или явную формулировку определенной задачи. Действия системы обычно подразделяются на запрос и предложение. В случае действия-запроса система предлагает пользователю определить (ввести) задачу. В случае дейст-

вия-предложения система предлагает пользователю выбрать из некоторого ограниченного множества задач интересующую его задачу.

Влияние действия на реакцию обычно представляют в виде следующих форм: команды (задание действия), меню (предложение для выбора реакции) и анкеты (предложения для выбора значений некоторых сущностей).

По способу влияния действия на реакцию выделяют:

- свободный (неограниченный) выбор, т. е. действие не накладывает ограничений на вид реакции;
- ограниченный выбор, т. е. действие ограничивает разнообразие реакций. Ограничения на возможные реакции могут быть заданы либо путем указания множества выбора (как в меню), либо заданием жесткого формата, который должен быть соблюден в реакции.

По способу спецификации шага можно говорить об автоматической (однозначной) спецификации задачи, обсуждаемой на данном шаге, и о возможной, но не обязательно однозначной спецификации задачи. Например, меню и синтаксически правильная команда вызывают однозначную спецификацию задачи системой, а высказывания на ограниченном естественном языке не гарантируют однозначной спецификации.

3.5. Подсистема анализа и синтеза сообщений

Задача подсистемы анализа и синтеза состоит в обработке отдельных сообщений системы и пользователя. Сообщения системы можно разделить на следующие основные типы:

- запросы к пользователю о значении некоторых атрибутов решаемой задачи (инициатор – система);
- сообщение пользователю результатов решений (инициатор – система);
- объяснение пользователю действий или знаний системы (инициатор – пользователь);
- генерация новых знаний, введенных в систему с целью показать, как эти знания поняты системой (инициатор – пользователь) [12].

Сообщения пользователя, анализируемые системой, можно разделить на следующие типы:

- ответ пользователя на запрос о значении некоторого атрибута (инициатор – система);
- оценка пользователем результата решения, предложенного системой (инициатор – система);
- запрос пользователя на объяснение действий или знания системы (инициатор – пользователь);
- факт, содержащий новое знание, обычно новое правило (инициатор – пользователь).

Обработка сообщений пользователя сводится к анализу входных сообщений, а обработка сообщений системы – к синтезу выходных сообщений. Сложность методов анализа и синтеза зависит как от языка общения, так и от языка, используемого для представления знаний.

Так, например, на этапе консультации язык общения может быть строго формализован фиксированным набором запросов системы и множеством возможных ответов пользователя. В этих условиях задача синтеза сводится к генерации подготовленных заранее вопросов, а задача анализа – к обработке слов и словосочетаний, требующих для флективных языков морфологического анализа.

На этапах объяснения и приобретения знаний язык общения более сложен. Здесь уже невозможно предвидеть разнообразие способов выражения на естественном языке запросов или фактов (правил), вводимых пользователем. На этих этапах требуется анализировать не отдельные словосочетания, а предложения. Таким образом, задача анализа сводится к разбиению предложений на словосочетания и последующей обработке словосочетаний, т. е., кроме использования морфологии, требуется привлечение синтаксиса и семантики. Однако в большинстве ЭС удается обойтись простейшей семантической обработкой. Простота семантического анализа обусловлена ограниченностью области экспертизы существующих ЭС.

Задача синтеза на этапах приобретения знаний и объяснения в существующих системах сводится к использованию шаблонов и (или) заранее подготовленных сообщений. Необходимо отметить, что в случае взаимодействия с пользователями на флективных языках (например, на русском языке, в отличие от английского языка) при применении шаблонов неизбежно используются элементы морфологического синтеза.

Подсистема анализа и синтеза сообщений анализирует входные сообщения пользователя и синтезирует выходные сообщения, адресованные пользователю [12]. Тип сообщений пользователя или системы определяется диалоговой подсистемой. Общая схема подсистемы анализа и синтеза приведена на рис. 3.6.

Данная подсистема имеет элементы избыточности, что позволяет ей использоваться в различных ЭС.

Предлагаемая базовая подсистема осуществляет анализ входного сообщения с помощью программ морфологического, синтаксического, семантического анализа. Многоэтапность анализа вызвана сложностью естественного языка (ЕЯ) и в первую очередь такими его особенностями, как разветвленная синонимия и омонимия ЕЯ и контекстная зависимость высказываний и слов ЕЯ и т. п.

При выполнении морфологического и синтаксического анализа (МА и СИА) используется хранимая в словаре подсистемы информация

о языке общения. Словарь содержит морфологическую и синтактико-семантическую информацию об индивидуальных особенностях слов русского языка, не содержащуюся в грамматике языка. Результатом работы МА является выделение основ (корней) слов, отождествление этих основ со словарем и приписывание им морфологической информации (МИ). Под МИ подразумеваются части речи, род, число, падеж, время и т. п. Результатом работы этапа СИА является построение для входного сообщения соответствующей ему обобщенной синтаксической структуры, отражающей взаимосвязи слов в сообщении. На выходе семантического анализа (СЕА) формируется внутреннее представление входного сообщения, отражающее знания системы о предметной области.

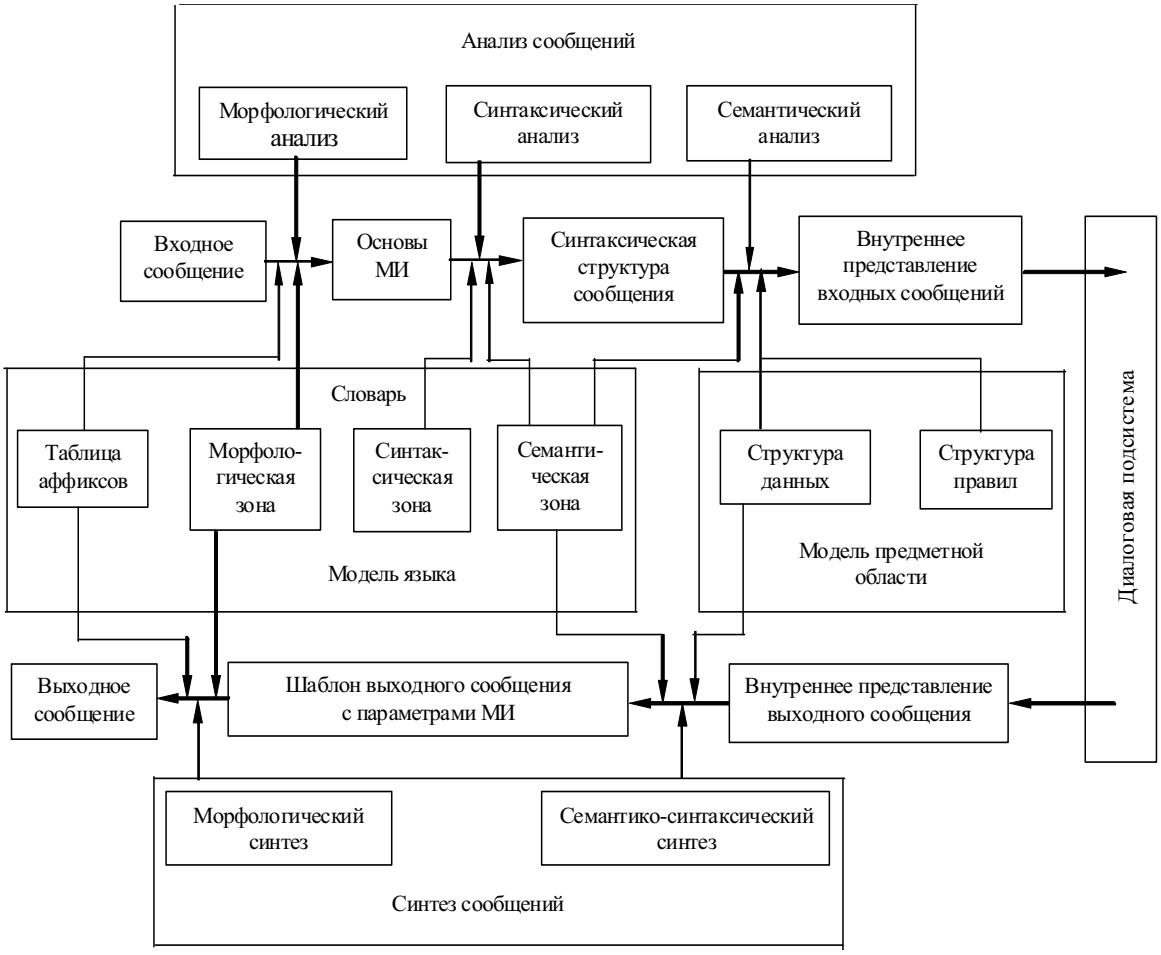


Рис. 3.6. Схема подсистемы анализа и синтеза

В ЭС применяется упрощенный синтез выходных сообщений. В большинстве приложений используется метод шаблонов, содержащий элементы семантики и синтаксиса. Шаблоны ставятся в соответствие основным конструкциям внутреннего представления. Шаблон представляет собой текст на ЕЯ с некоторыми пробелами. В процессе синте-

за сообщений осуществляется подстановка на места пробелов необходимых слов в соответствующей форме (род, число, падеж).

3.6. Морфологический анализ входных сообщений

Под МА понимается обработка *словоформ* вне связи с контекстом. *Словоформой* будем называть отрезок текста между двумя соседними пробелами (при этом знаки препинания считаются отдельными словоформами) [12]. Функцией МА является идентификация словоформы и приписывание словоформе характеризующего ее комплекса морфологической информации (КМИ). КМИ состоит в общем случае из совокупности строк МИ.

Существуют два основных метода реализации МА: декларативный и процедурный. В *декларативном методе* реализации МА в словаре системы хранятся все возможные словоформы каждого слова с приписанной им морфологической информацией. По сути дела, в декларативном МА нет собственно МА, а хранится его результат. Поэтому декларативный МА работает быстрее, чем процедурный. Задача декларативного МА состоит только в поиске словоформы в словаре и переписывании из словаря КМИ, соответствующего данной словоформе. В связи с тем что количество различных словоформ у одного слова довольно велико (у существительного – до 12, у прилагательного – до 36, а у глагола с учетом отглагольных форм – до 100), декларативный МА имеет, по сравнению с процедурным, следующие недостатки:

- значительно возрастает трудоемкость подготовки морфологической зоны словаря, т. к. человек должен занести в словарь для каждого слова все его словоформы с соответствующими им строками МИ;
- увеличиваются затраты памяти, т. к. для каждого слова (лексемы) хранятся все его словоформы.

При *процедурном* МА в словаре системы хранятся основы слов. Процедурный МА выполняет следующие функции: выделяет в текущей словоформе основу, идентифицирует ее и приписывает данной словоформе соответствующий ей КМИ.

Рассмотрим один из возможных методов выполнения МА словоформ русского языка, так называемый «обратный» метод. При работе алгоритм процедурного МА использует информацию из морфологической зоны словаря и из таблиц аффиксов. Удобно разделить словарь на две части: словарь основ (СО) и словарь готовых словоформ (СГФ). Общая схема алгоритма состоит из следующих шагов:

- поиск словоформы в словаре СГФ;

- выделение основы;
- поиск в словаре основ;
- обработка словосочетаний;
- предсинтаксис [12].

В СГФ целесообразно хранить неизменяемые слова и слова с нерегулярными формами изменения. Примерами таких слов являются предлоги, наречия, неизменяемые существительные (например, пальто), знаки препинания, цифры, некоторые формы глагола (идти – шел) и т. п. Для этих слов МА не требуется вообще.

Задача первого блока МА состоит в том, чтобы определить, не относится ли очередная словоформа входного сообщения к числу тех, которые хранятся в СГФ. Если словоформа найдена, то из СГФ переписывается вся соответствующая ей морфологическая информация и морфологический анализ данной словоформы заканчивается. В противном случае выполняется второй блок МА. Его работа сводится к последовательным проверкам возможностей вложения в анализируемую словоформу справа налево окончаний и суффиксов. При этом для ускорения проверок вкладываются (отсекаются) аффиксы с большим числом букв, а информация о вкладывающихся в них меньших аффиксах получается не поиском, а за счет отсылок.

В результате вложения всех возможных аффиксов словоформе ставится в соответствие одна или несколько гипотетических основ (ГО) и для каждой основы – КМИ. ГО ищутся в словаре основ во время работы третьего блока МА. В случае нахождения ГО в словаре основ и совпадения части речи ГО с частью речи словарной основы данная ГО и ее КМИ признаются правильными. В противном случае ГО признается ошибочной и отбрасывается. При нахождении в словаре ГО вместе с морфологической зоной считываются и ее синтактико-семантические зоны.

Задача четвертого блока алгоритма состоит в выполнении операции «склеивания», т. е. в преобразовании некоторых словоформ, отделенных друг от друга пробелами или другими словоформами, к одной основе. Данная операция выполняется для упрощения синтаксического и семантического анализа.

Завершает работу МА пятый блок алгоритма, называемый «предсинтаксис». В задачу данного блока входит подготовка данных, упрощающая работу синтаксического анализа:

- формируются номера уровней словоформ входного сообщения;
- помечаются слова, возможно обусловленные предыдущими словами;
- используются предварительные синтаксические фильтры, устраняющие избыточную морфологическую информацию.

Первая из перечисленных подзадач состоит в выделении уровня каждой словоформы. При этом словоформы основного предложения образуют нулевой уровень, а словоформы придаточных предложений и вложенных друг в друга оборотов (причастных, деепричастных) имеют более высокие уровни по числу вложенности. Выделение уровней позволяет на этапе синтаксического анализа ускорить обработку за счет устранения взаимосвязи слов между разными уровнями.

3.7. Синтаксический анализ входных сообщений

Задачей СИА является построение синтаксической структуры входного предложения (осуществление разбора предложения) на основе МИ о словоформах и синтаксических правил объединения слов и словосочетаний. Синтаксическая структура отражает синтаксические связи, существующие между словами в предложении. Ее получение начинается с построения всевозможных связей между словами, которые в последующем отсеиваются на основе локальных и глобальных «фильтров». Конкретный вид структуры определяется выбранной системой синтаксических отношений (ССИО) [12].

Существует несколько способов описания синтаксической структуры, но два из них – система составляющих и дерево зависимостей – являются наиболее употребительными.

Остановимся подробнее на *системе составляющих*. Произвольная непустая последовательность словоформ называется *цепочкой*. Число словоформ в цепочке x называется длиной цепочки и обозначается $|x|$. Если для каких-либо цепочек x, y, z_1, z_2 имеет место равенство $x = z_1 y z_2$, то говорят, что цепочка y входит в цепочку x. Вхождения словоформ в цепочку называются ее *точками*. Если l и m – точки одной и той же цепочки $x = z_1 l z_2 = y_1 m y_2$ и если при этом $|z_1| < |y_1|$, то $l < m$ и говорят, что l расположена левее m, а m – правее l. Для любых двух точек l и m цепочки x, таких, что $l \le m$, введем понятие *отрезка цепочки* x, представляющего множество точек t, удовлетворяющих неравенствам $l \le t \le m$.

Пусть x – произвольная непустая цепочка. Множество C отрезков цепочки x называется *системой составляющих этой цепочки*, если оно удовлетворяет двум условиям:

1) множество C содержит отрезок, состоящий из всех точек цепочки x, и все одноточечные отрезки x;
2) любые два отрезка из C либо не пересекаются, либо один из них содержится в другом.

Элементы C называются составляющими. Одноточечные отрезки называются *точечными (тривиальными) составляющими*.

Для наглядного изображения системы составляющих каждая нетривиальная составляющая заключается в скобки, причем левые и правые скобки одной составляющей могут быть помечены одинаковой меткой для ее выделения. Например, для предложения «Мы увидели древние стены города» допустима система составляющих:

(Мы увидели ((древние стены) города));
1 23 3 21
(Мы увидели (древние (стены города))).
1 2 3 321

Система составляющих указывает в предложении словосочетания разных уровней, не вводя при этом иерархии среди словосочетаний одного уровня.

Остановимся подробнее на способе описания синтаксической структуры с помощью *деревьев зависимостей* **(**деревьев синтаксического подчинения).

Пусть x – произвольная непустая цепочка и X – множество всех точек x. Дерево зависимостей цепочки x можно изобразить в виде последовательности образующих ее точек, расставленных на прямой линии. Для всякой пары точек l, m цепочки x, для которой существует зависимость между этими точками, на рисунке проводится дуга из l в m, причем таким образом, чтобы все дуги были по одну сторону от прямой. При этом точку l называют *управляющей точкой* («хозяином»), а m *– подчиненной точкой* («слугой»). На рис. 3.7 приведен вид дерева зависимостей цепочки *agbocdef* .

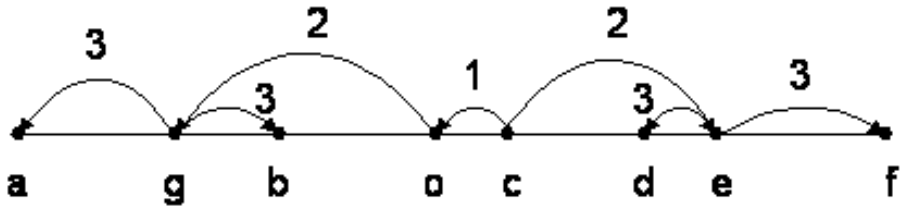


Рис. 3.7. Пример дерева зависимостей

Системы составляющих и деревья зависимостей характеризуют синтаксическую структуру предложения в разных аспектах. С помощью систем составляющих описываются в явном виде словосочетания, но игнорируется ориентация связей (т. е. не различаются «хозяин»

и «слуга»); вторые дают возможность рассматривать направленные связи, но только между отдельными словами.

Иногда для представления синтаксической структуры предложения используют смешанное представление, называемое *обобщенной синтаксической структурой* (ОСС). ОСС выражает, как и дерево зависимостей, ориентацию связей, но, в отличие от дерева зависимостей, ОСС снабжена информацией о словосочетаниях, образованных группами членов предложения (группой подлежащего, группой сказуемого, группой дополнений, группой обстоятельств и т. п.). Эти группы называются *именными группами* (ИГ). Кроме того, ОСС изображает в общем случае не один вариант разбора предложения, а несколько омонимичных (с точки зрения СИА) вариантов. Указанное обстоятельство позволяет уменьшить как количество возвратов от семантического анализа к синтаксическому анализу, так и количество вариантов разбора, генерируемых на этапе СИА.

Рассмотрим основные принципы построения алгоритма синтаксического анализа.

Традиционным методом построения синтаксической структуры фразы русского языка является метод фильтров [12]. В данном методе построение дерева зависимостей начинается с построения наборов всевозможных связей (синтаксических отношений) между словами. В чистом виде метод фильтров для практической реализации неприменим, т. к. число всевозможных связей между словами весьма велико, а число всевозможных способов выбора из них конкретного дерева зависимостей огромно. На практике для получения эффективных алгоритмов необходимо применять методы, направляющие и ускоряющие выбор правильных вариантов анализа.

Общая структура алгоритма СИА может быть определена следующим образом:

1. Выбор из всех возможных способов разбиения предложения на словосочетания (именные группы) приемлемого варианта синтаксических отношений.
2. Для выбранного разбиения предложения на ИГ осуществляется построение всех вариантов проведения СИО внутри каждой ИГ, т. е. построение обобщенного варианта разбора ИГ.
3. Если при выполнении пп. 1 и 2 СИО были установлены на все слова предложения, то алгоритм передает управление этапу СЕА. Иначе управление передается п. 1 для выбора очередного разбиения предложения на ИГ.
4. Если этап СЕА не смог обработать переданную ему синтаксическую структуру входного предложения, то этап СЕА возвращает управление п. 1.

3.8. Семантический анализ входных сообщений

На вход блока СЕА поступает текущий вариант синтаксического разбора обрабатываемого предложения. Задача СЕА состоит в том, чтобы принять или отвергнуть данный вариант работы СИА. Если вариант принят, то СЕА строит внутреннее представление этого предложения. Если вариант разбора отвергается, то СЕА передает управление на СИА для выработки очередного варианта разбора. В отличие от этапов МА и СИА, которые в значительной степени не зависят от предметной области и от способа представления знаний, принятого в конкретной ЭС, этап СЕА, как правило, не удается сделать универсальным [12].

Можно выделить четыре основные группы предикатов, характерных для деловой прозы. Это предикаты состояния, действия, функциональные и пустые. Предикаты состояния и действия характеризуют состояния объектов, которые реально существуют и которые можно наблюдать в моделируемом мире. Например: РАВНО, ИЗВЕСТНО, ВЗЯТЬ и т. д. Этим предикатам во внутреннем представлении системы соответствуют определенные конструкции, моделирующие окружающий мир.

Предикаты функционального типа представляют собой операторы над предикатами действия. Они указывают, какие операции необходимо совершить над предикатами действия при обработке входного предложения. Например: ВЫПОЛНИТЬ, УМЕНЬШИТЬ, ПЕРЕВЫПОЛНИТЬ и т. д. Функциональные предикаты не имеют представителя в модели мира. Они вырабатывают обращение к процедуре, которая производит обработку предикатов действия.

К пустым предикатам относят слова, которые не несут в данной предметной области реальной семантической нагрузки. Например: глаголы ОСУЩЕСТВЛЯТЬ, ЯВЛЯТЬСЯ и т. п. Обработка пустого предиката сводится к его устранению, что приводит к необходимости перестройки синтаксической структуры.

Этап СЕА можно разбить на следующие фазы:

1. Выбор очередного варианта назначения предикатных функций (т. е. определение количества и вида предикатных функций).
2. Перебор для каждой из выбранных предикатных функций всех возможных подстановок аргументов из слов (словосочетаний) обрабатываемого предложения; в случае успеха – завершение СЕА.
3. При неудачной подстановке определение очередного варианта; если вариант существует, то происходит переход к п. 1, в противном случае – к СИА.

3.9. Синтез выходных сообщений

Синтез сообщений используется ЭС при выполнении следующих действий:

- выдача результата работы системы;
- выдача текста объяснений;
- задание запроса пользователю о значении какого-либо атрибута.

В первых двух случаях синтезируется утвердительное предложение, а в третьем случае – вопросительное.

В большинстве ЭС для синтеза сообщений применяются простейшие средства, сводящиеся к использованию шаблонов. Суть синтеза по шаблонам состоит в том, чтобы для конкретной системы рассмотреть все типы сообщений, относящиеся к процессам общения и выдачи результатов работы ЭС, и для каждого типа разработать шаблон, который заполняется при обращении к пользователю. Например, при незнакомом слове сообщение может иметь вид:

«Слово … отсутствует в словаре системы».

Задача синтеза заключается в переводе «текста» машинного языка (М-языка) в текст ЕЯ. При этом необходимо:

- определить информацию, которую нужно сообщить пользователю;
- определить уровень общности синтезируемой информации;
- выделить обязательную и необязательную информацию, выражаемую в синтезируемых фразах;
- осуществить разбиение текста М-языка на фрагменты, соответствующие будущим фразам;
- определить лексемы для синтезируемой фразы;
- построить синтаксическую структуру фразы;
- приписать морфологическую информацию вершинам синтаксической структуры фразы;
- определить порядок слов;
- осуществить морфологический синтез словоформ.

Суть СЕА заключается в таком преобразовании текста М-языка, при котором его части могли бы соответствовать будущим фразам и предложениям ЕЯ. При этом требуется учет как языкового, так и смыслового фактора. Фраза должна быть приемлема по размерам, быть стилистически доступной и т. п. Иногда для этого достаточно использовать простые правила с учетом ограничения, например, на число существительных, на число определений, выражаемых придаточными предложениями. Такие преобразования осуществляются за счет правил фрагментирования текста М-языка. Результатом се-

мантического синтеза будет структура М-языка, разбитая на фрагменты, соответствующие будущим фразам.

Задача синтаксического синтеза – конкретизация синтаксических структур с учетом отношений между лексемами. Здесь выбирается форма фраз (например, актив–пассив) и морфологические характеристики лексем (например, определенный род, число, падеж и т. д.).

Задача морфологического синтеза – построение конкретных словоформ ЕЯ по словарю и заданной морфологической информации. Морфологический синтез реализуется декларативным и процедурным способами. При декларативном способе в словаре системы каждой нормализованной единице ставятся в соответствие возможные наборы МИ и считываются соответствующие им словоформы. При процедурном способе словоформы формируются с помощью таблиц аффиксов, расклассифицированных по частям речи.

Морфологический синтез завершает процесс синтезирования, после чего сообщение на ЕЯ выдается пользователю.

3.10. Диалоговая подсистема

Диалог можно рассматривать на трех уровнях:

- общая структура (макроструктура) диалога, характеризующая тип диалога;
- тематическая структура, отражающая структуру задачи, решаемой в текущем диалоге данного типа;
- структура шага диалога, отражающая взаимодействие участников в элементарном акте диалога [12].

Общая структура диалога. На этом уровне действия диалоговой подсистемы могут быть заданы в виде последовательности следующих этапов:

- инструктаж (на данном этапе система объясняет свое назначение, определяет порядок ведения диалога и перечисляет средства, которые доступны пользователю);
- определение задачи;
- решение задачи;
- объяснения в ходе решения задачи;
- выдача результата и его оценка (успех, неудача);
- объяснения после решения задачи;
- определение причин неудачи и приобретение новых знаний.

Приведенный перечень этапов (макроструктура диалога) является типичным для большинства ЭС. Перечисленные этапы не обязательно должны присутствовать в каждом конкретном диалоге, но если они имеются, то следуют в порядке, заданном макроструктурой. Тот или

иной этап может отсутствовать либо в связи с явным указанием пользователя, либо по умолчанию. Например, если результат работы оценен пользователем как положительный, то по умолчанию система может пропустить этапы объяснения и приобретения знаний.

Тематическая структура диалога. На данном уровне необходимо решить следующие вопросы:

- определить структуру задачи, т. е. разбить исходную задачу на упорядоченное множество подзадач;
- распределить подзадачи общей задачи между участниками диалога, т. е. указать, решение каких подзадач система осуществляет сама, а за решением каких обращается к пользователю;
- обеспечить такую последовательность вызова исходных данных и подзадач, решаемых пользователем, которая будет ему понятна;
- определить механизм, осуществляющий генерацию обращений к пользователю за исходными данными и за значениями атрибутов, получаемых от пользователя.

В ЭС MYCIN первый из перечисленных вопросов решается с помощью дерева контекстов. Дерево контекстов может быть проинтерпретировано как последовательность подзадач, на которые разбивается исходная задача:

- собрать сведения о пациенте;
- собрать сведения об инфекции;
- собрать сведения о культурах, которые были взяты у пациента сейчас и ранее;
- определить, какие микроорганизмы присутствуют в каждой культуре;
- определить, какие микроорганизмы являются «патогенными», т. е. порождающими болезни;
- определить, какие лекарства и в каких дозах надо назначить пациенту.

Итак, дерево контекстов разбивает задачу на подзадачи способом, понятным пользователю. С точки зрения организации диалога важно отметить, что разбиение на подзадачи находит непосредственное отражение не только в ходе решения, но и в последовательности развития тем диалога.

Второй вопрос (распределение подзадач между участниками) решен MYCIN следующим образом. Все подзадачи, решаемые системой, сводятся к определению значения какого-либо атрибута с помощью некоторого правила. Система пытается определить без обращения к пользователю значения всех атрибутов, за исключением тех, о которых ей известно, что они должны быть запрошены у пользователя. Если пользователь не может сообщить значение атрибута, то система все же пытается вывести это значение.

Структура шага диалога. Шаг диалога состоит из действия и реакции и характеризуется следующими параметрами:

1) инициатор и тип инициирования;
2) способ влияния действия на реакцию;
3) способ спецификации задачи.

На данном уровне задача диалоговой подсистемы состоит в определении значений параметров текущего шага диалога. Обычно в ЭС инициатором шага является система, т. е. она задает действие, ожидая реакцию пользователя. Исключение составляют ситуации, в которых пользователь перехватывает инициативу у системы. В случае перехвата инициативы пользователь прерывает шаг диалога и создает внутри него один или несколько шагов диалога, в которых инициатива принадлежит ему.

Действия системы по типу инициирования имеют вид как запросов, так и предложений. Действия пользователя обычно оформляются в виде заранее заданного набора команд. Фиксированность этого набора команд позволяет системе легко определить как ситуацию, в которой пользователь перехватывает инициативу, так и способ реакции системы на это прерывание.

В большинстве случаев ЭС, совершив действие, ограничивают выбор возможных реакций либо множеством возможных ответов, либо указанием синтаксиса допустимого ответа. Как правило, в рамках одной системы используются оба способа. Например, в MYCIN ограничение множества возможных ответов задается явно (в виде меню) или неявно (в виде списка ожидаемых значений атрибута).

Использование действий, ограничивающих выбор возможных реакций, позволяет упростить обработку сообщений пользователя и в ряде случаев, особенно для английского языка, избежать необходимости анализа фраз ЕЯ. Обычно удается обойтись без обработки предложений ЕЯ на всех этапах работы системы, кроме этапа приобретения знаний.

3.11. Объяснительные способности экспертных систем

Важность объяснений в ЭС вызвана рядом факторов. Во-первых, трудно ожидать, что пользователи будут знать все возможности и понимать все действия ЭС. В связи с этим пользователям требуется помощь ЭС. Во-вторых, значимость объяснений обусловлена тем, что ЭС предназначены для использования в слабо формализованных областях, т. е. для решения задач, не имеющих алгоритмических решений. В условиях отсутствия теории, являющейся надежной гарантией правильности полученных результатов, возникает особая необходимость в разработке средств, дающих пользователям возможность убедиться в достоверности методов и знаний, используемых ЭС для получения решения. В качестве таких средств в ЭС используются объяснительные способности.

Наиболее часто используются следующие определения термина «объяснение»:

1. Объяснение есть ответ на вопрос «Почему?».
2. Объяснение – совокупность приемов, помогающих установить достоверность суждений относительно какого-либо неясного, запутанного дела или имеющего целью вызвать более отчетливое представление о более или менее известном явлении.

Объяснить что-либо – значит раскрыть связи между объектами объяснения (фактами, событиями, процессами) и другими уже известными и ранее объясненными явлениями с целью осознания места объясняемых фактов в системе природных или социальных взаимосвязей и законов; объяснить – это значит сделать ясным, понятным, осмыслить для самого себя или растолковать кому-либо. Объяснить явление – значит показать, что оно подчиняется общему закону [12, 13].

Следует отметить, что каждое из приведенных определений охватывает различные аспекты объяснения, но не охватывает всех сторон этого явления.

Близким к понятию «объяснение» является понятие «обоснование». Обоснование – мыслительная процедура, основанная на использовании определенных знаний, норм и установок для принятия каких-либо утверждений, оценок или решений о практических действиях. Целью обоснования является обсуждение и анализ правомочности и целесообразности определенных действий (решений) в некоторой ситуации, составление возможных альтернатив и выбор из них наиболее эффективных.

Применительно к ЭС обосновать действия системы – это показать, что они являются разумными в рассматриваемой ситуации [12].

Объяснение можно охарактеризовать следующим набором параметров: цель, объект, способ, адресат. Объяснительные способности ЭС должны быть ориентированны на всех, кто с ними взаимодействует.

Специфика задач, решаемых пользователями разных типов, предъявляет к объяснительным способностям ЭС различные требования. Основная цель использования объяснительных способностей для студента – обучение, для эксперта и инженера по знаниям – локализация ошибок, для пользователя-специалиста – обеспечение доверия к результату, для пользователя неспециалиста – достижение взаимопонимания.

Итак, можно выделить следующие цели, преследуемые при использовании объяснений в ЭС:

- локализовать ошибки системы путем исследования метода рассуждения;
- повысить доверие пользователя к системе, что способствует положительной оценке пользователем пригодности системы к практическому использованию;

- достичь взаимопонимания между пользователем и системой, что повышает вероятность успешного решения поставленной пользователем задачи;
- обучить пользователя.

Параметр *«объект»* определяет объясняемую сущность. Выделение этого параметра оправдано тем, что способ объяснения часто зависит от объясняемой сущности и круг объясняемых сущностей как в существующих, так и в перспективных системах не охватывает всех знаний и процедур и необходимо указывать, что может быть объяснено, а что – нет.

Объясняемые сущности можно классифицировать по следующим аспектам:

- область интерпретации (предметная область, система, пользователь, язык);
- уровень общности (конкретные факты, абстрактные факты, метафакты);
- тип сущности (процесс, объект и т. д.);
- свойства и значения сущности (статические/динамические, точные/приближенные, полные/неполные и т. д.) и т. п.

Например, система MYCIN может давать объяснения относительно сущности двух типов: процесса построения системой умозаключений и знаний, известных системе.

При этом знания классифицируются на статические и динамические, т.к. в зависимости от этого свойства используют различные механизмы объяснений.

Параметр *«способ»* имеет несколько аспектов, и наиболее важными из них являются: тип объяснения, уровень детальности объяснения и язык объяснения.

В научной литературе обычно выделяют пять основных типов объяснения: причинные (каузальные), объяснения через закон, функциональные (целевые, мотивационные), структурные и генетические (исторические) [12].

Причинные объяснения вскрывают причинные взаимосвязи между некоторыми явлениями. Важность причинных объяснений вытекает из того, что причинность всеобща, т. к. нет явлений, которые не имели бы своих причин, как нет явлений, которые не порождали бы тех или иных следствий.

Объяснение через закон сводится к установлению, в соответствии с каким законом, теорией, моделью возникло или происходило объясняемое явление. В этом случае объяснение можно рассматривать как логическую операцию дедукции: выведение частных следствий из общего закона (теории, модели).

Например, газ ведет себя данным образом, потому что он состоит из молекул (поведение которых предсказывает кинетическая теория газа).

Функциональные объяснения сводятся к установлению функций, выполняемых той или иной частью системы. Эти объяснения строятся по принципу «*X* нужен для того, чтобы мог произойти *Y*». Например, «мимикрия (защитная окраска и форма) нужна насекомым для того, чтобы скрываться от врагов».

Структурное объяснение дается посредством описания структуры, которая обеспечивает выполнение функций объясняемой системы в целом.

Генетическое (историческое) объяснение состоит в раскрытии условий, причин и законов, приведших к текущему состоянию системы, предмета, явления. Генетическое объяснение вскрывает происхождение сущности и способ ее образования.

Параметр *«способ»* можно рассматривать и с точки зрения уровня детальности объяснений. Необходимость давать объяснения на различных уровнях детальности вызвана разнообразием целей пользователей, разнообразием их уровня знаний, изменением во времени знаний конкретного пользователя и т. п. Параметр *«способ»* включает еще один аспект – «язык объяснения». В ЭС объяснения, как правило, осуществляются на ограниченном ЕЯ или языке графических образов.

Учет параметра *«адресат»* в объяснениях ЭС затруднен по следующим причинам:

- имеются различные типы пользователей, преследующих при взаимодействии с системой различные цели;
- квалификация конкретного пользователя данного типа изменяется во времени, вследствие получения им новых знаний;
- логика работы ЭС и способ хранения знаний не идентичны представлениям пользователя, но объяснения должны быть понятными и касаться запрашиваемой сущности на соответствующих уровнях общности и детальности.

При практической реализации функции объяснительной компоненты ЭС обычно включают генерацию любых ответов на запросы пользователей.

Выделяют следующие подходы к созданию программ, обладающих объяснительными способностями:

- записанные объяснения, т. е. использование для объяснений заранее подготовленных текстов на ЕЯ;
- генерация объяснений непосредственно из программных кодов или из трека выполнения программы;
- формирование объяснений на основе моделей и принципов предметной области с использованием методов автоматического программирования.

ГЛАВА 4
ПРИМЕНЕНИЕ НЕЧЕТКОЙ ЛОГИКИ В ЭКСПЕРТНЫХ СИСТЕМАХ

4.1. Предпосылки возникновения нечеткой логики

Зачастую поиск оптимального решения практической задачи на основе классических методов математики затруднен. Причина заключается в том, что необходимо осуществить корректный подбор приемлемого аналитического описания решаемой задачи. Даже в случае успешной реализации аналитического описания поставленной задачи для ее решения могут потребоваться непомерные временные и материальные затраты. Однако существует другой подход к решению проблемы.

Дело в том, что человек способен находить оптимальные решения, пользуясь лишь абстрактными сведениями и субъективными представлениями о задаче. В жизни нам постоянно приходится оперировать неточными знаниями и формально не определенными понятиями. Разумеется, что классическая математика в таких условиях не применима. Указанные обстоятельства и привели к возникновению новой математической дисциплины – нечеткой логики, позволяющей приблизить математику к реальному миру.

Нечеткая логика (fuzzy logic) является надмножеством классической булевой логики. Она расширяет возможности классической логики, позволяя применять концепцию неопределенности в логических выводах. Под термином «нечеткая логика» фактически понимается непрерывная логика, поскольку в данном случае вместе со значениями «ложь» и «истина» применяются значения между ними [1, 2].

Как новая область математики нечеткая логика была представлена в 1960-х гг. профессором Калифорнийского университета Лотфи Заде. Первоначально нечеткая логика разрабатывалась как средство моделирования неопределенности человеческого языка.

Его основная идея состояла в том, что человеческий способ рассуждений, опирающийся на ЕЯ, не может быть описан в рамках традиционных математических понятий. Этим понятиям присуща строгая однозначность интерпретации, а все, что связано с использованием ЕЯ, имеет многозначную интерпретацию [3, 9, 13–16].

Лотфи Заде ввел понятие лингвистической переменной [1]. *Лингвистическая переменная* – это переменная, значения которой определяются набором вербальных (т. е. словесных) характеристик некоторого свойства. Например, лингвистическая переменная «возраст» определяется через набор: младенческий, детский, юношеский, молодой, зрелый, преклонный и старый.

Таким образом, основной целью введения нечеткой логики является создание аппарата, способного моделировать человеческие рассуждения и объяснять человеческие приемы принятия решений в ходе решения различных задач. В настоящее время нечеткая логика применяется при разработке систем, понимающих тексты на ЕЯ, при создании планирующих систем, опирающихся на неполную информацию, для обработки зрительных сигналов, при управлении техническими, социальными и экономическими системами, в системах ИИ и робототехнических системах.

4.2. Нечеткая логика

Одно из базовых понятий в нечеткой логике – это теория нечетких множеств. Эта теория занимается рассмотрением множеств, определяемых небинарными отношениями вхождения. В булевой логике существует только два варианта: либо элемент принадлежит множеству (степень вхождения равна 1), либо не принадлежит ему (степень вхождения равна 0). В нечеткой логике принимается во внимание степень вхождения во множество данного элемента, которая может непрерывно изменяться в интервале от 0 до 1. Указанной степени вхождения элемента во множество соответствует понятие функции принадлежности элемента множеству.

Для комбинирования нецелочисленных значений истинности в нечеткой логике определяются эквиваленты операций $И, ИЛИ, НЕТ$ [3, 14]:

$p_1 И p_2 = \min(p_1, p_2)$ (т. е. меньшее);

$p_1 ИЛИ p_2 = \max(p_1, p_2)$ (т. е. большее);

$НЕ p_1 = 1 - p_1$ (т. е. обратное значение).

Существенной в нечеткой логике является проблема взвешивания сведений. Предположим, что имеется следующий набор продукционных правил.

Правило 1:

ЕСЛИ x программирует на ЭВМ
И x получает новую информацию через Интернет,
ТО x выберет специальность по информатике.

Правило 2:

ЕСЛИ x не склонен к изучению гуманитарных наук
И x не любит доказывать теоремы,
ТО x выберет специальность по информатике.

Предположим, что мы видели, как x программировал задачу на компьютере (определенность равна 1), и вполне уверены (0,8), что x получает новую информацию через Интернет. Тогда условия, входящие в правило 1, имеют совместное значение степени истинности, равное 0,8, поскольку в случае логической функции *И* мы используем операцию $\min$.

Для правила 2 мы знаем, что «x не склонен к изучению гуманитарных наук» (0,5) и «x не любит доказывать теоремы» (степень истинности 0,25), тогда степень истинности заключения «x выберет специальность по информатике» равна 0,25 (меньшему из значений).

Таким образом, возникает проблема определения результирующей степени истинности заключения на основании приведенных двух правил. Следует отметить, что исследование таких проблем относится в большей степени к теории свидетельств, чем к нечеткой логике.

Схема, использующая свидетельства для получения степени уверенности, была предложена Шортлиффом и применяется в ЭС MYCIN. Она основывается на коэффициентах уверенности, предназначенных для измерения степени доверия к заключению, которое является результатом полученных свидетельств. Коэффициент уверенности – это разность между двумя мерами:

$$КУ[h:e] = МД[h:e] - МНД[h:e],$$

где $КУ[h:e]$ – уверенность в гипотезе h с учетом свидетельства e; $МД[h:e]$ – мера доверия гипотезе h при заданном свидетельстве e; $МНД[h:e]$ – мера недоверия h при свидетельстве e.

Коэффициент $КУ$ может изменяться от –1 (абсолютная ложь) до 1 (абсолютная истина). Значения $МД$ и $МНД$ могут изменяться только от 0 до 1. Следует отметить, что $КУ$, $МД$ и $МНД$ не являются вероятностными мерами.

Шортлиффом была предложена формула уточнения, по которой новую информацию можно сочетать со старыми результатами. Она применяется к мерам доверия и недоверия, связанным с каждой гипотезой. Формула для меры доверия имеет следующий вид:

$$МД[h:e_1, e_2] = МД[h:e_1] + МД[h:e_2](1 - МД[h:e_1]),$$

где запятая между e_1 и e_2 означает, что e_2 следует за e_1. Аналогичным образом уточняются значения меры недоверия.

Смысл формулы состоит в том, что влияние второго свидетельства e_2 на гипотезу h при заданном свидетельстве e_1 сказывается в смещении меры доверия в сторону полной определенности на расстояние, зависящее от второго свидетельства. Эта формула имеет два важных свойства:

1) она симметрична относительно следования e_1 и e_2;
2) по мере накопления подкрепляющих свидетельств *МД* (или *МНД*) движется в сторону полной определенности.

Рассмотрим пример, указывая в скобках значение *МД* для свидетельств.

Правило 1:
ЕСЛИ x программирует на ЭВМ (0,75)
И x не любит теоретические дисциплины (0,6),
ТО x выберет специальность по информатике.

Правило 2:
ЕСЛИ x любит увлекаться точными науками (0,5)
ИЛИ x любит практику на ЭВМ (0,7),
ТО x выберет специальность по информатике.

Операция *И* в первом правиле определяет минимальное из значений 0,75 и 0,6, т. е. 0,6. Операция *ИЛИ* во втором правиле требует взятия максимального из значений 0,5 и 0,7, т. е. 0,7.

Тогда гипотеза, что «x выбирает специальность по информатике» поддерживается на уровне 0,6 правилом 1 и на уровне 0,7 правилом 2. Применяя приведенную формулу, получаем

$$\text{МД [информатика: правило 1, правило 2]} =$$
$$= \text{МД [информатика: правило 1]} + \text{МД [информатика: правило 2]} \cdot$$
$$\cdot (1 - \textit{МД} [\textit{информатика}: \textit{правило}\ 1]) = 0{,}88.$$

Таким образом, объединенная мера доверия оказывается выше, чем при учете каждого свидетельства, взятого отдельно. Это согласуется с ожидаемым нами результатом, поскольку несколько показывающих одно и то же направление свидетельств подкрепляют друг друга. Следует отметить, что если поменять порядок правил 1 и 2, то на результате это не отразится. Такой набор правил с успехом использовался в ЭС MYCIN, что привело к их широкому применению в последующих разработках.

Отношение правдоподобия гипотез. Представляет интерес остановиться на применении теоремы Байеса для связывания информации, поступающей из различных источников [14]. Этот подход позволяет вычислить относительное правдоподобие конкурирующих гипотез исходя из силы свидетельств. В основе применяемого правила лежит формула

$$ОП(H:E) = P(E:H)/P(E:H'),$$

где отношение правдоподобия ($ОП$) определяется как вероятность события или свидетельства E, при условии заданной конкретной гипотезы H, деленная на вероятность этого свидетельства при условии ложности данной гипотезы (H'). Таким образом, если мы знаем вероятности свидетельства при заданной гипотезе и ее дополнение, то мы можем определить правдоподобие данной гипотезы на основе имеющегося свидетельства.

Например, если мы знаем вероятность появления отличных оценок на вступительном экзамене среди абитуриентов-медалистов и вероятность появления отличных оценок среди остальных абитуриентов, то мы сможем вычислить вероятность того, что абитуриент, сдавший вступительный экзамен на «отлично», является медалистом.

Отношение правдоподобия может быть использовано для уточнения шансов в пользу рассматриваемой гипотезы, если становится известно, что произошло событие E. Следуя [3, 14], приведем правило Байеса, используя понятие шансов. Шансы $O(A)$ за A против B при наличии некоторого события X можно записать в виде

$$O(A)=P(A/X)/P(B/X) = P(A/X)/[1-P(A/X)].$$

Полагая $O = O(A)$ и $P = P(A/X),$ получаем выражения для соотношений между величинами O и P:

$$O = P/(1-P); P = O/(1+O).$$

Байесовская схема уточнения сводится к выражению [14]

$$O'(H) = O(H)\cdot ОП(H{:}E),$$

где $O(H)$ – *априорные* шансы в пользу H, а $O'(H)$ – результирующие *апостериорные* шансы, при условии наступления события E, в соответствии с соотношением правдоподобия.

При этом информация от различных источников может учитываться простым умножением. В случае заданных априорных шансов для конкурирующих гипотез и событий, про которые известно, что они произошли, легко вычисляются апостериорные шансы, а вслед за ними и вероятности. Отношения правдоподобия получаются из двумерной таблицы, показывающей, насколько часто случается каждое событие при каждой из гипотез.

В качестве примера в табл. 4.1 содержатся данные о продолжительности жизни 100 человек. Из них 44 человека прожили более 75 лет,

а остальные – 75 лет и меньше, причем указано, кто среди них был курильщиком, а кто – нет.

Таблица 4.1

Данные продолжительности жизни

Отношение к курению	Продолжительность жизни > 75 лет	Продолжительность жизни ≤ 75 лет	*Всего*
Курящие (чел.)	20	33	53
Некурящие (чел.)	24	23	47
Всего	44	56	100

Априорные шансы в этой выборке из 100 случаев в пользу того, что человек проживет более 75 лет

$$O(\textit{Долгожитель}) = 44 / 56 = 11 / 14 = 0{,}7857\,,$$

а отношения правдоподобия

$$ОП(\textit{Долгожитель} : \textit{Курящий}) = (20 / 44) / (33 / 56) = 0{,}8815\,;$$

$$ОП(\textit{Долгожитель} : \textit{Некурящий}) = (24 / 44) / (23 / 56) = 1{,}3280.$$

Предположим, что пол также принимается во внимание как еще одна переменная, имеющая отношение к долгожительству (табл. 4.2).

Таблица 4.2

Данные продолжительности жизни в зависимости от пола

Пол	Продолжительность жизни > 75 лет	Продолжительность жизни ≤ 75 лет	*Всего*
Мужчины (чел.)	20	36	56
Женщины (чел.)	24	20	44
Всего	44	56	100

Из табл. 4.2 следуют выражения для соотношений правдоподобия для мужчин

$$ОП(\textit{Долгожитель} : \textit{Мужчина}) = (20 / 44) / (36 / 56) = 0{,}7071$$

и для женщин

$$ОП(\textit{Долгожитель} : \textit{Женщина}) = (24 / 44) / (20 / 56) = 1{,}5273.$$

Теперь, учитывая, что априорные шансы в пользу продолжительной жизни (свыше 75 лет) равны 11 / 14 = 0,7857, мы можем вычислить апостериорные шансы того, что курящий мужчина проживет долгую жизнь, пользуясь выражением

$$O'(Долгожитель) = ОП(Долгожитель : Курящий) \cdot$$
$$\cdot ОП(Долгожитель : Мужчина) \cdot O(Долгожитель) =$$
$$= 0{,}8815 \cdot 0{,}7071 \cdot 0{,}7857 = 0{,}4897.$$

Это значение соответствует вероятности 0,3288, тогда как начальная вероятность была 0,44. Таким оказался результат учета двух негативных факторов.

Отношения правдоподобия всегда положительны, причем $ОП > 1$ указывает на свидетельства в пользу гипотезы, $ОП < 1$ – против нее, а $ОП = 1$ говорит о том, что свидетельства не влияют на правдоподобие рассматриваемой гипотезы.

Множитель $ОП$ показывает, насколько более вероятной становится данная гипотеза при наличии свидетельств, чем при их отсутствии.

Если свидетельства сами по себе вызывают сомнения, то целесообразно построить такое масштабированное $ОП'$, что

$$ОП' = ОП \cdot BC + (1 - BC),$$

где BC – вероятность того, что свидетельство надежно. Например, если свидетельство известно с вероятностью $p = 0{,}8$, то отношение правдоподобия, равное 1,2 (в пользу гипотезы), согласно приведенному соотношению, уменьшится до 1,16.

Таким образом, можно сделать вывод, что отношения правдоподобия дают следующие преимущества: допускают комбинирование нескольких источников данных и возможность их корректировки, если свидетельство ненадежно.

4.3. Нечеткие подмножества

Пусть E есть множество, A – подмножество E, т. е. $A \in E$. Принадлежность любого элемента x подмножеству A можно выразить с помощью *функции принадлежности* $\mu_A(x)$, значения которой указывают, является ли (да или нет) x элементом A:

$$\mu_A(x) = 1, \text{ если } x \in A,$$
$$\mu_A(x) = 0, \text{ если } x \notin A.$$

Предположим теперь, что характеристическая функция для элементов подмножества A может принимать не только значения 0 или 1, но и любое значение $a \in [0,1]$, т. е. $\mu_A(x) = a \in [0,1]$.

Математический объект, определяемый выражением $A = \{(x_1 \mid 0{,}2), (x_2 \mid 0{,}4), (x_3 \mid 1), (x_4 \mid 0)\}$, где x_i – элемент универсального множества E, а число после вертикальной черты – значение функции

принадлежности для этого элемента, будем называть *нечетким подмножеством* множества *E*.

На рис. 4.1 приведено графическое представление нечеткого множества с помощью его функции принадлежности [3].

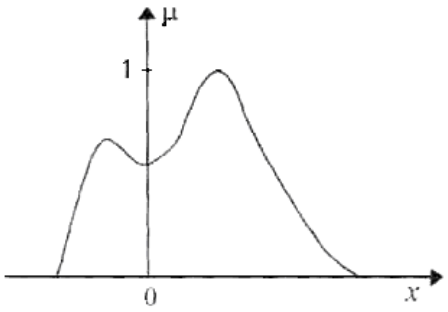


Рис. 4.1. Функция принадлежности

Строгое определение понятия нечеткого подмножества имеет следующий вид. Пусть *E* есть множество и *x* – элемент *E*. Тогда нечетким подмножеством *A* множества *E* называется множество упорядоченных пар

$$\{x|\mu_A(x)\},\forall x\in E,$$

где $\mu_A(x)$ – степень принадлежности *x* к *A*. Если $\mu_A(x)$ принимает свои значения во множестве *M* значений функции принадлежности, то можно сказать, что *x* принимает значения в *M* посредством μ_A. Множество *M* называют множеством принадлежностей.

Операции над нечеткими множествами. Рассмотрим различные операции теории обычных множеств применительно к нечетким подмножествам, а также введем новые операции для нечетких подмножеств. Пусть *E* – множество и *M* = [0,1] – множество принадлежностей, *A* и *B* – два нечетких подмножества из *E*.

Равенство. Два нечетких подмножества *A* и *B* равны (обозначается *A*=*B*) тогда и только тогда, когда

$$(A = B) \Leftrightarrow (\forall x \in E : \mu_A(x) = \mu_B(x)).$$

Если найдется, по крайней мере, один такой элемент *x* из *E*, что равенство $\mu_A(x) = \mu_B(x)$ не удовлетворяется, то *A* и *B* не равны ($A \neq B$).

Пересечение. Пересечение двух нечетких подмножеств *A* и *B*, обозначаемое $A \cap B$, определяют как наибольшее нечеткое подмножество, содержащееся одновременно в *A* и *B*.

$$(A \cap B) = (\forall x \in E : \mu_{A\cap B}) = MIN(\mu_A(x), \mu_B(x)).$$

На рис. 4.2 графически представлено пересечение двух нечетких подмножеств.

Объединение. Объединение двух нечетких подмножеств A и B, $A \cup B$ определим как наименьшее нечеткое подмножество, которое содержит как A, так и B:

$$(A \cup B) = (\forall x \in E : \mu_{A \cup B}) = MAX(\mu_A(x), \mu_B(x)).$$

На рис. 4.3 графически представлено объединение двух нечетких подмножеств.

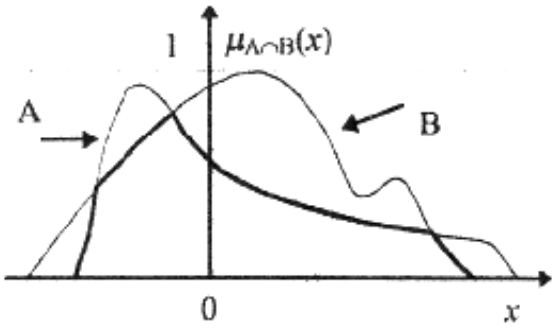


Рис. 4.2. Пересечение двух нечетких подмножеств

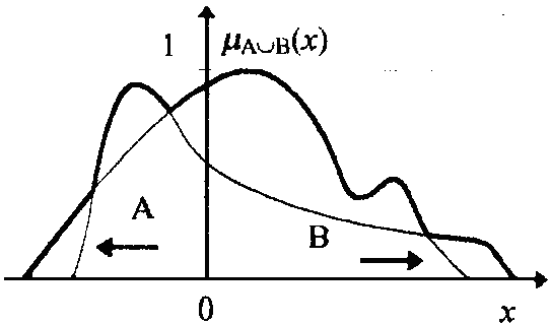


Рис. 4.3. Объединение двух нечетких подмножеств

Дополнение. Будем говорить, что A и B – два нечетких подмножества E дополняют друг друга, если

$$\forall x \in E : \mu_A(x) = 1 - \mu_B(x).$$

Это обозначается следующим образом:

$$B = \neg A \text{ или } A = \neg B.$$

На рис. 4.4 графически представлено дополнение нечеткого подмножества A.

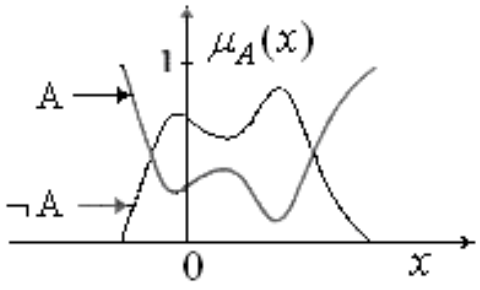


Рис. 4.4. Дополнение нечеткого подмножества

Дизъюнктивная сумма. Дизъюнктивная сумма двух нечетких подмножеств определяется в терминах объединений и пересечений следующим образом [3]:

$$A \oplus B = (A \cap \neg B) \cup (\neg A \cap B).$$

Пример:

$$A = \{(x_1 \mid 0{,}1), (x_2 \mid 0{,}5), (x_3 \mid 1), (x_4 \mid 0), (x_5 \mid 0{,}8)\}.$$

$$B = \{(x_1 \mid 0{,}6), (x_2 \mid 1), (x_3 \mid 0{,}4), (x_4 \mid 0{,}7), (x_5 \mid 0{,}8)\}.$$

$$\neg A = \{(x_1 \mid 0{,}9), (x_2 \mid 0{,}5), (x_3 \mid 0), (x_4 \mid 1), (x_5 \mid 0{,}2)\}.$$

$$\neg B = \{(x_1 \mid 0{,}4), (x_2 \mid 0), (x_3 \mid 0{,}6), (x_4 \mid 0{,}3), (x_5 \mid 0{,}2)\}.$$

$$A \cap \neg B = \{(x_1 \mid 0{,}1), (x_2 \mid 0), (x_3 \mid 0{,}6), (x_4 \mid 0), (x_5 \mid 0{,}2)\}.$$

$$\neg A \cap B = \{(x_1 \mid 0{,}6), (x_2 \mid 0{,}5), (x_3 \mid 0), (x_4 \mid 0{,}7), (x_5 \mid 0{,}2)\}.$$

$$A \oplus B = \{(x_1 \mid 0{,}6), (x_2 \mid 0{,}5), (x_3 \mid 0{,}6), (x_4 \mid 0{,}7), (x_5 \mid 0{,}2)\}.$$

Разность двух подмножеств определяется соотношением

$$A - B = (A \cap \neg B).$$

Используя данные, приведенные в предыдущем примере, получаем

$$A - B = (A \cap \neg B) = \{(x_1 \mid 0{,}1), (x_2 \mid 0), (x_3 \mid 0{,}6), (x_4 \mid 0), (x_5 \mid 0{,}2)\}.$$

Перемещение. Операция перемещения изменяет значения на величину λ. При $\lambda > 0$ производится перемещение функции вправо, а при $\lambda < 0$ – влево.

Соответствующее выражение имеет вид

$$\forall x \in E : \mu_B(x) \le \mu_A(x - \lambda), \lambda \in R.$$

Нормализация. Операция осуществляется в соответствии со следующей формулой:

$$\forall x \in E : \mu_B(x) = \frac{\mu_A(x)}{\max \mu_A(x)}.$$

4.4. Нечеткие правила вывода в экспертных системах

Нечеткое правило логического вывода представляет собой упорядоченную пару (*A*, *B*), где *A* – нечеткое подмножество пространства входных значений ***X***, а *B* – нечеткое подмножество пространства выходных значений ***Y.***

Например:

если цена **велика** и спрос **низкий**, то **оборот мал**, (4.4.1)

где *цена* и *спрос* – входные переменные; ***оборот*** – выходное значение; ***велика***, ***низкий*** и ***мал*** – функции принадлежности (нечеткие множества), определенные на множествах значений *цены*, *спроса* и *оборота,* соответственно [2].

Нечеткие правила вывода образуют базу правил. В нечеткой ЭС все правила работают одновременно, причем степень их влияния на выход может быть различной.

Процесс обработки нечетких правил вывода в ЭС состоит из четырех этапов:

1. Вычисление степени истинности левых частей правил (между «если» и «то») – определение степени принадлежности входных значений нечетким подмножествам, указанным в левой части правил вывода.
2. Модификация нечетких подмножеств, указанных в правой части правил вывода (после «то»), в соответствии со значениями истинности, полученными на первом этапе.
3. Объединение (суперпозиция) модифицированных подмножеств.
4. Скаляризация результата суперпозиции – переход от нечетких подмножеств к скалярным значениям.

Для определения степени истинности левой части каждого правила нечеткая ЭС вычисляет значения функций принадлежности нечетких подмножеств от соответствующих значений входных переменных. Например, для правила (1) определяется степень вхождения конкретного значения переменной *цена* в нечеткое подмножество ***велика***. Указанной степени вхождения переменной в подмножество можно поставить в соответствие истинность предиката «цена велика». К вычисленным значениям истинности могут применяться логические операции.

Полученное значение истинности используется для модификации нечеткого множества, указанного в правой части правила. Для выполнения такой модификации используют один из двух методов: «минимума» (correlation-min encoding) и «произведения» (correlation-product encoding). Первый метод ограничивает функцию принадлежности для множества, указанного в правой части правила, значением истинности левой части (рис. 4.5).

В методе «произведения» значение истинности левой части используется как коэффициент, на который умножаются значения функции принадлежности (рис. 4.6). Результатом выполнения правила является нечеткое множество.

Выходы всех правил вычисляются нечеткой ЭС отдельно. При этом в правой части некоторых правил может быть указана одна и та же нечеткая переменная. Для определения обобщенного результата необходимо учитывать все правила. С этой целью система производит суперпозицию нечетких множеств, связанных с каждой из таких переменных. Эта операция называется нечетким объединением правил вывода. Например, правая часть правил

если цена мала, то спрос велик,

если цена велика, то спрос мал

содержит одну и ту же переменную – *спрос*. Два нечетких подмножества, получаемых при выполнении этих правил, должны быть объединены ЭС [2].

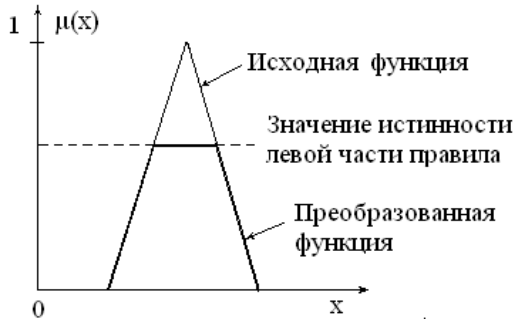


Рис. 4.5. Метод «минимума»

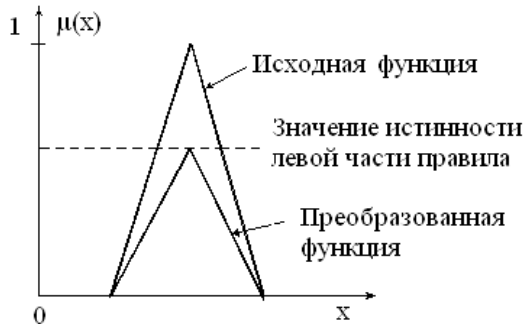


Рис. 4.6. Метод «произведения»

Суперпозиция функций принадлежности нечетких множеств в случае метода «MaxCombination» определяется следующим образом:

$$\mu_{sum}(x) = \max\{\mu_i(x)\}; \forall x, i \in [1, n].$$

На рис. 4.7 приведен пример указанной суперпозиции.

Суть метода суперпозиции «SumCombination» состоит в суммировании значений всех функций принадлежности

$$\mu_{sum}(x) = \sum_{i=1}^{n} \mu_i(x); \forall x, i \in [1, n].$$

Соответствующее графическое представление приведено на рис. 4.8.

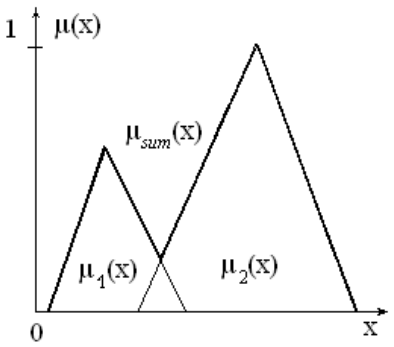


Рис. 4.7. Метод «MaxCombination»

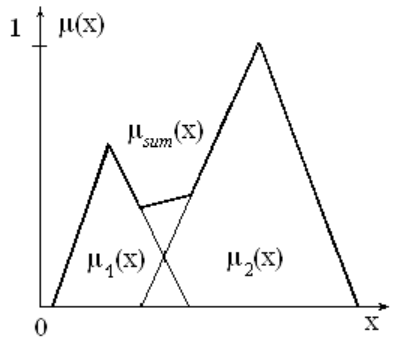


Рис. 4.8. Метод «SumCombination»

Конечным этапом обработки базы правил вывода является переход от нечетких значений к конкретным скалярным значениям. Процесс преобразования нечеткого множества в единственное значение называется «скаляризацией» или «дефазификацией» (defuzzification). Чаще всего в качестве такого значения используется «центр тяжести» функции принадлежности нечеткого множества (centroid defuzzification method). Выражения для определения значения x_c в случае непрерывной и дискретной функции принадлежности $\mu(x)$ имеют следующий вид [21]:

$$x_c = \frac{\int_x \mu(x) x dx}{\int_x \mu(x) dx};$$

$$x_c = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)}.$$

На рис. 4.9 приведен графический пример скаляризации методом «центра тяжести».

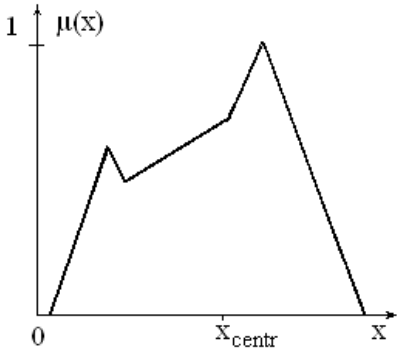


Рис. 4.9. Скаляризация методом «центра тяжести»

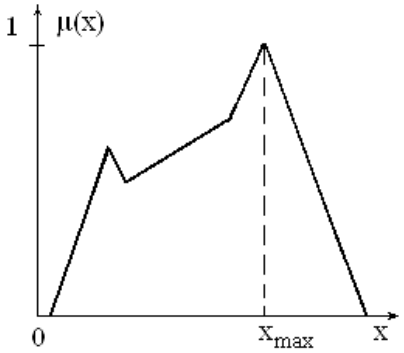


Рис. 4.10. Скаляризация методом «максимума»

Другим распространенным подходом скаляризации является использование максимального значения функции принадлежности (modal defuzzification method) (рис. 4.10). Следует отметить, что выбор методов суперпозиции и скаляризации осуществляется в каждом конкретном случае в зависимости от желаемого поведения нечеткой ЭС [2, 21].

4.5. Задания для лабораторных работ

Целью лабораторной работы является создание студентом ЭС. Для реализации ЭС рекомендуется использование языков и сред программирования: Prolog, C ++, Delphi.

Ниже излагаются варианты тем для разработки ЭС. Выбор варианта производится в соответствии с желанием студента, на основании его знаний о предметной области.

Варианты заданий:

1. ЭС, рекомендующая распределение времени при подготовке к экзаменам.
2. ЭС по выбору темы для бакалаврской работы.
3. ЭС по диагностике состояния здоровья пациента.

4. ЭС по выбору вуза и специальности для абитуриента.
5. ЭС, определяющая тип темперамента человека.
6. ЭС по выбору маршрута и способа передвижения из одного населенного пункта в другой.
7. ЭС по принятию финансовых решений в области малого предпринимательства.
8. ЭС по выбору места работы после окончания ТПУ.
9. ЭС, определяющая неисправность автомобиля и дающая рекомендации по ее устранению.
10. ЭС по выбору автомобиля.
11. ЭС для принятия решения о приеме на работу в компьютерную фирму нового сотрудника.
12. ЭС поиска неисправностей в компьютере.
13. ЭС по выбору стиральной машины.
14. ЭС, рекомендующая конфигурацию персонального компьютера.
15. ЭС, прогнозирующая исход футбольного матча.
16. ЭС по выбору системы защиты информации.
17. ЭС оценки качества программного обеспечения.
18. ЭС, принимающая решения о формировании бюджета семьи.
19. ЭС по определению оптимального маршрута движения автомобиля «Скорой помощи» по вызовам.
20. ЭС по определению типа геологической породы.
21. ЭС, рекомендующая конфигурацию сервера локальной вычислительной сети.
22. ЭС по выбору инструментальных средств при создании web-сайтов.

Отчет по лабораторной работе должен содержать:

- цель работы;
- постановку задачи;
- метод решения задачи;
- структурную схему алгоритма;
- листинг программы;
- результаты работы экспертной системы;
- выводы;
- список использованной литературы.

ГЛАВА 5
ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

5.1. Введение

Развитие природных систем на протяжении многих веков привлекало внимание ученых. Неоднократно совершались попытки выделить и осмыслить основополагающие принципы и механизмы, лежащие в основе изменений, происходящих в живой природе. Предлагалось множество различных концепций, пока в 1858 г. Чарльз Дарвин не опубликовал свою знаменитую работу «Происхождение видов», в которой были провозглашены принципы наследственности, изменчивости и естественного отбора. Однако на протяжении почти 100 последующих лет оставались неясными механизмы, отвечающие за наследственность и изменчивость организмов. В 1944 г. Эйвери, Маклеод и Маккарти опубликовали результаты своих исследований, доказывавших, что за наследственные процессы ответственна «кислота дезоксирибозного типа». Это открытие послужило толчком к многочисленным исследованиям во всем мире, и 27 апреля 1953 г. в журнале «Nature» вышла статья Уотсона и Крика, где была описана модель двухцепочечной спирали ДНК.

Знание эволюционных принципов и генетических основ наследственности позволило разработать как модели молекулярной эволюции [23], описывающие динамику изменения молекулярных последовательностей, так и макроэволюционные модели, используемые в экологии, истории и социологии для исследования экосистем и сообществ организмов [23, 24].

Эволюционные принципы используются не только для моделирования, но и для решения прикладных задач оптимизации. Множество алгоритмов и методов, использующих для поиска решения эволюционный подход, объединяют под общим названием *эволюционные вычисления* (ЭВ) или *эволюционные алгоритмы* (ЭА) [25]. Существуют следующие основные виды ЭА:

- генетический алгоритм [26, 27];
- эволюционное программирование [28];
- эволюционные стратегии [29, 30];
- генетическое программирование [31].

В данной главе будет рассмотрен генетический алгоритм (ГА) как один из самых распространенных эволюционных алгоритмов. Краткое описание видов ЭА приведено в [32].

Круг задач, решаемых с помощью ГА, очень широк. Ниже перечислены некоторые задачи, для решения которых используются ГА [27, 33, 36]:

- задачи численной оптимизации;
- задачи о кратчайшем пути;
- задачи компоновки;
- составление расписаний;
- аппроксимация функций;
- отбор (фильтрация) данных;
- настройка и обучение искусственной нейронной сети;
- искусственная жизнь;
- биоинформатика;
- игровые стратегии;
- нелинейная фильтрация;
- развивающиеся агенты/машины.

Сама идея применения эволюционных принципов для машинного обучения присутствует также и в известном труде Тьюринга, посвященном проблемам создания «мыслящих» машин [37].

5.2. Генетический алгоритм

Идея генетических алгоритмов предложена Джоном Холландом в 60-х гг., а результаты первых исследований обобщены в его монографии «Адаптация в природных и искусственных системах» [26], а также в диссертации его аспиранта Кеннета Де Йонга [34].

Как уже говорилось выше, ГА используют для работы эволюционные принципы наследственности, изменчивости и естественного отбора. Общая схема ГА представлена на рис. 5.1 [40].

ГА работает с *популяцией особей*, в *хромосоме (генотипе)* каждой из которых закодировано возможное решение задачи (*фенотип*). В начале работы алгоритма популяция формируется случайным образом (блок «Формирование начальной популяции» на рис. 5.1). Для того чтобы оценить качество закодированных решений, используют *функцию приспособленности*, которая необходима для вычисления *приспособленности* каждой особи (блок «Оценивание популяции»). По результатам оценивания особей наиболее приспособленные из них выбираются (блок «Селекция») для скрещивания. В результате *скрещивания* выбранных особей посредством применения генетического оператора *кроссинговера* создается *потомство*, генетическая информация которого формируется в результате обмена хро-

мосомной информацией между родительскими особями (блок «Скрещивание»). Созданные потомки формируют новую популяцию, причем часть потомков *мутирует* (используется генетический оператор *мутации*), что выражается в случайном изменении их генотипов (блок «Мутация»). Этап, включающий последовательность «Оценивание популяции» – «Селекция» – «Скрещивание» – «Мутация», называется *поколением*. Эволюция популяции состоит из последовательности таких поколений.

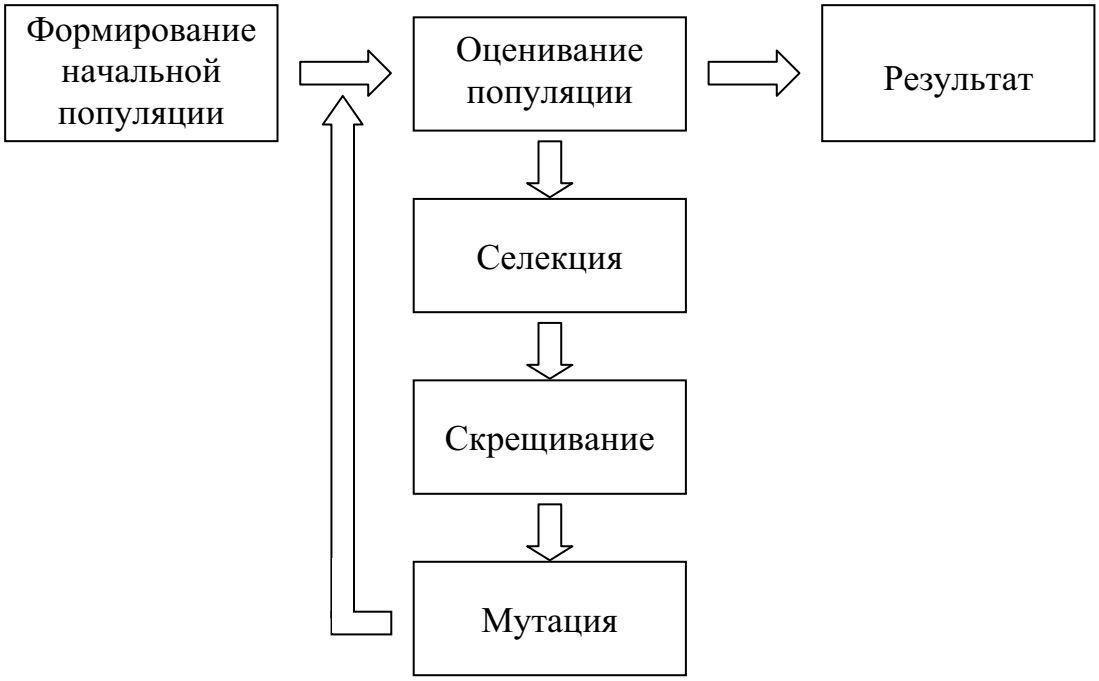


Рис. 5.1. Общая схема ГА

Длительность эволюции может определяться следующими факторами:

- нахождение решения в результате эволюционного поиска;
- ограниченность количества поколений;
- ограниченность количества вычислений функции приспособленности (целевой функции);
- вырождение популяции, когда степень разнородности хромосом в популяции становится меньше допустимого значения.

5.3. Параметры и этапы генетического алгоритма

5.3.1. Кодирование информации и формирование популяции

Выбор способа кодирования является одним из важнейших этапов при использовании эволюционных алгоритмов. В частности, должно выполняться следующее условие: должна быть возможность закодировать (с допустимой погрешностью) в хромосоме любую точку из рассматриваемой области пространства поиска.

Невыполнение этого условия может привести как к увеличению времени эволюционного поиска, так и к невозможности найти решение поставленной задачи.

Как правило, в хромосоме кодируются численные параметры решения. Для этого возможно использование целочисленного и вещественного кодирования.

Целочисленное кодирование. В классическом ГА хромосома представляет собой битовую строку, в которой закодированы параметры решения поставленной задачи. На рис. 5.2 показан пример кодирования четырех 10-разрядных параметров в 40-разрядной хромосоме.

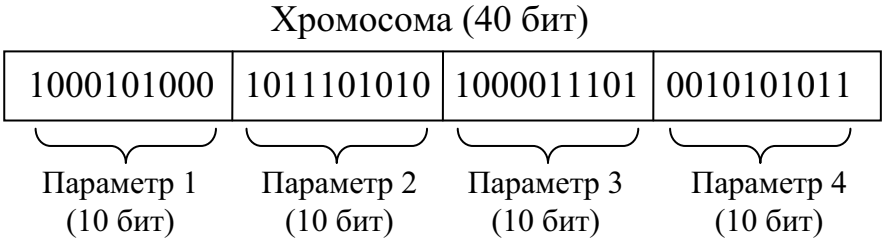


Рис. 5.2. Пример целочисленного кодирования

Как правило, считают, что каждому параметру соответствует свой *ген*. Таким образом, можно также сказать, что хромосома на рис. 5.2 состоит из четырех 10-разрядных генов. Несмотря на то что каждый параметр закодирован в хромосоме целым числом (в виде двоичной последовательности), ему могут быть поставлены в соответствие и вещественные числа. Ниже представлен один из вариантов прямого и обратного преобразования «целочисленный ген → вещественное число».

Если известен диапазон $[x_{\min}; x_{\max}]$, в пределах которого лежит значение параметра и m – разрядность гена, то этот диапазон разбивают на 2^m равных отрезков, и каждому отрезку соответствует определенное значение гена. При этом для перевода значений из закодированного значения в дробные применяют следующие формулы:

$$r = \frac{g(x_{\max} - x_{\min})}{2^m - 1} + x_{\min},$$

$$g = \frac{(r - x_{\min})(2^m - 1)}{(x_{\max} - x_{\min})},$$

где r – вещественное (декодированное) значение параметра; g – целочисленное (закодированное) значение параметра.

Например, если искомое значение параметра лежит в промежутке [1; 2] и каждый ген кодируется 16 разрядами, то (при содержимом гена $ABCD_{16} = 43981_{10}$) соответствующее дробное значение равно

$$r = 43981 \cdot (2 - 1) / (2^{16} - 1) + 1 = 0{,}6711 + 1 = 1{,}6711.$$

Если же декодированное значение равно 1,3275, то соответствующий ген после обратного преобразования будет содержать (с округлением в меньшую сторону)

$$g = (1{,}3275 - 1)(2^{16} - 1) / (2 - 1) = 0{,}3275 \cdot 65535 = 21462{,}7125 \approx$$
$$\approx 21462_{10} = 0101\ 0011\ 1101\ 0110_2.$$

Вещественное кодирование. Часто бывает удобнее кодировать в гене не целое число, а вещественное. Это позволяет избавиться от операций кодирования/декодирования, используемых в целочисленном кодировании, а также увеличить точность найденного решения. Пример вещественного кодирования представлен на рис. 5.3.

Формирование начальной популяции. Как правило, начальная популяция формируется случайным образом. При этом гены инициализируются случайными значениями. Пример случайной инициализации популяции на псевдоязыке представлен ниже.

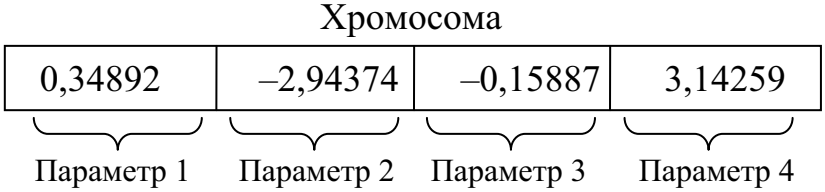


Рис. 5.3. Пример вещественного кодирования

```
i = 0;
ПОКА (i < РАЗМЕР_ПОПУЛЯЦИИ) {
j = 0;
ПОКА (j < ЧИСЛО_ГЕНОВ) {
ОСОБЬ[i].ГЕН[j] = СЛУЧАЙНАЯ_ВЕЛИЧИНА;
j = j+1;
}
i = i+1;
}
```

5.3.2. Оценивание популяции

Оценивание популяции необходимо для того, чтобы выявить в ней более приспособленные и менее приспособленные особи. Для подсчета приспособленности каждой особи используется функция приспособленности (*целевая функция*)

$$f_i = f(\mathbf{G}_i),$$

где $\mathbf{G}_i = \{ g_{ik} : k = 1,2\ldots N \}$ – хромосома i-й особи; g_{ik} – значение k-го гена i-й особи; N – количество генов в хромосоме. В случае использования целочисленного кодирования (см. предыдущий раздел) для вычисления значения функции приспособленности часто бывает необходимо преобразовать закодированные в хромосоме целочисленные значения к вещественным числам. Другими словами,

$$f_i = f(\mathbf{X}_i),$$

где $\mathbf{X}_i = \{ x_{ik} : k = 1,2\ldots N \}$ – вектор вещественных чисел, соответствующих генам i-й хромосомы.

Как правило, использование эволюционного алгоритма подразумевает решение задачи максимизации (минимизации) целевой функции, когда необходимо найти такие значения параметров функции f, при которых значение функции максимально (минимально). В соответствии с этим, если решается задача минимизации и $f(\mathbf{G}_i) < f(\mathbf{G}_j)$, то считают, что i-я особь лучше (приспособленнее) j-й особи. В случае задачи максимизации, наоборот, если $f(\mathbf{G}_i) > f(\mathbf{G}_j)$, то i-я особь считается более приспособленной, чем j-я особь.

5.3.3. Селекция

Селекция (*отбор*) необходима, чтобы выбрать более приспособленных особей для скрещивания. Существует множество вариантов селекции, опишем наиболее известные из них.

Рулеточная селекция. В данном варианте селекции вероятность i-й особи принять участие в скрещивании p_i пропорциональна значению ее приспособленности f_i и равна

$$p_i = \frac{f_i}{\sum_j f_j}.$$

Процесс отбора особей для скрещивания напоминает игру в «рулетку». Рулеточный круг делится на сектора, причем площадь i-го сектора пропорциональна значению p_i. После этого n раз «вращается» рулетка, где n – размер популяции, и по сектору, на котором останавливается рулетка, определяется особь, выбранная для скрещивания.

Селекция усечением. При отборе усечением после вычисления значений приспособленности для скрещивания выбираются ln лучших особей, где l – «порог отсечения», $0 < l < 1$, n – размер популяции. Чем меньше значение l, тем сильнее давление селекции, т. е. меньше шансы на выживание у плохо приспособленных особей. Как правило, выбирают l в интервале от 0,3 до 0,7.

Турнирный отбор. В случае использования турнирного отбора для скрещивания, как и при рулеточной селекции, отбираются n особей.

Для этого из популяции случайно выбираются t особей, и самая приспособленная из них допускается к скрещиванию. Говорят, что формируется турнир из t особей, t – размер турнира. Эта операция повторяется n раз. Чем больше значение t, тем больше давление селекции. Вариант турнирного отбора, когда $t = 2$, называют бинарным турниром. Типичные значения размера турнира $t = 2, 3, 4, 5$.

5.3.4. Скрещивание и формирование нового поколения

Отобранные в результате селекции особи (называемые *родительскими*) скрещиваются и дают потомство. Хромосомы потомков формируются в процессе обмена генетической информацией (с применением оператора *кроссинговера*) между родительскими особями. Созданные таким образом потомки составляют популяцию следующего поколения. Ниже будут описаны основные операторы кроссинговера для целочисленного и вещественного кодирования. Будем рассматривать случай, когда из множества родительских особей случайным образом выбираются две особи и скрещиваются с вероятностью P_C, в результате чего создаются два потомка. Этот процесс повторяется до тех пор, пока не будет создано n потомков. Вероятность скрещивания P_C является одним из ключевых параметров генетического алгоритма и в большинстве случаев ее значение находится в диапазоне от 0,6 до 1. Процесс скрещивания на псевдоязыке выглядит следующим образом (предполагается, что размер подпопуляции родительских особей равен размеру популяции, `RANDOM` – случайное число из диапазона [0; 1]):

```
k = 0;
ПОКА (k < РАЗМЕР_ПОПУЛЯЦИИ) {
i = RANDOM * РАЗМЕР_ПОПУЛЯЦИИ;
j = RANDOM * РАЗМЕР_ПОПУЛЯЦИИ;
ЕСЛИ (PC > RANDOM) {
СКРЕЩИВАНИЕ (РОДИТЕЛЬ[i], РОДИТЕЛЬ[j],
ПОТОМОК[k], ПОТОМОК[k+1]);
k = k+2;
} ИНАЧЕ {
ПОТОМОК[k] = РОДИТЕЛЬ[i];
ПОТОМОК[k+1] = РОДИТЕЛЬ[j];
}
}
```

Целочисленное кодирование. Для целочисленного кодирования часто используются 1-точечный, 2-точечный и однородный операторы кроссинговера.

1-точечный кроссинговер работает аналогично операции перекреста для хромосом при скрещивании биологических организмов. Для этого

выбирается произвольная точка разрыва и для создания потомков производится обмен частями родительских хромосом. Иллюстративный пример работы 1-точечного кроссинговера представлен на рис. 5.4*а*.

Для оператора *2-точечного кроссинговера* выбираются две случайные точки разрыва, после чего для создания потомков родительские хромосомы обмениваются участками, лежащими между точками разрыва (рис. 5.4*б*). Отметим, что для 2-точечного оператора кроссинговера начало и конец хромосомы считаются «склеенными», в результате чего одна из точек разрыва может попасть в начало/конец хромосом и в таком случае результат работы 2-точечного кроссинговера будет совпадать с результатом работы 1-точечного кроссинговера [34]. На рис. 5.4*б* точка разрыва в месте склеивания хромосом показана пунктирными стрелками.

При использовании *однородного оператора кроссинговера* разряды родительских хромосом наследуются независимо друг от друга. Для этого определяют вероятность p_0, что i-й разряд хромосомы 1-го родителя попадет к первому потомку, а 2-го родителя – ко второму потомку. Вероятность противоположного события равна $(1 - p_0)$. Каждый разряд родительских хромосом «разыгрывается» в соответствии со значением p_0 между хромосомами потомков. В большинстве случаев вероятность обоих событий одинакова, т. е. $p_0 = 0{,}5$.

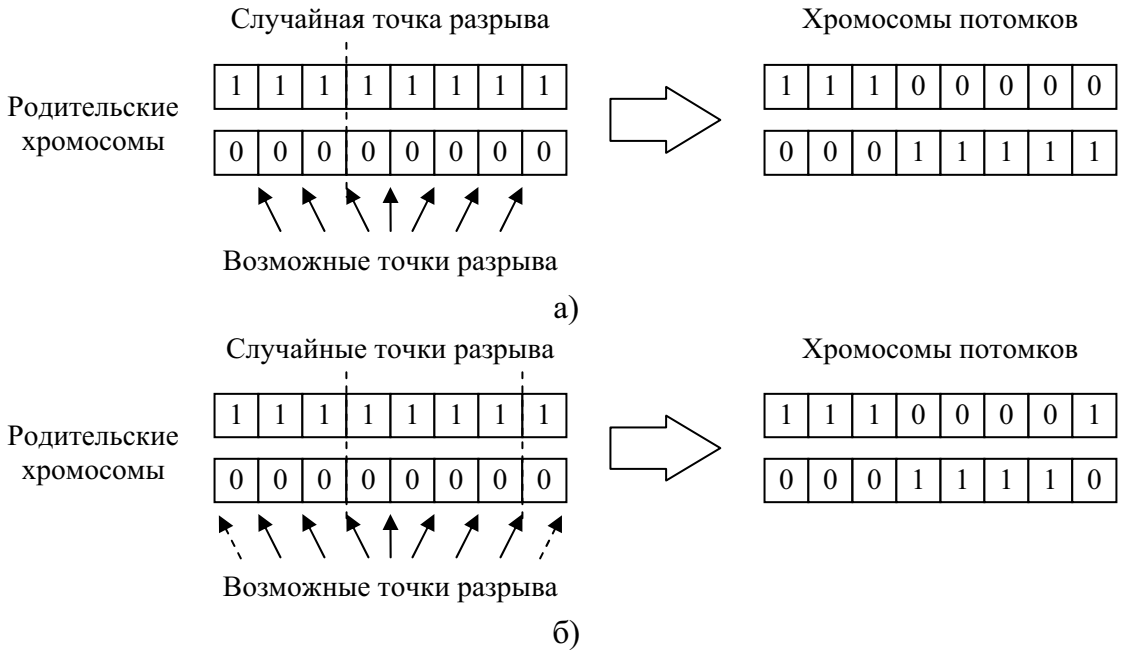


Рис. 5.4. Примеры работы:
а) 1-точечного оператора кроссинговера;
б) 2-точечного оператора кроссинговера

Вещественное кодирование. Для вещественного кодирования рассмотрим 2-точечный, арифметический и *BLX-*α операторы кроссинговера.

2-точечный кроссинговер для вещественного кодирования в целом аналогичен 2-точечному кроссинговеру для целочисленного кодирования. Различие заключается в том, что точка разрыва не может быть выбрана «внутри» гена, а должна попасть между генами (рис. 5.5).

При использовании арифметического и *BLX-*α операторов кроссинговера обмен информацией между родительскими особями и потомками производится с учетом значений генов родителей.

Обозначим $g_k^{(1)}$ и $g_k^{(2)}$ – k-е гены родительских особей, $1 \le k \le N$, N – количество генов в хромосоме. Пусть также $h_k^{(1)}$ и $h_k^{(2)}$ – k-е гены потомков. Тогда для арифметического кроссинговера

$$h_k^{(1)} = \lambda g_k^{(1)} + (1-\lambda) g_k^{(2)};$$

$$h_k^{(2)} = \lambda g_k^{(2)} + (1-\lambda) g_k^{(1)},$$

где $0 \le \lambda \le 1$.

Если используется *BLX-*α кроссинговер, то значение k-го гена потомка выбирается случайным образом (равномерное распределение) из интервала $[c_{min} - \alpha\Delta_k, c_{max} + \alpha\Delta_k]$, где α – константа,

$$c_{\min} = \min\{g_k^{(1)}, g_k^{(2)}\};$$

$$c_{\max} = \max\{g_k^{(1)}, g_k^{(2)}\};$$

$$\Delta_k = c_{\max} - c_{\min}.$$

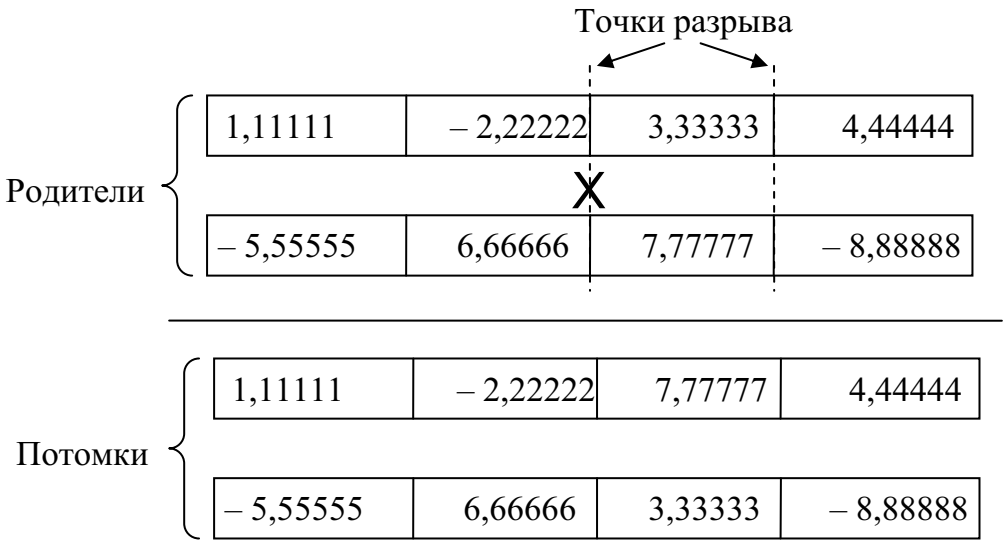


Рис. 5.5. Пример работы 2-точечного кроссинговера для вещественного кодирования

Изображение интервала, используемого для *BLX-*α кроссинговера, показано на рис. 5.6.

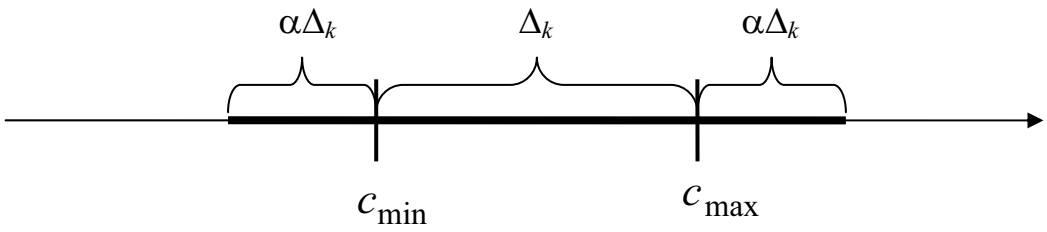


Рис. 5.6. Интервал для BLX-α кроссинговера

Разрушающая способность кроссинговера. Операторы кроссинговера характеризуются *способностью к разрушению* (*dispurtion*) родительских хромосом. Кроссинговер для целочисленного кодирования считается более разрушительным, если в результате его применения расстояние по Хэммингу между получившимися хромосомами потомков и хромосомами родителей велико. Другими словами, способность целочисленного кроссинговера к разрушению зависит от того, насколько сильно он «перемешивает» (рекомбинирует) содержимое родительских хромосом. Так, 1-точечный кроссинговер считается слабо разрушающим, а однородный кроссинговер в большинстве случаев является сильно разрушающим оператором. Соответственно, 2-точечный кроссинговер по разрушающей способности занимает промежуточную позицию по отношению к 1-точечному и однородному операторам кроссинговера.

В случае кроссинговера для вещественного кодирования способность к разрушению определяется тем, насколько велико расстояние в пространстве поиска между точками, соответствующими хромосомам родителей и потомков. Таким образом, разрушающий эффект 2-точечного кроссинговера зависит от содержимого родительских хромосом. Разрушающая способность арифметического кроссинговера зависит от значения параметра λ, например, при $\lambda \to 1$ и $\lambda \to 0$, способность к разрушению будет низкой. Для *BLX-*α кроссинговера разрушающая способность зависит как от значения α, так и от разности значений соответствующих генов родительских особей.

Отметим, что одновременно со способностью к разрушению говорят также о способности к созданию (*creation, construction*) кроссинговером новых особей. Тем самым подчеркивается, что, разрушая хромосомы родительских особей, кроссинговер может создать совершенно новые хромосомы, не встречавшиеся ранее в процессе эволюционного поиска.

Формирование нового поколения. Как уже упоминалось выше, в результате скрещивания создаются потомки, которые формируют популяцию следующего поколения.

Отметим, что обновленная таким образом популяция не обязательно должна включать одних только особей-потомков. Пусть доля обнов-

ляемых особей равна T, $0 < T < 1$, тогда в новое поколение попадает Tn потомков, n – размер популяции, а $(1 - T)n$ особей в новой популяции являются наиболее приспособленными родительскими особями (так называемые *элитные особи*). Параметр T называют *разрыв поколений* (*generation gap*) [32]. Использование элитных особей позволяет увеличить скорость сходимости генетического алгоритма.

7.3.5. Мутация

Оператор *мутации* используется для внесения случайных изменений в хромосомы особей. Это позволяет «выбираться» из локальных экстремумов и тем самым эффективнее исследовать пространство поиска. Аналогично оператору кроссинговера работа оператора мутации зависит от вероятности применения мутации P_M.

Рассмотрим базовые варианты оператора мутации в зависимости от способа представления генетической информации.

Целочисленное кодирование. Одним из основных операторов мутации для целочисленного кодирования является битовая мутация. В случае целочисленного кодирования мутация изменяет отдельные разряды в хромосоме. Для этого каждый разряд инвертируется с вероятностью P_M. Ниже приведен пример мутации на псевдоязыке:

```
ДЛЯ КАЖДОЙ k ОСОБИ В ПОПУЛЯЦИИ {
ДЛЯ КАЖДОГО i РАЗРЯДА В ХРОМОСОМЕ k {
ЕСЛИ (PM > RANDOM) {
БИТОВАЯ_МУТАЦИЯ (ОСОБЬ[k], i);
}
}
}
```

В силу того что применение мутации разыгрывается столько раз, сколько разрядов содержится в хромосоме, значение P_M выбирают небольшим, чтобы сильно не разрушать найденные хорошие хромосомы. Один из типичных вариантов $P_M = L^{-1}$, где L – длина хромосомы в битах, в этом случае каждая хромосома мутирует в среднем один раз.

Вещественное кодирование. Оператор мутации для вещественного кодирования изменяет содержимое каждого гена с вероятностью P_M. При этом величина изменения выбирается случайно в некотором диапазоне $[-\xi; +\xi]$, например, [–0,5; 0,5], и может иметь как равномерное, так и любое другое распределение, к примеру нормальное с $m_x = 0$, $\sigma_x = 0{,}5$. Таким образом, пример мутации для вещественного кодирования на псевдоязыке выглядит следующим образом (`RND` – случайное число, распределенное по заранее определенному закону):

```
ДЛЯ КАЖДОЙ k ОСОБИ В ПОПУЛЯЦИИ {
ДЛЯ КАЖДОГО i ГЕНА В ХРОМОСОМЕ k {
ЕСЛИ (PM > RANDOM) {
ОСОБЬ[k].ГЕН[i] = ОСОБЬ[k].ГЕН[i] + RND;
}
}
}
```

Для того чтобы избежать сильных изменений содержимого хромосомы в результате мутации, значение вероятности P_M выбирается небольшим. Например, $P_M = N^{-1}$, где N – количество генов в хромосоме. Также возможна адаптивная подстройка величины диапазона 2ξ изменения значения гена в результате мутации.

5.4. Настройка параметров генетического алгоритма

Результат работы генетического алгоритма сильно зависит от того, каким образом настроены его параметры. Основными параметрами ГА являются:

- длительность эволюции (количество поколений);
- размер популяции;
- интенсивность (давление) селекции;
- тип оператора кроссинговера;
- вероятность кроссинговера P_C;
- тип оператора мутации;
- вероятность мутации P_M;
- величина разрыва поколений T.

Отметим, что вышеприведенный список может быть легко расширен, но перечисленные параметры присутствуют практически в любой реализации ГА. Различные параметры влияют на разные аспекты эволюционного поиска, среди которых можно выделить два наиболее общих:

1. Исследование пространства поиска (exploration).
2. Использование найденных «хороших» решений (exploitation).

Первый аспект отвечает за способности ГА к эффективному поиску решения и характеризует способности алгоритма избегать локальных экстремумов. Второй аспект важен для постепенного улучшения имеющихся результатов от поколения к поколению на основе уже найденных «промежуточных» решений. Пренебрежение исследовательскими способностями приводит к существенному увеличению времени работы ГА и ухудшению результатов из-за «застревания» алгоритма в локальных экстремумах. В итоге становится возможной *преждевременная сходимость* генетического алгоритма (также говорят о *вырождении популяции*), когда решение еще не найдено, но в популяции практически все особи становятся одинаковыми и долгое время (порядка нескольких десятков и сотен поколений) не наблюдается улучшения приспособленности.

Игнорирование найденных решений может привести к тому, что работа ГА будет напоминать случайный поиск, что также отрицательно сказывается на эффективности поиска и качестве получаемых решений.

Основная цель в настройке параметров ГА и одновременно необходимое условие для стабильного получения хороших результатов работы алгоритма – это достижение баланса между исследованием пространства поиска и использованием найденных решений.

Взаимосвязь между параметрами генетического алгоритма, а также влияние параметров на эволюционный процесс имеют сложный характер. На рис. 5.7 схематично изображено влияние изменения некоторых параметров ГА на характеристики эволюционного поиска.

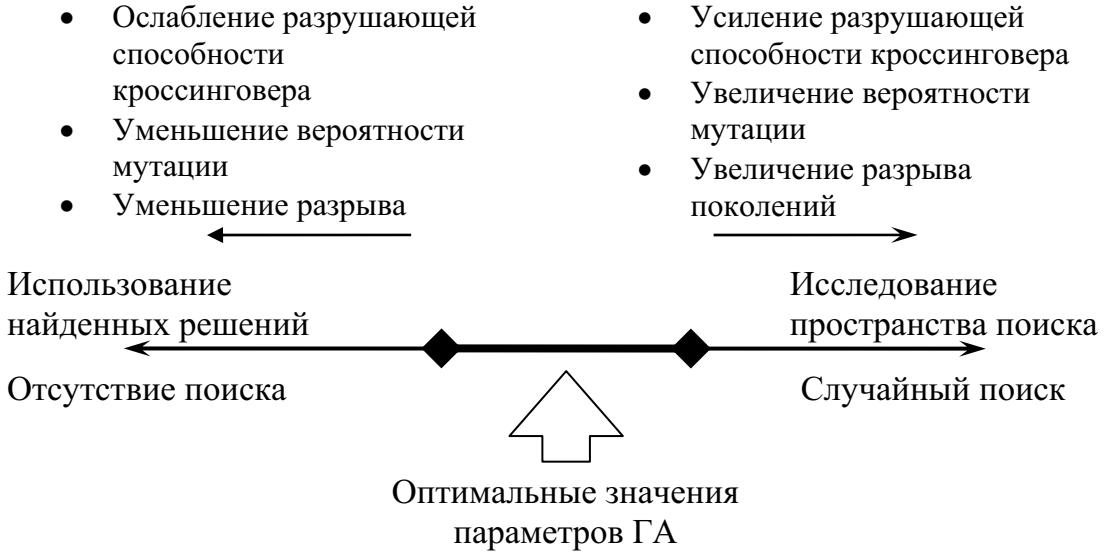


Рис. 5.7. Влияние параметров ГА на характеристики эволюционного поиска

Неправильная настройка параметров может стать причиной различных проблем в работе ГА. Краткий список часто встречающихся проблем и возможные пути их исправления приведены в табл. 5.1.

Таблица 5.1

Проблемы в работе ГА и возможные пути их исправления

Проблема	Возможные способы исправления
1. Плохая приспособленность решений.	1. Увеличение числа поколений эволюционного поиска. 2. Увеличение численности популяции. 3. Изменение критерия оценки особей. 4. Исправление способа формирования родительских пар для скрещивания. 5. Исправление стратегии скрещивания и формирования нового поколения.

Окончание табл. 5.1

Проблема	Возможные способы исправления
2. Преждевременная сходимость (вырождение популяции).	1. Изменение стратегии выбора родительских пар для скрещивания. 2. Отслеживание появления в популяции идентичных особей и их удаление. 3. Использование сильно разрушающего оператора кроссинговера. 4. Увеличение вероятности мутации.
3. Низкая «стабильность» эволюции популяции (значительные колебания значения средней приспособленности от поколения к поколению).	1. Применение «элитизма» (уменьшение разрыва поколений). 2. Уменьшение вероятности мутации. 3. Использование кроссинговера со слабой разрушающей способностью.
4. Преобладание удовлетворительных результатов над хорошими.	1. Изменение стратегии выбора родительских пар для скрещивания. 2. Изменение операторов скрещивания и/или мутации. 3. Распараллеливание поиска. Инициализация нескольких независимых популяций, которые развиваются независимо и, время от времени, обмениваются особями.

5.5. Канонический генетический алгоритм

Канонический ГА разработан Джоном Холландом и описан в его книге «Адаптация в естественных и искусственных системах» (1975 г.) [26]. Этот алгоритм представляет одну из базовых моделей эволюционного поиска, подробно исследованную в 70–80-х гг. XX в. Канонический ГА имеет следующие характеристики:

- целочисленное кодирование;
- все хромосомы в популяции имеют одинаковую длину;
- постоянный размер популяции;
- рулеточная селекция;
- одноточечный оператор кроссинговера;
- битовая мутация;
- новое поколение формируется только из особей-потомков (разрыв поколений $T = 1$).

5.6. Пример работы и анализа генетического алгоритма

При использовании ГА для решения задачи оптимизации необходимо:

1. Определить количество и тип оптимизируемых переменных задачи, которые необходимо закодировать в хромосоме.
2. Определить критерий оценки особей, задав функцию приспособленности (целевую функцию).
3. Выбрать способ кодирования и его параметры.
4. Определить параметры ГА (размер популяции, тип селекции, генетические операторы и их вероятности, величина разрыва поколений).

Отметим, что параметры ГА, определяемые в п. 4, а также (иногда) в пп. 2 и 3, часто определяются методом проб и ошибок, на основе анализа получаемых результатов. Для анализа результатов работы ГА необходимо произвести несколько запусков алгоритма, для повышения достоверности выводов о качестве получаемых результатов, т. к. результат работы ГА носит вероятностный характер. Описанная общая схема решения задачи с использованием ГА показана на рис. 5.8.

Рассмотрим пример использования ГА для решения задачи минимизации следующей функции (сферическая функция):

$$z = \sum_{i=1}^{n} x_i^2, n = 10, x_i \in [-5{,}12;\ 5{,}11]; \qquad (5.1)$$

$$z \to \min.$$

Параметр n задает количество переменных функции z. Необходимо найти такие значения переменных x_i, при которых функция z принимает наименьшее значение. Будем использовать общую схему решения (рис. 5.8):

1. Определение неизвестных переменных задачи. По условию поставленной задачи необходимо найти значения переменных x_i, минимизирующие значение функции z, поэтому в хромосоме будем кодировать значения x_i. Таким образом, каждый i-й ген хромосомы будет соответствовать i-й переменной функции z.

2. Задание функции приспособленности. Будем определять приспособленность особи в зависимости от значения, которое принимает функция z при подстановке в нее вектора параметров, соответствующих хромосоме этой особи. Поскольку рассматривается задача минимизации функции z, то будем также считать, что чем меньше значение z, тем приспособленнее особь. Приспособленность i-й особи f_i будем определять по следующей формуле:

$$f_i = z_i,$$

где z_i – значение функции z в точке, соответствующей i-й особи.

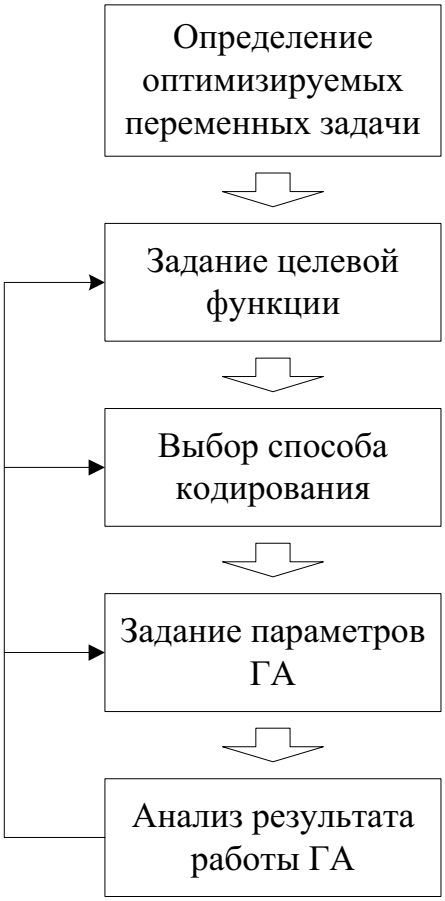


Рис. 5.8. Общая схема решения задачи с использованием ГА

3. Выбор способа кодирования. В качестве способа представления генетической информации рассмотрим целочисленное кодирование с точностью кодирования параметров 0,01. Тогда в имеющемся по условию задачи диапазоне изменения значений параметров [–5,12; 5,11] можно закодировать (5,12 – (–5,11)) / 0,01 + 1 = 1 024 различных значений переменной. Единица прибавляется, т. к. значение переменной, равное 0, также учитывается.

Для того чтобы представить 1 024 различных значений переменной, достаточно использовать $\log_2 1\,024 = 10$ бит на каждую переменную. Таким образом, будет использоваться целочисленное кодирование с 10-разрядными генами.

4. Определение параметров ГА. Для решения задачи рассмотрим популяцию из 20 особей. При отборе особей для скрещивания будем использовать турнирную селекцию с бинарным турниром. В качестве генетических операторов будем использовать 1-точечный кроссинговер и битовую мутацию. Вероятности применения операторов скрещивания и мутации установим равными 0,7 и 0,05, соответственно. Новое поколение будем формировать только из особей-потомков, т. е. величина разрыва поколений T равна 1.

Результат работы генетического алгоритма с выбранными параметрами представлен на рис. 5.9. Показаны зависимости изменения среднего <z> и наименьшего $z_{\min}$ в популяции значения функции z от номера поколения t. Данные усреднены по 100 независимым запускам.

По данным рис. 5.9 видно, что после 20-го поколения значение $z_{\min}$ колеблется в достаточно большом диапазоне. Из этого следует, что потери хороших особей в результате мутации велики и следует уменьшить вероятность мутации. Установим значение этого параметра равным $L^{-1} = 0{,}01$, где L – длина хромосомы в битах, в данном случае $L = 100$. Результаты работы ГА с измененным значением вероятности мутации показаны на рис. 5.10.

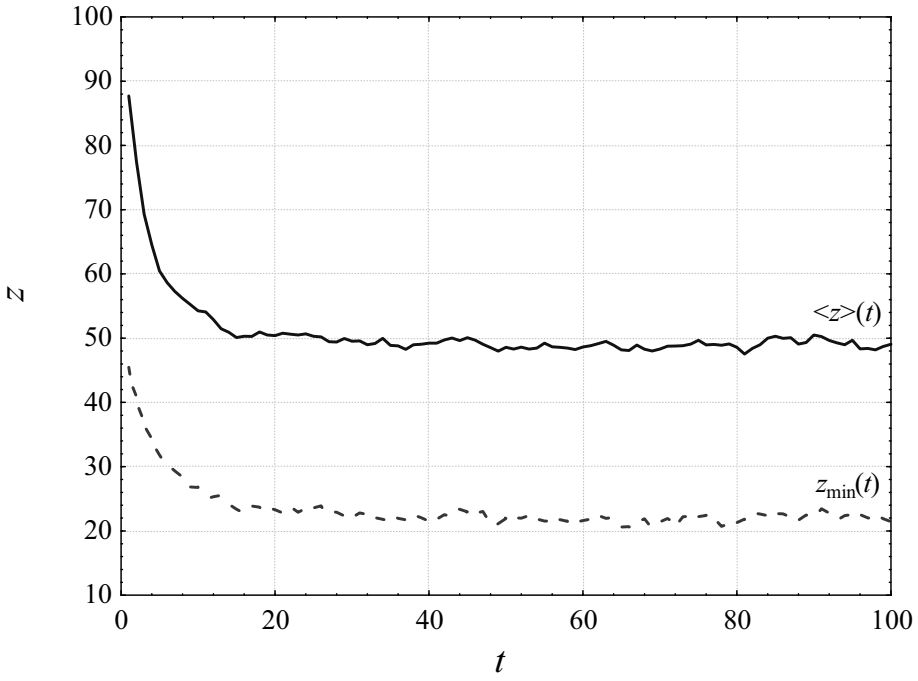


Рис. 5.9. Изменение $z_{min}(t)$ и <z>(t). Популяция из 20 особей, бинарный турнирный отбор, одноточечный кроссинговер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}05$)

Из сравнения графиков на рис. 5.9 и 5.10 следует, что уменьшение вероятности мутации улучшило результат работы ГА. Также отметим, что теперь эволюционный процесс стабилизировался значительно позднее, примерно после 60-го поколения. Усредненное по всем запускам минимальное значение функции z, достигнутое за первые 100 поколений, равно ~ 1,016. Чтобы улучшить результат, увеличим давление селекции путем увеличения размера турнира до 4. Результат представлен на рис. 5.11.

Увеличение давления селекции привело к ускорению эволюционного поиска за счет удаления из популяции особей со средней и плохой приспособленностью. В результате стабилизация наступила после 40-го поколения, а усредненное по всем запускам минимальное полученное значение

функции z равно ~0,013. Наименьшее значение функции z достигается в точке $x_i = 0$, $i = 1,2…10$ и равно 0. В случае поиска минимума функции z с точностью 0,01 для ГА с параметрами, соответствующими графикам на рис. 5.11, решение было найдено в 69 запусках из 100. При этом в среднем было использовано 1 698,68 вычислений целевой функции.

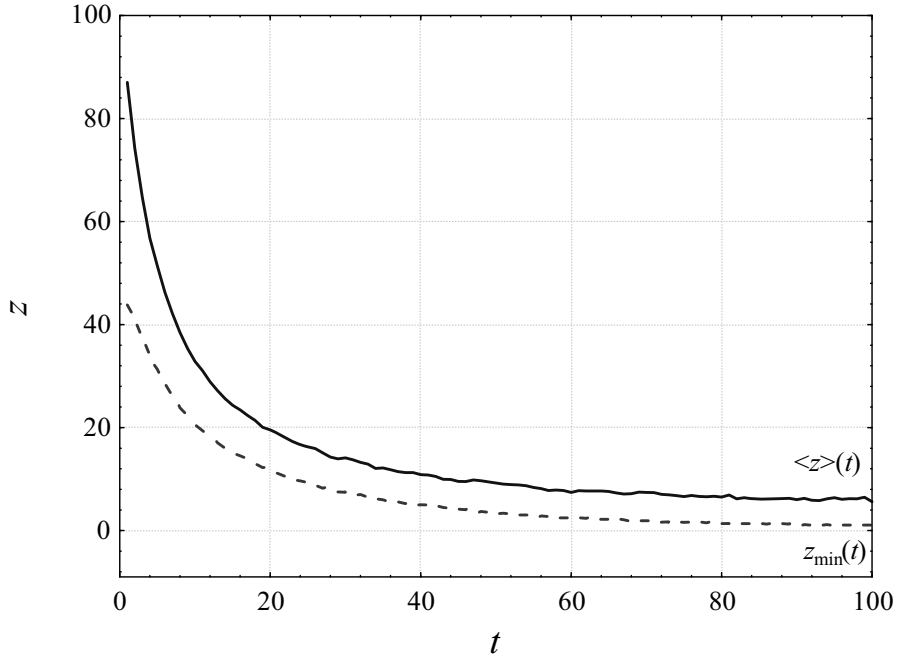


Рис. 5.10. Изменение $z_{min}(t)$ и <z>(t). Популяция из 20 особей, бинарный турнирный отбор, одноточечный кроссинговер ($P_C = 0,7$), битовая мутация ($P_M = 0,01$)

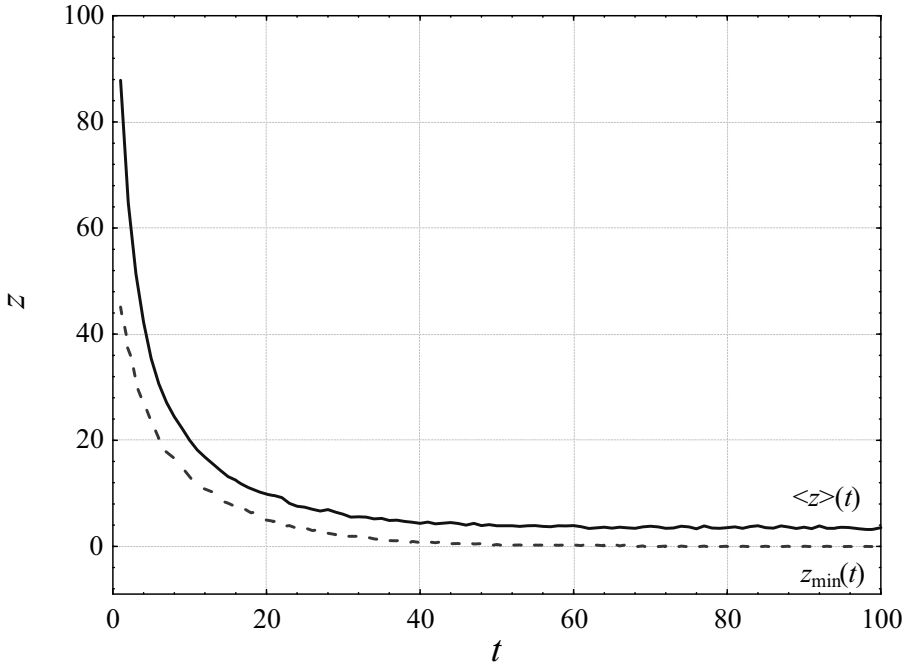


Рис. 5.11. Изменение $z_{min}(t)$ и <z>(t). Популяция из 20 особей, турнирный отбор (t = 4), одноточечный кроссинговер ($P_C = 0,7$), битовая мутация ($P_M = 0,01$)

Чтобы повысить стабильность результатов, увеличим размер популяции до 50 особей. Полученные кривые $z_{\min}(t)$ и $\langle z \rangle(t)$ изображены на рис. 5.12. Во всех 100 запусках найден минимум функции z с точностью не меньше 0,01. Среднее количество вычислений целевой функции, использованное для нахождения решения, равно 3 145,34.

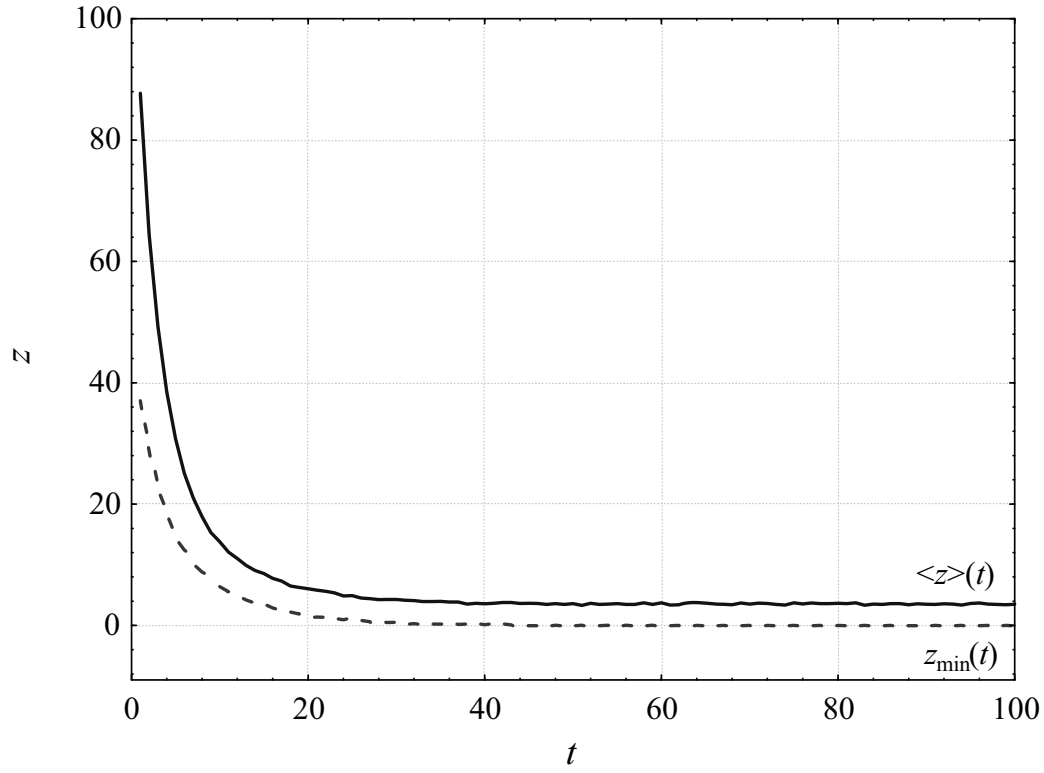


Рис. 5.12. Изменение $z_{min}(t)$ и $\langle z \rangle(t)$. Популяция из 50 особей, турнирный отбор (t = 4), одноточечный кроссинговер (P_C = 0,7), битовая мутация (P_M = 0,01)

5.7. Общие рекомендации к программной реализации генетического алгоритма

Программную реализацию ГА можно создать, используя как объектно-ориентированный, так и структурный подход. Ниже предлагается способ реализации различных компонентов ГА с использованием обоих подходов (табл. 5.2).

Приведенный в табл. 5.2 способ реализации ГА не является эталонным и, вполне возможно, далек от идеала. Данные в табл. 5.2 могут служить в качестве «опорных» для конкретной реализации генетического алгоритма. Отметим, что бóльшую гибкость и расширяемость программной реализации не только ГА, но и любого другого алгоритма и системы вообще можно достичь, используя компонентно-ориентированный подход и паттерны проектирования [35].

Таблица 5.2

Варианты реализации компонентов ГА

Компонент генетического алгоритма	Структурный подход	Объектно-ориентированный подход
Особь	Одномерный массив для записи значений генов. Размерность массива совпадает с количеством генов у одной особи (количество генов равно числу настраиваемых параметров).	Класс «Особь», содержащий массив генов.
Популяция	Двумерный массив, в котором *i*-я строка содержит гены *i*-й особи.	Отдельный класс «Популяция», содержащий одномерный массив объектов класса, представляющего особь.
Оценивание популяции	Подпрограмма оценки строк массива популяции в соответствии с выбранной целевой функцией.	Метод управляющего класса, оценивающий популяцию в соответствии с выбранной целевой функцией.
Приспособленность популяции	Одномерный массив, в котором *i*-й элемент соответствует приспособленности *i*-й особи.	Одномерный массив со значениями ошибок особей, входящий в управляющий класс.
Особи, выбранные для скрещивания	Двумерный массив, строки которого соответствуют хромосомам особей, выбранным для скрещивания.	Объект класса «Популяция», содержащий объекты класса «Особь», соответствующие выбранным особям.
Реализация скрещивания, мутации, формирования нового поколения	Подпрограммы, обрабатывающие элементы массива, представляющего популяцию особей, а также популяцию особей, выбранных для скрещивания.	Методы управляющего класса, работающие с основной популяцией и популяцией особей для скрещивания.

5.8. Задания для лабораторных работ

1. Аппроксимировать набор точек линейной функцией

$$y(x) = a \cdot x + b.$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

2. Аппроксимировать набор точек экспоненциальной функцией

$$y(x) = a \cdot \exp(b \cdot x).$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

3. Найти минимум функции

$$y(x) = x^2 + 4.$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

4. Найти максимум функции:

$$y(x) = 1 / x;\ x \in [-4; 0).$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

5. Найти точку перегиба функции

$$f(x) = (x - 1{,}5)^3 + 3.$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

6. Найти точку пересечения функции с осью Ox.

$$f(x) = \ln(x + 1) - 2{,}25,\ x > -1.$$

Вариант А. Использовать целочисленное кодирование.
Вариант Б. Использовать вещественное кодирование.

7. Сгенерировать с помощью ГА слово «МИР».

8. Найти с помощью ГА особь, гены которой соответствуют в формате RGB фиолетовому цвету (96, 96, 159).

ГЛАВА 6
ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

6.1. Биологические нейронные сети

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию, первоначально поступающую от органов чувств. Она состоит из тела клетки и двух типов внешних древоподобных ветвей: *аксона* и *дендритов*. Нейрон выполняет прием, преобразование и дальнейшую передачу информации другим нейронам. Информация переносится в виде импульсов нервной активности, имеющих электрохимическую природу [19].

Нейрон получает сигналы от других нейронов через дендриты и передает сигналы, генерированные телом клетки, вдоль аксона (играющего роль передатчика), который разветвляется на волокна. На окончаниях этих волокон находятся синапсы. *Синапс* является функциональным узлом между двумя нейронами. Он представляет собой волокно аксона одного нейрона и дендрит другого. Синапсы способны обучаться в процессе передачи через них сигналов за счет изменения своих характеристик.

Кора головного мозга человека является протяженной, образованной нейронами в слое толщиной 2–3 мм с площадью около 2 200 см^2 (что вдвое превышает площадь поверхности стандартной клавиатуры). Приблизительно 10^{11} нейронов коры головного мозга участвуют в 10^{14}–10^{15} передающих связях, протяженность которых достигает величины до одного метра и более. Каждый нейрон связан с 10^3–10^4 другими нейронами. Существует гипотеза, что степень умственного развития человека определяется не только числом нейронов, но главным образом количеством связей между ними.

Нейроны взаимодействуют посредством короткой серии импульсов, как правило, продолжительностью несколько миллисекунд. Для передачи сообщения применяется частотно-импульсная модуляция. Частота может изменяться от нескольких единиц до сотен герц, что в десятки миллионов раз медленнее, чем быстродействующие переключательные электронные схемы. Но такую сложную для компьютера задачу, как распознавание лица, человек решает за несколько сотен миллисекунд. Поскольку скорость выполнения операций нейронами составляет несколько миллисекунд, то мож-

но сделать вывод, что вычисления требуют не более 100 последовательных стадий. Для решения сложных задач мозг «запускает» программы, содержащие около 100 шагов. Оценки показывают, что количество информации, передаваемое от одного нейрона к другому, должно быть маленьким (несколько бит). По-видимому, основная информация не передается непосредственно, а захватывается и распределяется в связях между нейронами.

6.2. Формальный нейрон

Искусственные нейронные сети (ИНС) появились в результате применения математического аппарата к исследованию функционирования нервной системы [18]. Полученные при этом результаты успешно применяются при решении проблем распознавания образов, моделирования, прогнозирования, оптимизации и управления [18, 38].

Основной структурной и функциональной частью нейронной сети является *формальный нейрон* (formal neuron), представленный на рис. 6.1, где $x_0, x_1 ... x_n$ – компоненты вектора входных сигналов, $w_0, w_1 ... w_n$ – значения весов входных сигналов нейрона, а y – выходной сигнал нейрона.

Формальный нейрон состоит из элементов трех типов: умножителей (синапсов), сумматора и преобразователя. Синапс характеризует силу (вес) связи между двумя нейронами. Сумматор выполняет сложение входных сигналов, предварительно помноженных на соответствующие веса. Преобразователь реализует функцию одного аргумента – выхода сумматора. Эта функция называется функцией активации, или передаточной функцией нейрона. Исходя из данного описания, математическая модель нейрона может быть представлена следующим образом:

$$y = f(S);$$

$$S = \sum_{i=1}^{n} w_i x_i + b.$$

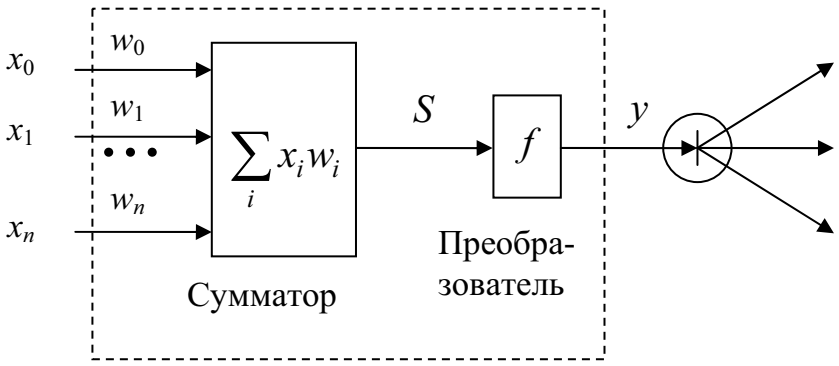


Рис. 6.1. Формальный нейрон (по [39])

Примеры некоторых активационных функций представлены в табл. 6.1 и на рис. 6.2.

Таблица 6.1

Функции активации нейронов

Название	Формула	Область значений
Пороговая	$f(S)=\begin{cases}0, S<\Theta\\1, S\geq\Theta\end{cases}$	(0, 1)
Линейная	$f(S)=aS$	$(-\infty, +\infty)$
Лог-сигмоидная	$f(S)=\frac{1}{1+e^{-aS}}$	(0, 1)
Гиперболический тангенс	$f(S)=\frac{e^{aS}-e^{-aS}}{e^{aS}+e^{-aS}}$	(–1, 1)

Описанный вычислительный элемент можно считать упрощенной математической моделью биологических нейронов. Чтобы подчеркнуть различие биологических и искусственных нейронов, вторые иногда называют *нейроподобными элементами* или *формальными нейронами*[1].

6.3 Нейронные сети

Описанные в разделе 6.1 формальные нейроны можно объединять таким образом, что выходные сигналы одних нейронов являются входными для других. Полученное множество связанных между собой нейронов называют *искусственными нейронными сетями* (artificial neural networks, ANN), или *нейронными сетями*.

Пример нейронной сети с тремя входами и одним выходом представлен на рис. 6.3: нейроны обозначены кружками, стрелками показано направление распространения сигналов, веса межнейронных связей указаны рядом с соответствующими связями.

Различают три общих типа нейронов, в зависимости от их положения в нейронной сети:

1. Входные нейроны (input nodes), на которые подаются входные для всей сети сигналы. Такие нейроны имеют, как правило, один вход с единичным весом, смещение отсутствует, а значение выхода нейрона равно входному сигналу (нейроны с индексами 0–2 на рис. 6.3).
2. Выходные нейроны (output nodes), выходные значения которых представляют результирующие выходные сигналы нейронной сети (нейрон с индексом 3 на рис. 6.3).

[1] В англоязычной литературе для обозначения нейрона в ИНС часто используют термин «node» – узел, вершина.

3. Скрытые нейроны (hidden nodes), не имеющие прямых связей с входными сигналами, при этом значения выходных сигналов скрытых нейронов не являются выходными сигналами ИНС (нейрон с индексом 4 на рис. 6.3).

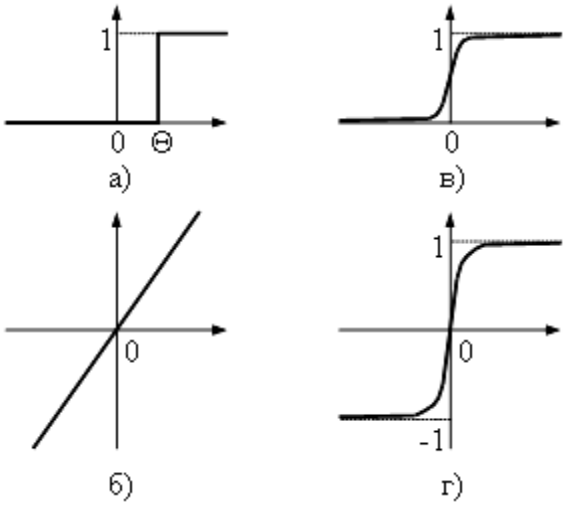


Рис. 6.2. Примеры активационных функций:
а) пороговая; б) линейная; в) лог-сигмоидная; г) гиперболический тангенс

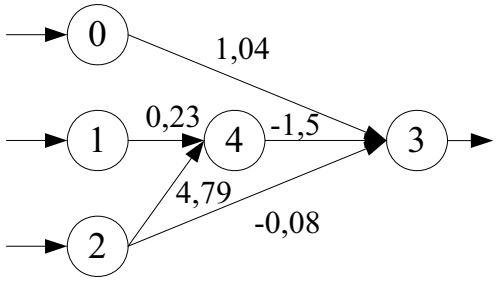
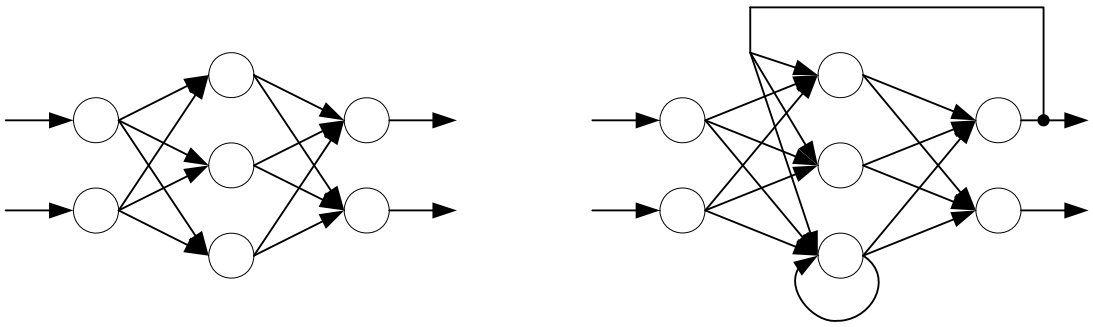


Рис. 6.3. Пример нейронной сети

Отметим, что структуру ИНС можно рассматривать как ориентированный граф, в котором узлы соответствуют нейронам, а ребра – межнейронным связям.

По структуре межнейронных связей различают два класса ИНС:

1. ИНС прямого распространения (feed-forward ANNs), в которых сигнал распространяется только от входных нейронов к выходным. Орграф, соответствующий таким ИНС, не имеет циклов и петель. Примером ИНС прямого распространения является ИНС на рис. 6.3 и 6.4*а*.
2. Рекуррентные ИНС (recurrent ANNs) – ИНС с обратными связями. В таких ИНС сигналы могут передаваться между любыми нейронами, вне зависимости от их расположения в ИНС. Орграф, соответствующий структуре рекуррентных ИНС, может иметь петли и циклы (рис. 6.4*б*).



а) б)

Рис. 6.4. Примеры структур нейронных сетей:
а) ИНС прямого распространения; б) рекуррентная ИНС

Среди различных структур ИНС наиболее известны многослойные ИНС (multi-layered ANNs) (рис. 6.5). Рассмотрим такие ИНС более подробно.

В многослойных сетях нейроны объединяются в слои таким образом, что нейроны одного слоя имеют одинаковые входные сигналы. Число нейронов в слое может быть произвольным и зависит в большей степени от решаемой задачи, чем от количества нейронов в других слоях. Внешние (входные) сигналы подаются на нейроны входного слоя (его часто нумеруют как нулевой), а выходами сети являются выходные сигналы нейронов последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети может быть один или несколько скрытых слоев. Выходные сигналы нейронов слоя (q) являются входными сигналами следующего слоя ($q + 1$). Одной из основных характеристик многослойных нейронных сетей является число слоев. В дальнейшем для описания структуры многослойных ИНС будем использовать количество присутствующих в ней скрытых слоев.

По структуре многослойные ИНС могут представлять как ИНС прямого распространения (рис. 6.4*а* и 6.5), так и рекуррентные (рис. 6.4*б*). Отметим также, что возможны многослойные сети, в которых существуют прямые связи между нейронами из несмежных слоев. Такие связи называют *перекрестными.* В данном пособии будут рассматриваться только многослойные нейронные сети без перекрестных и обратных связей.

ИНС могут быть использованы для решения широкого класса задач. Рассмотрим наиболее распространенные.

1. *Классификация* (распознавание образов). Задача классификации в ее классической постановке заключается в определении, к какому из известных классов (образов) принадлежит рассматриваемый объект при условии, что имеющиеся исходные примеры, используемые для обучения противоречивы и неполны. Примерами задач классификации являются задачи распознавания речи, рукописных символов, медицинская диагностика и др.

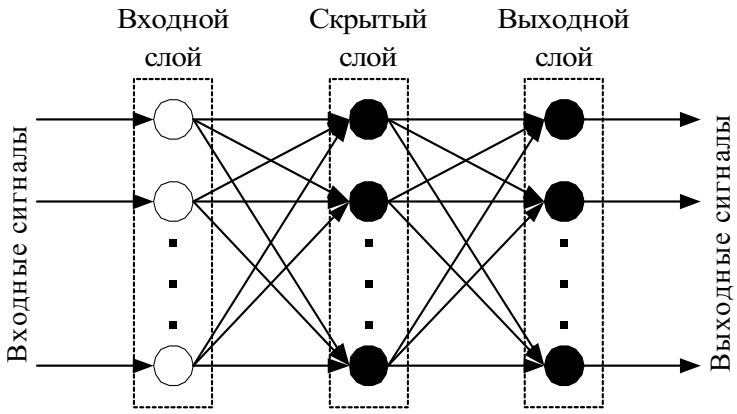


Рис. 6.5. Многослойная ИНС прямого распространения

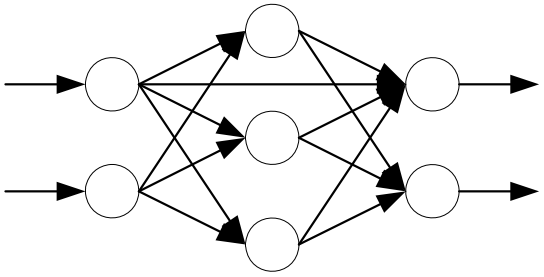


Рис. 6.6. Многослойная ИНС с перекрестной связью

2. *Аппроксимация*. При решении многих задач адаптивного управления и моделирования сложных систем появляется задача аппроксимации функции. Эта задача может быть определена как задача построения такой функциональной зависимости, которая бы проходила максимально близко к имеющимся точкам (результатам измерений).

3. *Прогнозирование*. Задача прогнозирования заключается в попытке предсказать состояние системы по ее поведению в течение некоторого промежутка времени в прошлом. Такая задача часто встречается при моделировании различных процессов и анализе временных рядов.

4. *Кластеризация*. Среди практических задач обработки данных важную роль занимает задача разбиения множества данных на отдельные кластеры таким образом, чтобы объекты внутри одного кластера обладали большим сходством, чем объекты из разных кластеров. При этом часто количество кластеров изначально неизвестно и должно быть определено в процессе обработки данных нейронной сетью.

5. *Оптимизация*. Данная задача состоит в поиске таких параметров функции, при которых эта функция принимает наибольшее (или наименьшее) значение. В ряде случаев удается свести функционирование ИНС к такому виду, что значения весов, при которых достигается локальный минимум энергии ИНС, будут соответствовать искомым значениям параметров функции.

Использование ИНС обладает следующими преимуществами:

1. Возможность решения трудноформализуемых задач, для которых трудно найти точный алгоритм решения (распознавание речи, рукописного текста). Отметим, что успешность применения ИНС существенно зависит от постановки задачи и исходных данных.
2. Массовый параллелизм в обработке информации. Данное преимущество позволяет реализовать нейросетевые алгоритмы и методы на параллельных вычислительных структурах, что особенно актуально в настоящее время в связи с распространением распределенных вычислений и массовым внедрением многоядерных центральных и графических процессоров для ПК, а также в связи с унификацией разнородных вычислений (научных, физических, графических и др.) на персональных компьютерах.
3. ИНС представляют единую концепцию для решения таких разнообразных задач, как задачи классификации, аппроксимации, моделирования, распознавания образов, принятия решений, обработки информации, кластеризации и др.
4. Возможность нестандартного решения известных задач, что расширяет и обогащает арсенал существующих средств и подходов, поскольку позволяет посмотреть на проблему и ее решение под «нестандартным» углом.

6.4. Обучение искусственной нейронной сети

Для ИНС прямого распространения без скрытых слоев с n_I входами и n_O выходами зависимость выходных сигналов ИНС от входных можно представить следующим образом (будем считать, что одномерный вектор представлен как вектор-столбец):

$$\mathbf{Y} = G(\mathbf{X}) = F(\mathbf{W}\mathbf{X}),$$

где $\mathbf{X} = \{x_i; i = 1...n_I\}$ и $\mathbf{Y} = \{y_j; j = 1...n_O\}$ – векторы входных и выходных сигналов, соответственно; $\mathbf{W} = \{w_{ij}; i = 1...n_I, j = 1...n_O\}$ – матрица весов межнейронных связей, в которой элемент w_{ij} соответствует весу связи от i-го входного нейрона к j-му выходному; $F(\cdot)$ – вектор-функция выходного сигнала слоя нейронов, вид которой зависит от выбранной активационной функции.

При добавлении скрытого слоя с n_H нейронами в структуру ИНС значение выражения для выходных сигналов ИНС изменится:

$$\mathbf{Y} = G(\mathbf{X}) = F(\mathbf{W}^{(1)}\mathbf{Y}^{(1)}) = F\left(\mathbf{W}^{(1)}F(\mathbf{W}^{(0)}\mathbf{X})\right),$$

где $\mathbf{W}^{(0)} = \{w_{ij}^{(0)}; i = 1...n_I, j = 1...n_H\}$ и $\mathbf{W}^{(1)} = \{w_{ij}^{(1)}; i = 1...n_H, j = 1...n_O\}$ – матрицы весов межнейронных связей для скрытого и выходного слоев, соответственно. Добавление дополнительных скрытых слоев соответствующим образом изменит вид функции выходного сигнала ИНС. Однако для выбранной структуры ИНС, при условии постоянного вектора входных сигналов **X** и фиксированной функции активации нейронов, значение выходного сигнала ИНС зависит только от значений весов связей.

Для решения практических задач важным является поиск такого набора значений весов межнейронных связей, при котором выходные сигналы ИНС изменяются в определенной зависимости от предъявляемого вектора входных сигналов. Процесс подстройки весов межнейронных связей называется *обучением нейронной сети***.** От того, насколько качественно будет выполнено обучение, зависит способность нейронной сети решать поставленную задачу.

Существуют два общих подхода к обучению ИНС: обучение с учителем и без учителя.

Обучение с учителем (supervised learning) подразумевает использование заранее сформированного множества обучающих примеров. Каждый пример содержит вектор входных сигналов и соответствующий вектор эталонных выходных сигналов, которые зависят от поставленной задачи. Данное множество называют *обучающей выборкой* или обучающим множеством. Обучение нейронной сети направлено на такое изменение весов связей ИНС, при котором значения выходных сигналов ИНС как можно меньше отличаются от требуемых значений выходных сигналов для данного вектора входных сигналов.

При обучении без учителя (unsupervised learning) подстройка весов связей производится либо в результате конкуренции между нейронами, либо с учетом корреляции выходных сигналов нейронов, между которыми существует связь. В случае обучения без учителя обучающая выборка не используется.

В дальнейшем будем рассматривать обучение с учителем, которое можно описать следующей последовательностью действий (практический пример представлен в разделе 6.6):

1. Подготовка обучающей выборки.
2. Выбор структуры ИНС.
3. Настройка весов связей (обучение ИНС).

В процессе обучения на вход нейронной сети предъявляются данные из обучающей выборки и, в соответствии со значениями выходных сигналов, определяющих ошибку функционирования сети, производится коррекция весов связей. В результате обучения должна быть получе-

на нейронная сеть, которая без перенастройки весов связей формирует с требуемой погрешностью выходные сигналы **Y** при подаче на вход сети любого набора входных сигналов из обучающего множества. Качество обученной нейронной сети проверяется с использованием данных, не участвовавших в процессе обучения.

Опишем каждый этап более подробно. Обучающая выборка состоит из множества пар векторов ($\mathbf{X}_i^{(T)}$, $\mathbf{Y}_i^{(T)}$), $i = 1...K$, где K – количество примеров в обучающей выборке; $\mathbf{X}_i^{(T)} = \{x_{i,j}^{(T)}; j = 1...n_I\}$ – вектор входных сигналов; $\mathbf{Y}_i^{(T)} = \{y_{i,j}^{(T)}; j = 1...n_O\}$ – вектор соответствующих $\mathbf{X}_i^{(T)}$ выходных сигналов.

Обучающая выборка формируется на основе уже известных данных по рассматриваемой проблеме. С одной стороны, чем больше и разнообразнее выборка, тем лучше будет результат обучения, но с другой – выборка может быть избыточной, что увеличивает время обучения. Отметим, что сформированная обучающая выборка может быть дополнительно обработана с использованием статистических и других методов для уменьшения мощности вектора входных сигналов путем удаления неинформативных и малоинформативных компонентов, которые не оказывают существенного влияния на результат обучения.

Выбор структуры нейронной сети оказывает влияние на характеристики функции выхода ИНС, т. к. структура ИНС определяет расположение и количество межнейронных связей и, соответственно, количество весов этих связей, которые необходимо настроить в результате обучения. Однозначной методики выбора количества скрытых слоев и нейронов в них нет, и вопрос о том, насколько успешным является тот или иной выбор, зачастую решается на основании экспериментальных результатов обучения и тестирования ИНС. Таким образом, структура сети выбирается разработчиком методом проб и ошибок, исходя из его личного опыта, а также, в ряде случаев, из характеристик обучающих данных. Отметим, что в большинстве случаев достаточно не более 2–3 скрытых слоев.

После того как определены обучающие данные и структура сети, производится настройка весов связей. Веса инициализируются случайными значениями, как правило, из диапазона [–0,01; 0,01] – [–0,1; 0,1], для того чтобы впоследствии избежать возможного насыщения функций активации нейронов и повышенной чувствительности выхода ИНС к несущественным (в пределах погрешности) изменениям сигналов.

В процессе обучения происходит коррекция (подстройка) значений весов связей. При этом пары векторов ($\mathbf{X}_i^{(T)}$, $\mathbf{Y}_i^{(T)}$) из обучающего мно-

жества могут предъявляться многократно. Один «прогон» всех наборов данных из обучающей выборки вместе с коррекцией весов составляет одну эпоху обучения. Типичная длительность обучения может составлять от десятков до нескольких десятков тысяч эпох в зависимости от поставленной задачи, структуры ИНС, качества самих данных и выбранного алгоритма подстройки весов связей.

В результате обучения может возникнуть проблема переобучения сети. Дело в том, что обучающая выборка может содержать неточности, связанные, например, с погрешностями измерения, округлением или субъективностью оценок. Длительное обучение сети может привести к тому, что обученная сеть будет в процессе работы повторять эти неточности. Данная проблема возникает, если значение ошибки обучаемой сети близко к нулю. Поэтому часто обучение останавливают, когда ошибка достигает значения 0,01–0,001.

Рассмотрим распространенный алгоритм обратного распространения ошибки и его модификацию, использующую инерционность.

8.5. Алгоритм обратного распространения ошибки

В настоящее время существует множество алгоритмов обучения. Наиболее известный из них – *алгоритм обратного распространения ошибки*. Данный алгоритм используется для минимизации отклонения реальных значений выходных сигналов нейронной сети от требуемых.

В качестве функции ошибки ИНС будем рассматривать следующую величину:

$$E(w) = \frac{1}{2}\sum_i E_i = \frac{1}{2}\sum_{i,k}(f_{i,k} - y_{i,k}^{(T)})^2, \qquad (6.1)$$

где $f_{i,k}$ – значение выходного сигнала k-го выходного нейрона сети при подаче на ее входы i-го набора обучающих данных; $y_{i,k}^{(T)}$ – требуемое значение выходного сигнала k-го выходного нейрона для i-го набора данных для обучения. Суммирование ведется по всем нейронам выходного слоя и по всем наборам данных из обучающей выборки. Обучение ИНС направлено на минимизацию функции $E(w)$.

Минимизация методом градиентного спуска обеспечивает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(q)} = \eta \frac{\partial E}{\partial w_{ij}}, \qquad (6.2)$$

где $\Delta w_{ij}^{(q)}$ – величина изменения веса связи, соединяющей i-й нейрон $(q-1)$ слоя с j-м нейроном слоя q; η – коэффициент скорости обучения,

$0 < \eta < 1$. Таким образом, вес связи изменяется пропорционально ее вкладу в значение ошибки нейрона, для которого эта связь является входной, т. к. частная производная по весу $\frac{\partial E}{\partial w_{ij}}$ показывает зависимость скорости изменения функции ошибки E от изменения этого веса.

Опустим преобразования формулы (6.2) и представим сразу конечный результат (6.3). Подробный вывод формул приведен, например, в [20] и [21]. Изменение веса связи определяется следующим образом:

$$\Delta w_{ij}^{(q)} = -\eta \delta_j x_i, \tag{6.3}$$

где δ_j – значение ошибки j-го нейрона в слое q; x_i – значение i-го входного сигнала для j-го нейрона слоя q. Данная формула применима и для настройки смещений нейронов, только вместо x_i необходимо подставить «1».

Отметим, что значение ошибки нейрона определяется в зависимости от его положения в сети.

Для нейронов выходного слоя

$$\delta_i = \left(f_{i,k}(S)\right)'\left(f_{i,k} - y_{i,k}\right), \tag{6.4}$$

где $y_{i,k}$ – требуемое, а $f_{i,k}$ – фактическое значение выходного сигнала k-го нейрона для i-го набора данных из обучающей выборки; $\left(f_{i,k}(S)\right)'$ – значение производной активационной функции k-го нейрона для i-го набора обучающих данных.

Если нейрон принадлежит одному из скрытых слоев, то

$$\delta_i^{(q)} = \left(f_i^{(q)}(S)\right)' \sum_j w_{ij} \delta_j^{(q+1)}, \tag{6.5}$$

где $\delta_i^{(q)}$ – ошибка i-го нейрона в слое q; $\delta_j^{(q+1)}$ – ошибка j-го нейрона в (q+1) слое; w_{ij} – вес связи, соединяющей эти нейроны; $\left(f_{i,k}(S)\right)'$ – значение производной активационной функции i-го нейрона слоя q. Таким образом, значение ошибки нейрона пропорционально его «влиянию» на величины ошибок нейронов следующего слоя, а также скорости изменения его выходного сигнала для k-го набора обучающих данных.

Рассмотрим ИНС с нейронами с лог-сигмоидными функциями активации

$$f(S) = \frac{1}{1 + e^{-aS}}, \tag{6.6}$$

где a – константа; S – взвешенная сумма входных сигналов нейрона, тогда

$$f'(S) = af(S)(1 - f(S)), \tag{6.7}$$

и формулы (6.4), (6.5) соответственно примут вид

$$\delta_i = af_{i,k}(1-f_{i,k})(f_{i,k}-y_{i,k}); \tag{6.8}$$

$$\delta_i^{(q)} = af_i(1-f_i)\sum_j w_{i,j}\delta_j^{(q+1)}. \tag{6.9}$$

Для реализации алгоритма обратного распространения ошибки может быть использована следующая последовательность действий:

1. Предъявление очередного набора из обучающей выборки на вход нейронной сети.
2. Вычисление выходного сигнала сети.
3. Определение величин ошибок нейронов выходного слоя по формуле (6.4) или (6.8).
4. Определение величин ошибок нейронов скрытых слоев по формулам (6.5) или (6.9).
5. Однократная коррекция весов связей.
6. Если в обучающей выборке есть неиспользованные в данной эпохе наборы данных, то переход на шаг 1.
7. Подсчет ошибки сети по формуле (6.1). Если ошибка меньше заданной, то конец обучения, иначе – начало новой эпохи обучения и переход на шаг 1.

Отметим, что алгоритм обратного распространения ошибки применим только для нейронных сетей, содержащих нейроны с дифференцируемой функцией активации. Рассмотренный алгоритм не подходит для настройки сетей, построенных на нейронах с пороговыми функциями активации. Для таких сетей применяются другие алгоритмы обучения, например дельта-правило Уидроу-Хоффа [21, 38]. Однако во многих случаях в нейронных сетях используются нейроны с сигмоидальными функциями активации (лог-сигмоидные функции и гиперболический тангенс), что дает возможность применять для настройки и обучения таких сетей градиентные алгоритмы.

В настоящее время создано немало модификаций алгоритма обратного распространения ошибки [21]. Одним из простых, но довольно эффективных является алгоритм, использующий инерционность обучения. В нем вес связи изменяется в соответствии со следующей формулой:

$$w_{ij}(t+1) = w_{ij}(t) - \eta\delta_j x_i + \alpha\Delta w_{ij}(t-1),$$

где α – коэффициент инерционности от 0 до 1. Данный алгоритм сходится в среднем в 4–5 раз быстрее стандартного, но использует больший объем памяти.

6.6. Работа нейронной сети

Для того чтобы понять возможности применения ИНС, необходимо более подробно изучить особенности функционирования отдельных нейронов и ИНС в целом.

Для начала рассмотрим один простейший нейрон с двумя входами и пороговой функцией активации (рис. 6.7) с порогом $\theta = 0$. Тогда выход нейрона y равен 1, если взвешенная сумма входных сигналов больше либо равна 0. В обратном случае выходной сигнал нейрона равен 0.

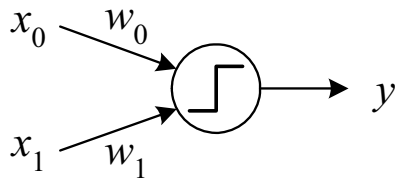


Рис. 6.7. Нейрон с двумя входами и пороговой функцией активации

Области на плоскости входных сигналов, в которых значение выхода нейрона постоянно, показаны на рис. 6.8.

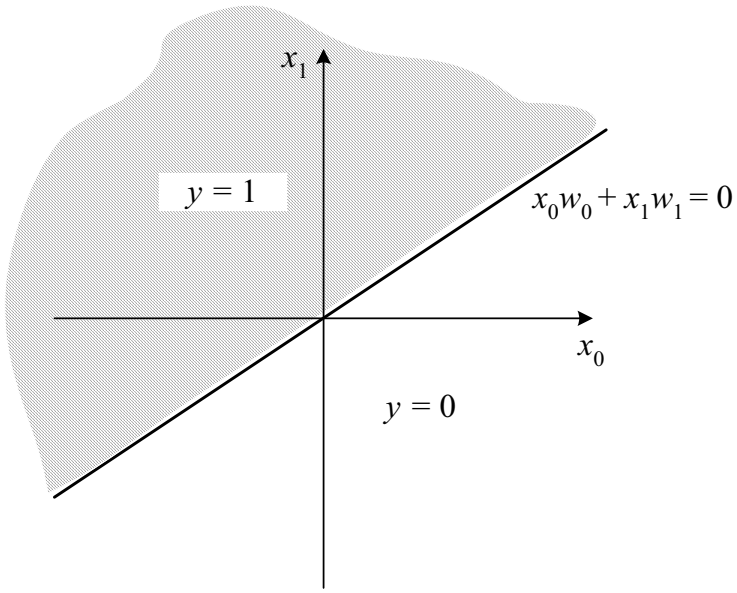


Рис. 6.8. Области с постоянным значением выходного сигнала для нейрона на рис. 6.7

Очевидно, что прямая, разделяющая эти области, имеет уравнение

$$x_0 w_0 + x_1 w_1 = 0 .$$

Изменение весов связей нейрона приведет к изменению наклона прямой относительно координатных осей, а добавление в уравнение постоянного смещения вертикально смещает рассматриваемую прямую

относительно оси Ox_1. При этом уравнение разделяющей прямой примет следующий вид:

$$x_0 w_0 + x_1 w_1 + b = 0\,.$$

Настройкой значений весов связей и смещения можно добиться требуемого положения разделяющей прямой, например, для реализации с помощью ИНС логической операции «ИЛИ» (рис. 6.9).

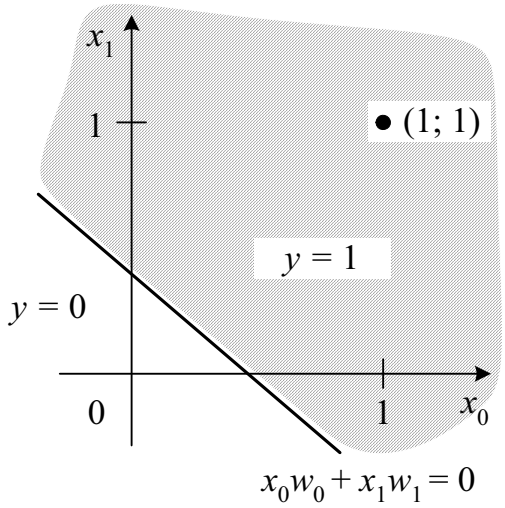


Рис. 6.9. Пример разделяющей поверхности для нейрона, реализующего логическую операцию «ИЛИ»

Естественно, нейрон может иметь больше двух входных сигналов и отличную от пороговой функцию активации. Однако в этом случае в силу многомерности пространства входных сигналов значительно усложняется только визуальное представление зависимости выходного сигнала нейрона от входного вектора. Принцип работы формального нейрона остается неизменным: существуют области пространства входных признаков, в которых выходной сигнал нейрона принимает определенное значение. При этом разделяющая прямая также становится многомерной (ее размерность равна (n – 1), где n – размерность вектора входных сигналов нейрона) и называется разделяющей гиперплоскостью.

Несколько нейронов, имеющих одинаковые входные сигналы, могут быть настроены (обучены) таким образом, чтобы реализовывать различные операции. Например, в случае использования двух нейронов с пороговой функцией активации, один из нейронов может реализовать логическую операцию «ИЛИ», а другой – инверсию «И». Разделяющие гиперплоскости таких нейронов для двумерного входного вектора показаны на рис. 6.10.

Тогда в случае, если выходные сигналы этих нейронов будут поданы на вход нейрона, обученного логической операции «И», то такая нейронная сеть из трех нейронов будет реализовывать логическую опе-

рацию «Исключающее ИЛИ», которую невозможно получить с использованием одного нейрона. Поскольку невозможно провести единственную разделяющую гиперплоскость (прямую, для случая двух входных сигналов) таким образом, чтобы отделить пару сигналов (0; 1) и (1; 0) от пары сигналов (0; 0) и (1; 1).

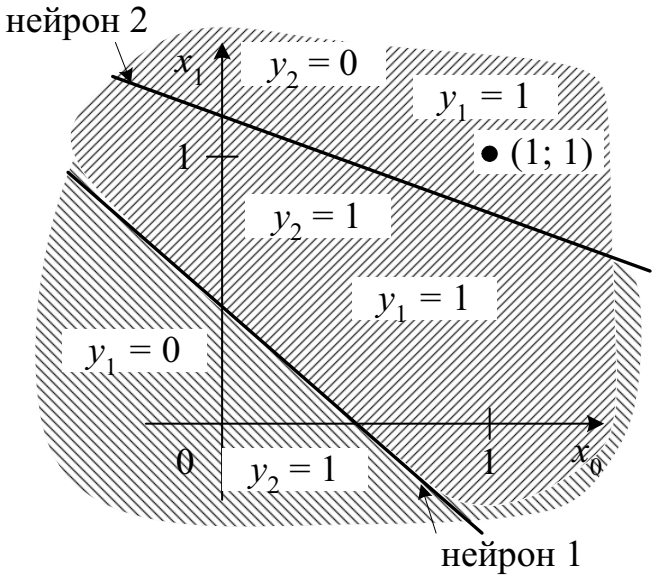


Рис. 6.10. Примеры разделяющих поверхностей для нейронов, реализующих логические операции «ИЛИ» (нейрон 1) и инверсия «И» (нейрон 2)

Таким образом, правильная (с точки зрения рассматриваемой задачи) настройка разделяющих гиперплоскостей, соответствующих нейронам, позволяет реализовать с помощью сети связанных нейронов достаточно сложные функции и зависимости, что актуально при решении практических задач. При этом каскадная (многослойная) организация структуры связей нейронов дает возможность выделять в пространстве входных сигналов области достаточно сложной формы (в том числе и невыпуклые, и многосвязные), причем, чем больше используется слоев, тем более сложной может быть форма подобласти (рис. 6.11). Так для ИНС с нейронами с пороговой функцией активации без скрытых слоев возможно лишь линейное разделение пространства входных сигналов; с одним скрытым слоем – выделение односвязных выпуклых областей; с двумя и более скрытыми слоями – выделение областей произвольной формы и связности.

Использование функции активации, отличной от пороговой, например, сигмоидной, не приводит к значительному изменению характера разделяющей гиперплоскости. В этом случае в пространстве входного сигнала появляются области, соответствующие «переходным» значениям выхода нейрона между 0 и 1.

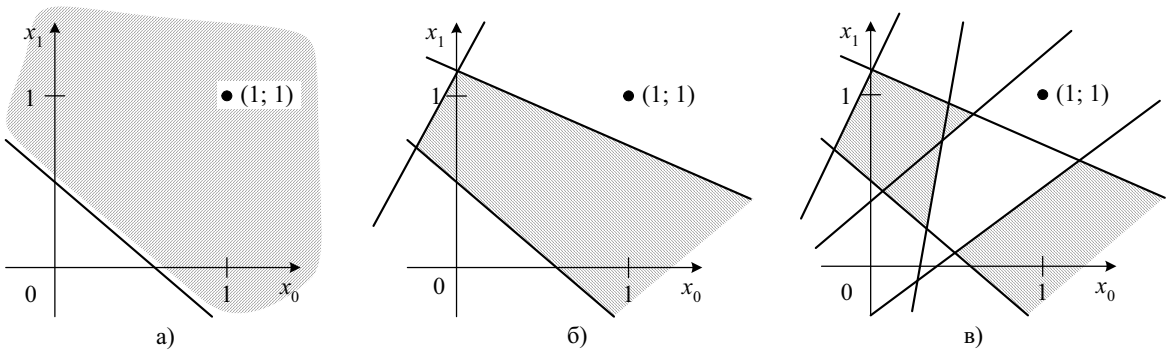


Рис. 6.11. Примеры зависимости формы выделяемых областей в пространстве входных сигналов от количества скрытых слоев в ИНС прямого распространения: а) нет скрытых слоев; б) 1 скрытый слой; в) 2 скрытых слоя

Таким образом, работа многослойной нейронной сети может быть представлена как глобальное взаимодействие распределенных нейронов, каждый из которых выполняет локальную задачу (активизируясь при определенных значениях входных сигналов). Обучение такой ИНС подразумевает решение задачи стохастической аппроксимации отображения входных сигналов ИНС в выходные часто без извлечения информации о характере и свойствах самого отображения.

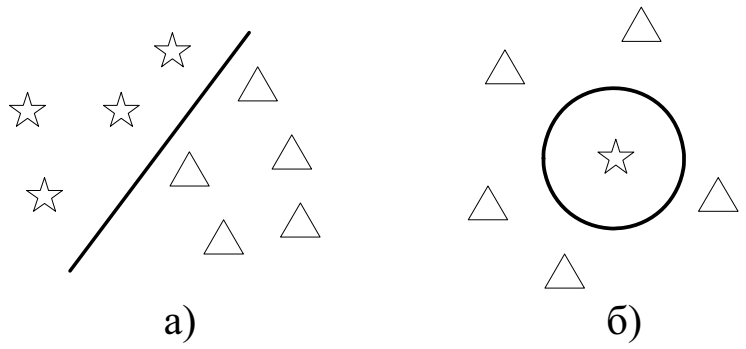


Рис. 6.12. Вид разделяющих поверхностей для нейронов с пороговой (а) и радиальной (б) функциями активации

Принципиально другой подход используется в ИНС с радиально-базисными функциями активации – РБФ-сети. Особенностью радиально-базисных функций является то, что они принимают ненулевые значения только в некоторой окрестности точки $\mathbf{c} = \{c_1, c_2 ... c_n\}$, называемой центром, где n – количество входных сигналов нейрона. Примером такой функции может являться многомерная функция Гаусса

$$y = \exp\left(-\frac{\|\mathbf{c} - \mathbf{x}\|^2}{2\sigma^2} \right),$$

где $\mathbf{x} = \{x_1, x_2 ... x_n\}$ – вектор входных сигналов нейрона; σ – ширина радиальной функции. На рис. 6.12 показан пример, иллюстрирующий отличия разделяющих поверхностей для нейронов с пороговой и радиальной функциями активации.

Структура РБФ-сети включает один скрытый слой с радиальными функциями активации нейронов и выходной слой с единственным нейроном с линейной функцией активации. Зависимость выходных сигналов ИНС от входных представляется в виде разложения по отдельным базисным функциям (соответствующих активационным функциям нейронов), веса которых определяются как веса соответствующих связей выходного нейрона. Таким образом, обучение РБФ-сетей сводится к поиску таких векторов центров и значений ширины активационных функций РБФ-нейронов, при которых достигается аппроксимация зависимости выходных сигналов ИНС от входных с требуемой точностью.

6.7. Пример работы и обучения нейронной сети

Рассмотрим работу и обучение нейронной сети на примере. Дана сеть с двумя входами, двумя скрытыми слоями, в каждом из которых по два нейрона, и одним выходом. Функции активации нейронов – логсигмоидные (6.6), $a = 1$. Скорость обучения η примем равной 0,8. Необходимо реализовать с помощью ИНС логическую операцию «ИЛИ». Таблица истинности представлена в табл. 6.2, где $x1$ и $x2$ – входные переменные, а y – значение результата.

Таблица 6.2

Таблица истинности для операции «ИЛИ»

$x1$	$x2$	y
0	0	0
0	1	1
1	0	1
1	1	1

Предположим, что на некотором этапе обучения получена сеть, соответствующая изображенной на рис. 6.6. Допустим также, что на некотором шаге веса сети равны значениям из табл. 6.3. Смещения нейронов обозначены как веса с индексом «–».

Роль обучающей выборки в рассматриваемом случае играют данные из табл. 6.2, в которой каждая строка соответствует одному набору данных. Пусть на вход нейронной сети поданы входные сигналы из 2-го обучающего набора ($x1 = 0$, $x2 = 1$). Тогда, в соответствии с таблицей истинности (табл. 6.2), выход сети должен равняться 1. Действительные

значения выходов нейронов приведены в табл. 6.4. На рис. 6.13 поясняется вычисление выходных сигналов ИНС.

Таблица 6.3

Веса связей ИНС

Номер слоя	Индекс нейрона	Индекс веса	Вес
1	0	–	0,02
1	0	0	0,12
1	0	1	0,35
1	1	–	–0,015
1	1	0	–0,5
1	1	1	0,24
2	0	–	–0,084
2	0	0	–0,33
2	0	1	0,27
2	1	–	0,037
2	1	0	–0,08
2	1	1	0,79
3	0	–	0,04
3	0	0	0,062
3	0	1	0,64

Таблица 6.4

Значения выходных сигналов нейронов

Номер слоя	Индекс нейрона	Значение выхода
1	0	0,591
1	1	0,556
2	0	0,468
2	1	0,606
3	0	0,612

Требуемое значение выхода сети «1», следовательно, ошибка сети на данном этапе равна

$$E_2 = 0{,}5\cdot(1 - 0{,}612)^2 = 0{,}075.$$

Ошибка выходного нейрона, согласно (6.8)

$$\delta_0^{(3)} = 0{,}612\cdot(1 - 0{,}612)\cdot(0{,}612 - 1) = -0{,}092.$$

Ошибки нейронов 2-го скрытого слоя по формуле (6.9)

$$\delta_0^{(2)} = 0{,}468\cdot(1 - 0{,}468)\cdot(-0{,}092)\cdot 0{,}062 = -0{,}001;$$

$$\delta_1^{(2)} = 0{,}606\cdot(1 - 0{,}606)\cdot(-0{,}092)\cdot 0{,}64 = -0{,}014.$$

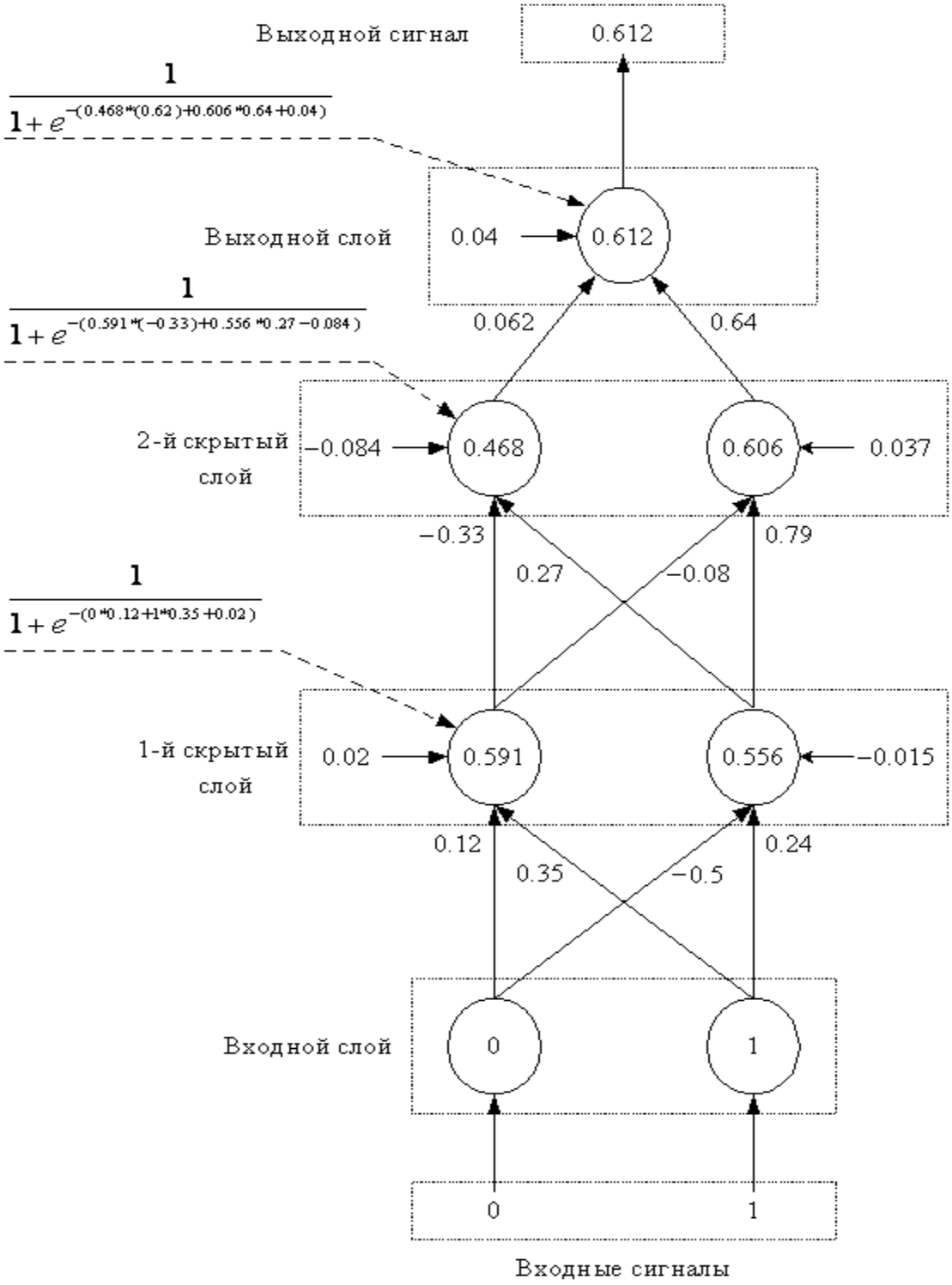


Рис. 6.13. Прямой проход для вычисления выхода ИНС

Ошибки нейронов 1-го скрытого слоя по формуле (6.9)

$$\delta_0^{(1)} = 0{,}591\cdot(1 - 0{,}591)\cdot((-0{,}001)\cdot(-0{,}33) + (-0{,}014)\cdot(-0{,}08)) = 0{,}0002;$$

$$\delta_1^{(1)} = 0{,}556\cdot(1 - 0{,}556)\cdot((-0{,}001)\cdot 0{,}27 + (-0{,}014)\cdot 0{,}79) = -0{,}003.$$

Вес первой связи выходного нейрона изменится, согласно (6.3) на следующую величину:

$$\Delta w_{00}^{(3)} = -0{,}8\ \delta_0^{(3)}\ f_0^{(2)} = -\,0{,}8\cdot(-0{,}092)\cdot 0{,}468 = 0{,}034.$$

Таким образом, новое значение веса будет равно

$$w_{00}^{(3)}(t+1) = w_{00}^{(3)}(t) + \Delta w_{00}^{(3)} = 0{,}062 + 0{,}034 = 0{,}096.$$

Аналогично производится коррекция остальных весов связей ИНС:

$$w_{10}^{(3)}(t+1) = 0{,}64 - 0{,}8\cdot(-0{,}092)\cdot 0{,}606 = 0{,}685;$$

$$w_{-0}^{(3)}(t+1) = 0{,}04 - 0{,}8\cdot(-0{,}092)\cdot 1 = 0{,}114;$$

$$w_{00}^{(2)}(t+1) = -0{,}33 - 0{,}8\cdot(-0{,}001)\cdot 0{,}591 = -0{,}329;$$

$$w_{10}^{(2)}(t+1) = 0{,}27 - 0{,}8\cdot(-0{,}001)\cdot 0{,}556 = 0{,}27;$$

$$w_{-0}^{(2)}(t+1) = -0{,}084 - 0{,}8\cdot(-0{,}001)\cdot 1 = -0{,}083;$$

$$w_{01}^{(2)}(t+1) = -0{,}08 - 0{,}8\cdot(-0{,}001)\cdot 0{,}591 = -0{,}073;$$

$$w_{11}^{(2)}(t+1) = 0{,}79 - 0{,}8\cdot(-0{,}014)\cdot 0{,}556 = 0{,}796;$$

$$w_{-1}^{(2)}(t+1) = 0{,}037 - 0{,}8\cdot(-0{,}014)\cdot 1 = 0{,}048;$$

$$w_{00}^{(1)}(t+1) = 0{,}12 - 0{,}8\cdot 0{,}0002\cdot 0 = 0{,}12;$$

$$w_{10}^{(1)}(t+1) = 0{,}35 - 0{,}8\cdot 0{,}0002\cdot 1 = 0{,}35;$$

$$w_{-0}^{(1)}(t+1) = 0{,}02 - 0{,}8\cdot 0{,}0002\cdot 1 = 0{,}02;$$

$$w_{01}^{(1)}(t+1) = -0{,}5 - 0{,}8\cdot(-0{,}003)\cdot 0 = -0{,}5;$$

$$w_{11}^{(1)}(t+1) = 0{,}24 - 0{,}8\cdot(-0{,}003)\cdot 1 = 0{,}242;$$

$$w_{-1}^{(1)}(t+1) = -0{,}015 - 0{,}8\cdot(-0{,}003)\cdot 1 = -0{,}013.$$

В случаях, когда вместо индекса начального нейрона связи стоит символ «–» производится настройка смещения нейрона, т. е. запись $w_{-1}^{(2)}(t+1)$ означает новое значение смещения у нейрона с индексом 1 во втором скрытом слое. Использованные для расчета значения ошибок выходов нейронов показаны на рис. 6.14.

Значения выходных сигналов ИНС при повторном подсчете выходов нейронов сети при использованном для коррекции весов наборе обучающих данных представлены в табл. 6.5 и на рис. 6.15.

Новое значение ошибки ИНС для 2-го набора обучающих данных равно

$$E_2 = 0{,}5\ (1 - 0{,}641)^2 = 0{,}065.$$

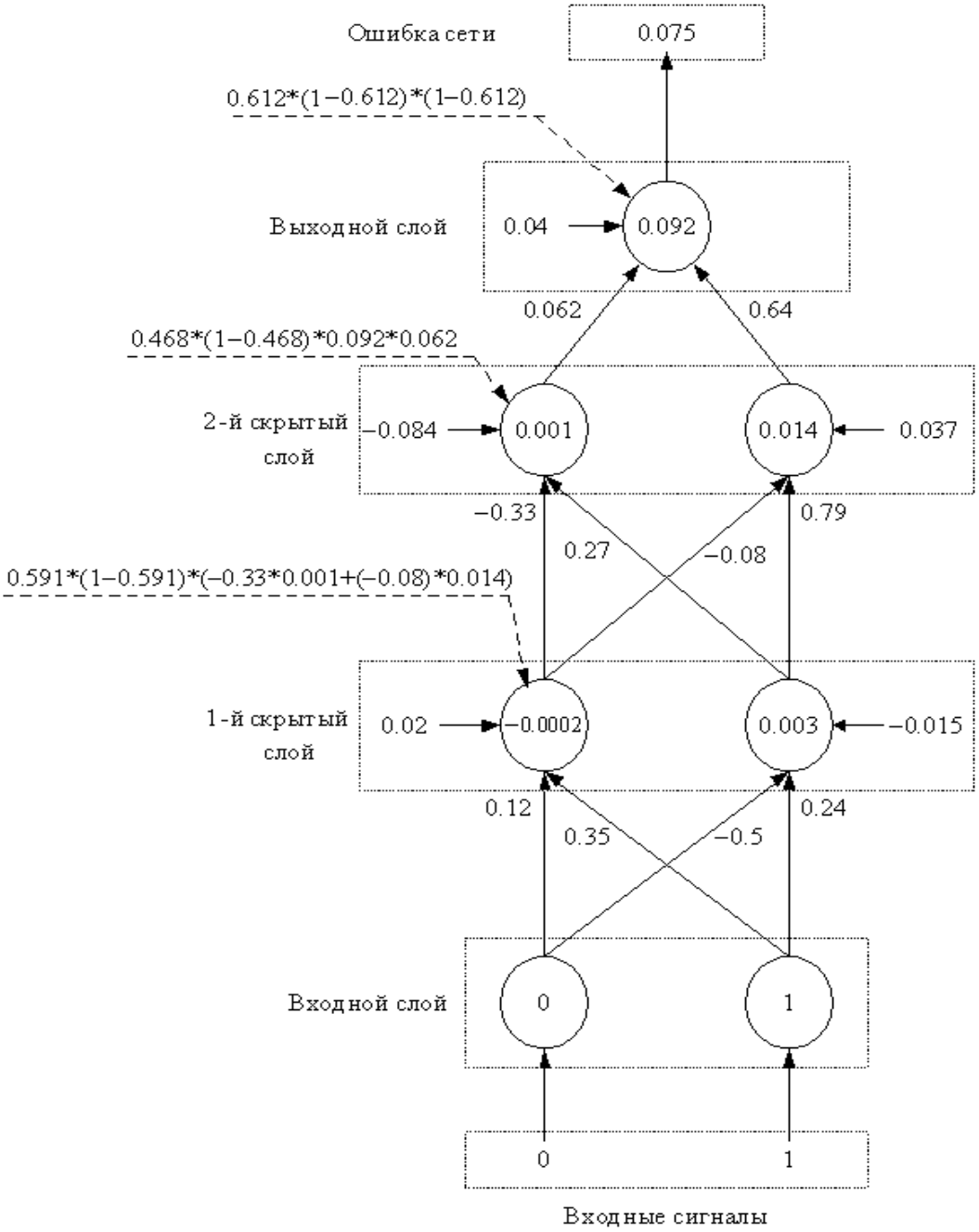


Рис. 6.14. Обратный проход для вычисления ошибок нейронов

Таблица 6.5

Значения выходных сигналов нейронов после корректировки весов связей ИНС

Номер слоя	Индекс нейрона	Значение выхода
1	0	0,591
1	1	0,557
2	0	0,468
2	1	0,610
3	0	0,640

Напомним, что значение ошибки E_2 до коррекции весов равнялось 0,075. Таким образом, в результате одного шага обучения значение ошибки уменьшилось на 0,01.

Для дальнейшего обучения ИНС на вход подается следующий вектор данных из обучающего множества и производится коррекция весов связей с учетом расхождения выходного сигнала ИНС и вектора требуемых выходных сигналов, соответствующего входному. Обучение ИНС продолжается до тех пор, пока ошибка выхода ИНС, вычисленная для всех наборов данных из обучающей выборки, по значению не будет меньше требуемой.

6.8. Программная реализация

Рассмотрим вопросы, связанные с программной реализацией многослойной нейронной сети. При написании программы необходимо знать следующие параметры сети: количество входов и выходов сети и количество скрытых слоев, а также количество и тип нейронов в них.

Поскольку нейроны входного слоя, как правило, не выполняют никаких функций, то их реализация необязательна, т. е. достаточно считать, что входные сигналы сети совпадают с входными сигналами нейронов первого скрытого слоя либо, если скрытых слоев нет, со входными сигналами нейронов выходного слоя. Кроме этого, для сети прямого распространения без обратных связей верно следующее:

- все нейроны одного слоя имеют одинаковые входные сигналы;
- входные сигналы слоя $(q + 1)$ являются выходными сигналами слоя q.

Таким образом, каждый нейрон достаточно описать следующими параметрами:

- веса входящих связей (от нейронов предыдущего слоя к данному);
- смещение;
- коэффициент a в активационной функции;
- указатель на массив входных сигналов;
- указатель/ссылка на элемент массива выходных сигналов слоя, которому принадлежит данный нейрон.

Нейронный слой можно считать нейронной сетью без скрытых слоев, поэтому важнейшими характеристиками для него будут:

- число входов – равно числу нейронов в предыдущем слое;
- число выходов – равно числу нейронов в данном слое.

В многослойной сети два смежных слоя (q) и $(q + 1)$ имеют один общий массив, который будет содержать выходные сигналы слоя (q) и входные сигналы слоя $(q + 1)$. Поэтому целесообразно реализовать этот массив только один раз: либо как массив выходных сигналов слоя (q) (рис. 6.16), либо как массив входных сигналов слоя $(q + 1)$ (рис. 6.17).

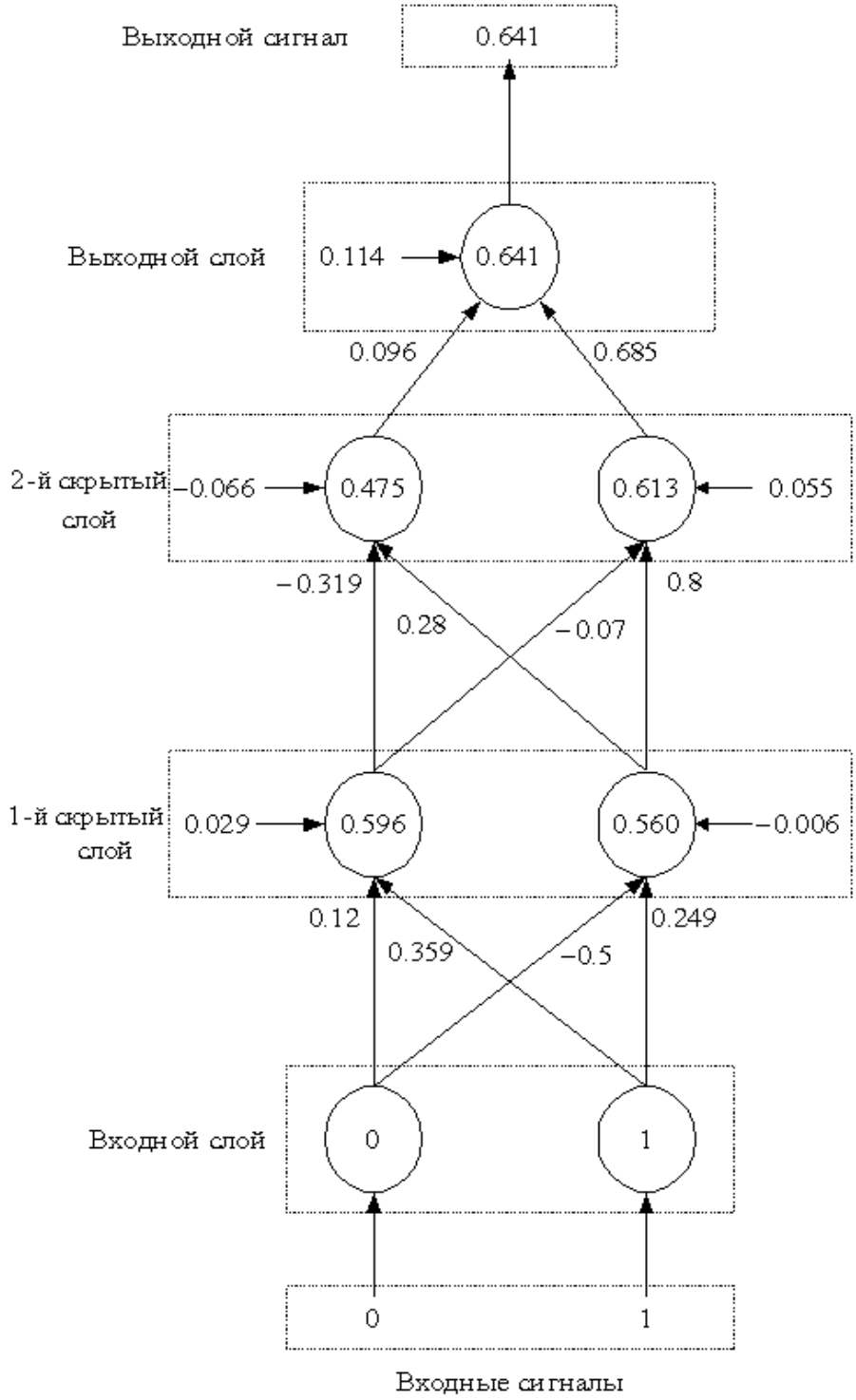


Рис. 6.15. Прямой проход после коррекции весов

В случае программной реализации ИНС с использованием объектно-ориентированного подхода можно выделить следующие объекты: нейрон, нейронный слой и нейронную сеть.

При этом класс каждого следующего объекта можно считать производным от класса предыдущего.

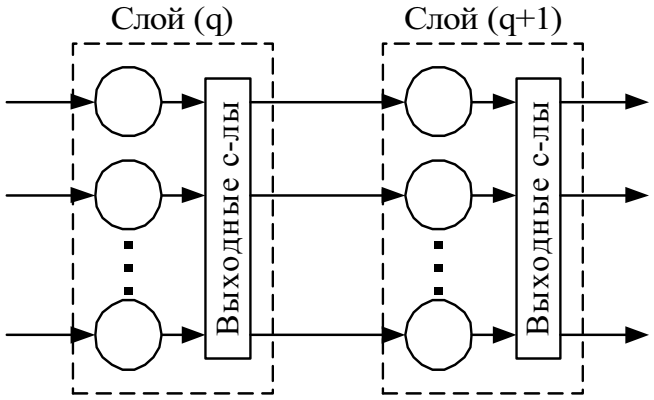


Рис. 6.16. Пример реализации сигналов ИНС в виде массива выходных сигналов слоя (q)

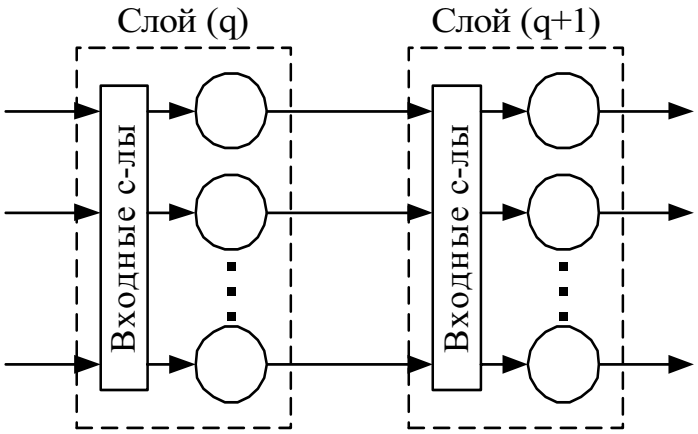


Рис. 6.17. Пример реализации сигналов ИНС в виде массива входных сигналов слоя (q + 1)

При структурном подходе нейронную сеть можно, например, представить как трехмерный массив, первый индекс которого обозначает слой, второй – нейрон в этом слое, а третий – вес соответствующей связи этого нейрона. Смещения нейронов могут быть заданы в этом же массиве либо отдельно.

Например, рассмотренная выше сеть с двумя входами, одним выходом и двумя скрытыми слоями по два нейрона в каждом может быть представлена в виде массива следующим образом (на языке программирования С):

```
double ***net;              // массив весов связей сети
net = new double**[3];      // создание указателей на слои
net[0] = new double*[2];    // создание указателей
                            // на нейроны 1-го слоя
net[1] = new double*[2];    // создание указателей
                            // на нейроны 2-го слоя
net[2] = new double*[1];    // создание указателей
                            // на нейроны 3-го слоя
```

```
net[0][0] = new double[2]; // создание массива весов
                           // для 1-го нейрона 1-го слоя
net[0][1] = new double[2]; // создание массива весов
                           // для 2-го нейрона 1-го слоя
net[1][0] = new double[2]; // создание массива весов
                           // для 1-го нейрона 2-го слоя
net[1][0] = new double[2]; // создание массива весов
                           // для 2-го нейрона 2-го слоя
net[2][0] = new double[2]; // создание массива весов
                           // для 1-го нейрона 3-го слоя
```

Чтение/запись весов осуществляется обращением к массиву следующим образом:

```
double temp = net[0][1][1];  // чтение значения веса 2-го
                             // сигнала 2-го нейрона 1-го
                             // слоя
net[1][0][1] = 0.12; // запись (изменение) значения веса
                     // 2-го сигнала 1-го нейрона 2-го
                     // слоя
```

Значения смещений можно задавать аналогичным образом через двумерный массив, в котором первый индекс обозначает слой, а второй – нейрон, к которому относится это смещение.

6.9. Задания для лабораторных работ

1. С помощью нейронной сети необходимо перекодировать прописные буквы в строчные (маленькие – в большие). На вход сети подается код «маленькой» буквы, с выхода «снимается» код соответствующей «большой» буквы.
2. Перевод нот из одной тональности в другую называется транспонированием. С помощью нейронной сети транспонируйте ноты на один тон выше. На вход сети подается код ноты, с выхода «снимается» код ноты на тон выше, октаву учитывать не нужно.
3. Реализуйте с помощью нейронной сети преобразование градусов в радианы.
4. Реализуйте с помощью нейронной сети конвертер валют из долларов в евро.
5. Имеется сеть с двумя входами, двумя выходами и некоторым количеством скрытых нейронов. Необходимо настроить сеть таким образом, чтобы сигналы со входа менялись на выходе сети местами. Если на вход поступили числа 0,75 и 0,34, то на выходе должны быть числа 0,34 и 0,75.
6. Научите нейронную сеть осуществлять операцию сложения двух чисел.

7. С помощью нейронной сети реализуйте определение знака зодиака по числу и месяцу. Знак зодиака определяется по величине сигнала выходного нейрона сети.
8. Дан набор точек (табл. 6.6). Аппроксимируйте данную зависимость полиномом второй степени с помощью нейронной сети. В отчете необходимо представить график с изображением исходных точек и кривой, полученной с помощью нейронной сети.
9. Реализуйте с помощью нейронной сети операцию умножения трех чисел из диапазона [0, 1].
10. На основании данных из табл. 6.7 продолжите числовой ряд с помощью механизма предсказания на основе нейронной сети. Ряд может содержать отрицательные числа.

Таблица 6.6

Точки для аппроксимации

X	Y
0	6,45
1	4,06
2	2,53
3	2,05
4	2,48
5	3,97
6	6,57
7	9,94
8	14,45

Таблица 6.7

Числовой ряд

a0	a1	A2	a3	a4	A5	a6	a7	a8	a9	a10
0,707	0,866	0,966	1	0,966	0,866	0,707	0,5	0,259	0	–0,259

Правильный ответ: –0,5, –0,707… (синусоида).

11. Создайте нейронную сеть, которая правильно классифицирует объекты, пользуясь данными из табл. 6.8.
Ответ: Если «Параметр 1» = 1, то 1-й класс, если «Параметр 2» = = «Параметр 3», то 2-й класс, в противном случае – 3-й класс.
12. Имеется физическая система с переменными объемом и внутренним давлением. Необходимо создать нейросетевой регулятор, поддерживающий постоянную температуру внутри этой системы. На вход регулятора подается изменение давления и объема, на выходе – изменение температуры, компенсирующее действие изменяющихся параметров.

Начальные условия: $V_0 = 15$ *дм*^{3}, $P_0 = 100$ *Па*, $T_0 = 280$ *К*. Расчет требуемого изменения температуры производится по формуле:

$$\Delta T = \frac{(P_o \Delta V + V_o \Delta P + \Delta V \Delta P) T_o}{P_o V_o},$$

где ΔV – изменение объема, а ΔP – изменение давления.

Таблица 6.8

Описание объектов

Объект	Параметр 1	Параметр 2	Параметр 3	Класс
1	1	1	0	1
2	1	0	1	1
3	0	1	1	2
4	0	1	0	3
5	0	1	1	2
6	0	0	1	3
7	0	1	0	3
8	1	1	1	1
9	0	0	0	2

13. Реализуйте с помощью нейронной сети сжатие бинарных изображений размером 16×16 пикселей с коэффициентом сжатия 2.
14. Научите нейронную сеть распознавать цифры от 0 до 9, заданные в матричном виде 5×7 (рис. 6.18).

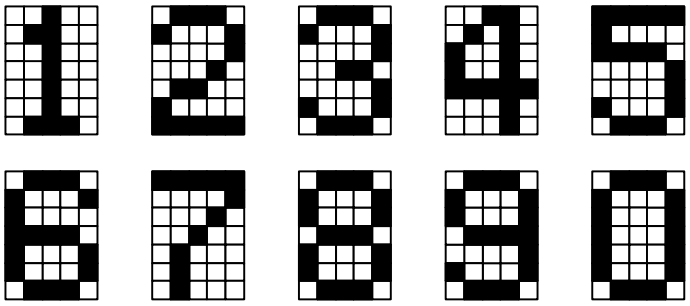


Рис. 6.18. Цифры для распознавания

СПИСОК СОКРАЩЕНИЙ

ЭС – экспертные системы
ИИ – искусственный интеллект
БЗ – база знаний
ППФ – правильно построенная формула
БД – база данных
ИЗ – источник знаний
ЕЯ – естественный язык
МА – морфологический анализ
СИА – синтаксический анализ
МИ – морфологическая информация
СЕА – семантический анализ
КМИ – комплекс морфологической информации
СО – словарь основ
СГФ – словарь готовых словоформ
ГО – гипотетическая основа
ССИО – система синтаксических отношений
ОСС – обобщенная синтаксическая структура
ИГ – именная группа
М-язык – машинный язык
ЭВ – эволюционные вычисления
ЭА – эволюционные алгоритмы
ГА – генетический алгоритм
ИНС – искусственные нейронные сети
РБФ – радиально-базисные функции

ПРИЛОЖЕНИЕ 1

Контрольные вопросы и задания

1. В чем заключается предмет исследования нейрокибернетики?
2. Охарактеризуйте принцип, положенный в основу кибернетики «черного ящика».
3. Опишите процесс мышления, протекающий в человеческом сознании.
4. Охарактеризуйте понятие чанков.
5. Укажите признаки, отличающие знания от данных.
6. Дайте определение формализованных и неформализованных знаний.
7. Укажите основные особенности и свойства экспертных систем.
8. Охарактеризуйте виды интеллектуальных систем.
9. Приведите классификацию экспертных систем по решаемой задаче.
10. Для чего предназначена экспертная система PROSPECTOR?
11. Охарактеризуйте экспертную систему MYCIN.
12. Опишите коммуникативные методы извлечения знаний.
13. Охарактеризуйте текстологические методы извлечения знаний.
14. Как осуществляется наблюдение при пассивном извлечении знаний из эксперта?
15. Опишите протокол «мыслей вслух», применяющийся при извлечении знаний.
16. Каким образом необходимо конспектировать содержание лекции при передаче знаний?
17. Охарактеризуйте анкетирование как один из активных индивидуальных методов извлечения знаний.
18. Каким образом необходимо построить интервью с экспертом?
19. Охарактеризуйте свободный диалог как метод извлечения знаний.
20. Опишите метод круглого стола при извлечении знаний.
21. В чем заключается идея «мозгового штурма» при извлечении знаний?
22. Охарактеризуйте экспертные игры как метод извлечения знаний.
23. Что понимается под деловой игрой?
24. Охарактеризуйте понятие диагностической игры.
25. Опишите классификацию компьютерных игр.
26. Охарактеризуйте понятия микроконтекста и макроконтекста.
27. В чем заключаются основные моменты понимания текста?

28. Охарактеризуйте понятия смысловой группы, смысловой вехи и ключевого слова в процедуре разбивки текста на части.
29. Какие вы знаете модели представления знаний?
30. Что представляет собой логическая модель представления знаний?
31. Из чего состоит продукционная система?
32. Опишите понятия прямых и обратных выводов, основанных на продукционных правилах.
33. Охарактеризуйте модель представления знаний в виде фреймов.
34. Каким образом осуществляется представление знаний в семантической сети?
35. Дайте определение корневого графа.
36. Опишите модель доски объявлений.
37. Дайте определение модели представления знаний в виде сценария.
38. Охарактеризуйте понятие каузального сценария.
39. Опишите архитектуру экспертных систем.
40. Как формируется база знаний?
41. Определите задачи машины вывода.
42. Каковы цели использования объяснений в экспертных системах?
43. Сформулируйте требования, которые должны выполняться при разработке экспертных систем.
44. Опишите концепцию «быстрого прототипа», применяемую при разработке экспертной системы.
45. Охарактеризуйте этапы технологии создания экспертных систем.
46. Опишите механизм вывода экспертной системы.
47. Каковы функции управляющего компонента экспертной системы?
48. Опишите схему взаимодействия пользователя с экспертной системой.
49. В чем заключаются задачи подсистемы анализа и синтеза сообщений?
50. Дайте определение декларативного и процедурного методов морфологического анализа входных сообщений.
51. Представьте общую структуру алгоритма синтаксического анализа входных сообщений.
52. Охарактеризуйте этапы семантического анализа входных сообщений.
53. Каким образом осуществляется синтез выходных сообщений?
54. Опишите общую структуру диалога.
55. Почему потребовалось вводить понятие нечеткой логики?
56. Дайте определение лингвистической переменной.
57. Каким образом коэффициент уверенности выражается через меры доверия и недоверия?
58. Приведите соотношение между мерами доверия, полученными при независимом учете первого и второго свидетельства, и объединенной мерой доверия, полученной при учете двух свидетельств.

59. Дайте определение отношения правдоподобия конкурирующих гипотез.
60. Охарактеризуйте понятие функции принадлежности.
61. Опишите операции над нечеткими множествами.
62. Охарактеризуйте нечеткие правила вывода в ЭС.
63. Приведите способы суперпозиции функций принадлежности нечетких множеств.
64. Опишите понятие дефазификации нечеткого множества.
65. Охарактеризуйте задачи, для решения которых применяется генетический алгоритм.
66. В чем заключаются отличия генетических алгоритмов от традиционных методов?
67. Приведите блок-схему работы генетического алгоритма.
68. Охарактеризуйте понятия целочисленного и вещественного кодирования.
69. Каким образом осуществляется оценивание особей в популяции?
70. Опишите способы селекции.
71. Охарактеризуйте принцип работы одноточечного, двухточечного и однородного операторов кроссинговера для целочисленного кодирования.
72. Опишите принцип работы двухточечного, арифметического и *BLX-*α операторов кроссинговера для вещественного кодирования.
73. Охарактеризуйте понятие разрушающей способности кроссинговера.
74. Опишите процесс формирования нового поколения.
75. Охарактеризуйте принципы работы оператора мутации для целочисленного и вещественного кодирования.
76. Каким образом осуществляется настройка параметров генетического алгоритма?
77. Охарактеризуйте канонический генетический алгоритм.
78. Опишите варианты реализации компонентов генетического алгоритма.
79. Охарактеризуйте биологические нейронные сети.
80. Приведите блок-схему функционирования формального нейрона.
81. Охарактеризуйте функции активации нейрона.
82. Опишите наиболее распространенные топологии искусственных нейронных сетей.
83. Какими преимуществами обладают искусственные нейронные сети по сравнению с другими методами?
84. Охарактеризуйте процесс обучения искусственных нейронных сетей прямого распространения.
85. Опишите алгоритм обратного распространения ошибки.
86. Приведите вид разделяющей поверхности для нейрона, реализующего логическую операцию «ИЛИ».

87. Опишите особенности радиально-базисных функций активации.
88. Приведите вид разделяющих поверхностей для нейронов с пороговой и радиальной функциями активации.
89. Охарактеризуйте особенности программной реализации многослойной искусственной нейронной сети с использованием объектно-ориентированного подхода.
90. Охарактеризуйте особенности программной реализации многослойной искусственной нейронной сети с использованием структурного подхода.

ПРИЛОЖЕНИЕ 2

Темы рефератов и индивидуальных заданий

Темы рефератов:

1. Эволюционное программирование (evolutionary programming).
2. Генетическое программирование (genetic programming).
3. Эволюционные стратегии (evolution strategies).
4. Меметичные алгоритмы (memetic algorithms).
5. Алгоритм дифференциальной эволюции (differential evolution).
6. Алгоритмы оценки распределений (estimation of distribution algorithms).
7. Алгоритмы ройной оптимизации (particles swarm optimization).
8. Системы обучающихся классификаторов (learning classifiers systems).
9. Адаптация параметров в эволюционных алгоритмах.
10. Интеллектуальная обработка изображений.
11. Методы распознавания образов.
12. Обработка изображений на основе применения клеточных автоматов.
13. Применение теории графов для обработки изображений и распознавания образов.
14. Методы масштабирования изображений.
15. Методы распознавания лиц.
16. Методы построения трехмерных изображений объектов по их двумерным изображениям.
17. Методы сегментации изображений.

***Темы индивидуальных заданий*:**

1. Реализация алгоритма дифференциальной эволюции (differential evolution).
2. Реализация алгоритма оценки распределений (estimation of distribution algorithms).
3. Реализация алгоритма ройной оптимизации (particles swarm optimization).

4. Реализация алгоритма обработки изображений на основе применения клеточных автоматов.
5. Разработка программы улучшения качества изображений, полученных в условиях плохой видимости, на основе применения средств пакета MATLAB.
6. Разработка программы улучшения качества изображений, содержащих шумовую компоненту, на основе применения средств пакета MATLAB.
7. Разработка программы сегментации изображений на основе применения средств пакета MATLAB.

ПРИЛОЖЕНИЕ 3

Ресурсы в сети Интернет

1. http://raai.org/ – Российская ассоциация искусственного интеллекта.
2. http://www.niisi.ru/iont/ni – Российская ассоциация нейроинформатики.
3. http://ransmv.narod.ru/ – Российская ассоциация нечетких систем и мягких вычислений.
4. http://lis.epfl.ch/ – Laboratory of Intelligent Systems.
5. http://nn.cs.utexas.edu/ – Neural networks research group (University of Texas).
6. http://datadiver.nw.ru/ – Data mining и технология «Deep Data Diver».
7. http://genetic-programming.org/ – Генетическое программирование.
8. http://www.gotai.net/ – Искусственный интеллект.
9. http://www.sigevolution.org – Sigevolution – электронный журнал-дайджест по эволюционным вычислениям.
10. http://www.makhfi.com/KCM_intro.htm – Введение в моделирование знаний.
11. http://groups.csail.mit.edu/medg/ftp/psz/k-rep.html – Представление знаний.
12. http://www.blackwellpublishing.com/journal.asp?ref=0266-4720&site=1 – Экспертные системы. Expert Systems. The Journal of Knowledge Engineering.

СПИСОК ЛИТЕРАТУРЫ

1. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2000. – 382 с.
2. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2000. – 352 с.
3. Змитрович А.И. Интеллектуальные информационные системы. – Минск: Тетра Системс, 1997. – 367 с.
4. Таунсенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ / пер. с англ. – М.: Финансы и статистика, 1990. – 320 с.
5. Искусственный интеллект: Кн. 1. Системы общения и экспертные системы. Справочник / под ред. Э.В. Попова. – М.: Радио и связь, 1990. – 464 с.
6. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирования / пер. с англ. – М.: Издательский дом «Вильямс», 2007. – 1152 с.
7. Gruber T.R. A translation approach to portable ontology specification // Knowledge Acquisition. – 1993. – № 5. – P. 199–220.
8. Дюк В., Самойленко А. Data Mining. – СПб.: Питер, 2001. – 368 с.
9. Люгер Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. – М.: Издательский дом «Вильямс», 2003. – 864 с.
10. Марселлус Д. Программирование экспертных систем на Турбо Прологе / пер. с англ. – М.: Финансы и статистика, 1994. – 256 с.
11. Осуга С. Обработка знаний / пер. с японск. – М.: Мир, 1989. – 293 с.
12. Попов Э.В. Экспертные системы. – М.: Наука, 1987. – 288 с.
13. Спицын В.Г. Базы знаний и экспертные системы. – Томск: Изд-во ТПУ, 2001. – 88 с.
14. Экспертные системы. Принцип работы и примеры / под ред. Р. Форсайда; пер.с англ. – М.: Радио и связь, 1987. – 221 с.
15. Джексон П. Введение в экспертные системы / пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 624 с.
16. Гаврилов А.В. Гибридные интеллектуальные системы. – Новосибирск: Изд-во НГТУ, 2003. – 164 с.
17. Цой Ю.Р., Спицын В.Г. Применение искусственных нейронных сетей для решения задач классификации и аппроксимации функций. – Томск: Изд-во ТПУ, 2003. – 21 с.
18. McCulloh W.S., Pitts W.H. A logical calculus of ideas immanent in nervous activity // Bull. Math. Biophysics. – 1943. – Vol. 5. – P. 115–119.

19. Куффлер С.В., Николс Дж.Г. От нейрона к мозгу. – М.: Мир, 1978. – 439 с.
20. Каллан Р. Основные концепции нейронных сетей: пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 288 с.
21. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002. – 344 с.
22. Спицын В.Г., Цой Ю.Р. Применение генетического алгоритма для аппроксимации функций. – Томск: Изд-во ТПУ, 2002. – 23 с.
23. Редько В.Г. Эволюционная кибернетика. – М.: Наука, 2003. – 156 с.
24. Бурцев М.С. Эволюция кооперации в многоагентной системе // Научная сессия МИФИ–2005. VII Всероссийская научно-практическая конференция «Нейроинформатика–2005»: Сборник научных трудов. В 2-х ч. Ч. 1. – М.: МИФИ, 2005. – С. 217–224.
25. Beyer H.-G., Schwefel H.-P., Wegener I. How to analyse Evolutionary Algorithms. Technical Report. No.CI-139/02. – University of Dortmund, Germany, 2002.
26. Holland J.H. Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.
27. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: ФИЗМАТЛИТ, 2003. – 432 с.
28. Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. – М.: Мир, 1969. – 230 с.
29. Rechenberg I. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Werkstatt Bionik und Evolutionstechnik, Stuttgart: Frommann-Holzboog, 1973.
30. Schwefel H.-P. Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie // Interdisciplinary Systems Research. – 1977. – Vol. 26.
31. Koza J. Genetic programming: a paradigm for genetically breeding computer population of computer programs to solve problems. MIT Press, Cambridge, MA, 1992.
32. Whitley D.L. Genetic Algorithms and Evolutionary Computing. Van Nostrand's Scientific Encyclopedia 2002.
33. Heitkotter J., Beasly D. The Hitch-Hiker’s Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ). [Электронный ресурс]. Режим доступа: ftp://rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic/.
34. De Jong K. An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation. – University of Michigan, Ann Arbor. – University Microfilms No. 76–9381. – 1975.
35. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software, Massachusetts: Addison-Wesley, 1995.
36. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы / под ред. В.М. Курейчика. – 2-е изд., испр. и доп. – М.: Физматлит, 2006. – 320 с.
37. Turing A.M. Computing machinery and intelligence // Mind, 1950. Vol. 236. № 59.
38. Хайкин С. Нейронные сети: полный курс, 2-е издание. – М.: Издательский дом «Вильямс», 2006.

39. Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей // Сибирский журнал вычислительной математики.– 1998. – Т. 1. № 1. – С. 12–24.
40. Цой Ю.Р., Спицын В.Г. Эволюционный подход к настройке и обучению искусственных нейронных сетей // Нейроинформатика. – 2006. – Т. 1. – № 1 – С. 34–61. [Электронный ресурс]. Режим доступа: http://www.ni.iont.ru/Journal/.
41. Цой Ю.Р., Спицын В.Г. Нейроэволюционный подход // Нейрокомпьютеры: разработка и применение. – 2005. – № 6. – С. 15–25.
42. Tsoy Yu.R., Spitsyn V.G. Using genetic algorithm with adaptive mutation mechanism for neural networks design and training // Optical memory and neural networks, (Information Optics). – 2004. – Т. 13. – № 4. – P. 225–232.
43. Цой Ю.Р., Спицын В.Г., Чернявский А.В. Нейроэволюционное улучшение качества изображений // Сборник научных трудов VIII Всероссийской научно-технической конференции «НЕЙРОИНФОРМАТИКА–2006». – М.: Изд-во МИФИ, 2006. – Т. 1. – С. 181–189.
44. Цой Ю.Р., Спицын В.Г. Исследование генетического алгоритма с динамически изменяемым размером популяции // Труды Международной научно-технической конференции «Интеллектуальные системы (IEEE AIS'05)». Научное издание. – М.: Изд-во физико-математической литературы, 2005. – С. 241–246.
45. Цой Ю.Р., Спицын В.Г. Применение генетического алгоритма для решения задачи адаптивного нейроуправления // Сборник научных трудов VII Всероссийской научно-технической конференции «НЕЙРОИНФОРМАТИКА–2005». – М.: Изд-во МИФИ, 2005. – Т. 1. – С. 35–43.
46. Цой Ю.Р., Спицын В.Г. К выбору размера популяции // Труды Международной научно-технической конференции «Интеллектуальные системы (IEEE AIS'04)». Научное издание. – М.: Изд-во физико-математической литературы. 2004. – Т. 1. – С. 90–96.

ОГЛАВЛЕНИЕ

Учебное издание

СПИЦЫН Владимир Григорьевич
ЦОЙ Юрий Робертович

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ

Учебное пособие

Редактор	*Т.Л. Владимирова*
Верстка	*К.С. Чечельницкая*
Дизайн обложки	*О.Ю. Аршинова* *О.А. Дмитриев*

Подписано к печати 26.12.08. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. 8,84. Уч.-изд. л. 8,0.
Заказ 851. Тираж 200 экз.

Томский политехнический университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2000

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.