МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

В. С. РОСТОВЦЕВ

Принципы построения экспертных систем

Учебное пособие

Киров
2018

004.891(07)
Р785

Рекомендовано к изданию кафедрой электронных вычислительных машин факультета автоматики и вычислительной техники Институт математики и информационных систем Института математики и информационных систем ВятГУ.

Допущено редакционно-издательской комиссией методического совета ВятГУ в качестве учебного пособия для студентов направления 09.03.01 «Информатика и вычислительная техника» по дисциплинам «Теория принятия решений, Системы обработки знаний» всех профилей подготовки

Рецензенты:

кандидат технических наук, доцент, заведующий кафедрой автоматики и телемеханики ВятГУ В. И. Семёновых

кандидат технических наук, доцент, заместитель директора ООО «Литек» Н. А. Чарушин

Ростовцев В.С. Принципы построения экспертных систем: учебное пособие.– 3-е изд., перераб. и доп. - Киров: Изд-во ВятГУ, 2018.-190 с.

Подписано к использованию ___. ___. 2018. Заказ № _____

Редакционно-издательский отдел ВятГУ

Лаборатория множительной техники

610601, г. Киров, ул. Московская, 36

Вятский государственный университет, 2018г.

Оглавление

ПРЕДИСЛОВИЕ 5

1. ОСНОВНЫЕ ПОНЯТИЯ И НАЗНАЧЕНИЕ ЭКСПЕРТНЫХ СИСТЕМ 7
1.1. Понятие экспертной системы 7
1.2. Компоненты типовой статической экспертной системы 9
1.3. История развития экспертных систем 12
1.4. Классификация экспертных систем 16
1.5. Классификация инструментальных средств создания экспертных систем 18

2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ 31
2.1. Данные и знания 31
2.2. Классификация знаний 34
2.3. Логическая модель представления знаний 36
2.4. Продукционная модель представления знаний 40
2.5. Фреймовая модель представления знаний 41
2.6. Семантическая модель представления знаний 47
2.7. Нейронная модель представления знаний 49
2.8. Представление знаний в виде нечетких правил 50

3. РАЗРАБОТКА ИНСТРУМЕНТАЛЬНОЙ ПРОДУКЦИОННОЙ ЭКСПЕРТНОЙ СИСТЕМЫ 60
3.1. Концепция построения инструментальной экспертной системы 60
3.2. Основы теории приближенных рассуждений 67
3.3. Характеристика инструментальной экспертной системы 81

4. НЕЙРОСЕТЕВЫЕ ЭКСПЕРТНЫЕ СИСТЕМЫ 88
4.1. Представление знаний в нейросетевой экспертной системе 88
4.2. Способность к обобщению полученных знаний 91
4.3. Нормализация входной и выходной информации 97
4.4. Обоснование создания нейросетевой экспертной системы 100
4.5. Программы моделирования искусственных нейронных сетей 103
4.6. Примеры реализации нейронных экспертных систем 107
4.7. Пример учебной нейросетевой экспертной системы 116

5. НЕЧЕТКИЕ ЭКСПЕРТНЫЕ СИСТЕМЫ 118
5.1. Краткий обзор нечетких экспертных систем 118
5.2. Реализация обучающей нечеткой экспертной системы 123
5.3. Подбор параметров нечеткой экспертной системы 126

6. ГИБРИДНЫЕ ЭКСПЕРТНЫЕ СИСТЕМЫ 132
6.1. Представление знаний в мягкой экспертной системе 132
6.2. Пример гибридной экспертной системы 136
6.3. Основные понятия теории генетических алгоритмов 141

7.Онтологии 153
7.1. Общие сведения об онтологии 153
7.2. Языки описания онтологий 161
7.3. Основные правила разработки онтологий 164
7.4. Обзор существующих онтологий 165
8. Экспертные системы на базе прецедентов 171
8.1. Рассуждения по прецедентам 171
8.2 Достоинства и недостатки использования прецедентов 174
8.3. Примеры реализации 176
8.4. Принципы создания экспертной системы на базе прецедентов 181
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 187
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 188

ПРЕДИСЛОВИЕ

Учебное пособие создано в результате работы автора над лекциями, лабораторными и самостоятельными работами для студентов направления 09.03.01 Информатика и вычислительная техника в ходе преподавания дисциплин «Теория принятия решений», «Системы обработки знаний» в Вятском государственном университете.

Цель пособия – изложение в минимальном объеме достаточного для понимания материала по принципам построения различных видов экспертных систем.

Настоящее пособие охватывает теоретический материал от основных понятий в области инженерии знаний до способов представления знаний и принципов построения продукционных, нейросетевых, нечетких и гибридных экспертных систем. Каждая глава заканчивается контрольными вопросами.

Первая глава посвящена основным понятиям в области экспертных систем, истории развития, классификации. Вторая глава содержит сведения о способах представления знаний в экспертных системах. Рассмотрены модели представления знаний: логическая, продукционная, фреймовая, семантическая, нейронная и нечеткая.

В третьей главе описаны продукционные экспертные системы, основы теории приближенных рассуждений и пример реализации инструментальной экспертной системы.

Четвертая глава посвящена вопросам разработки нейросетевых экспертных систем.

Пятая глава содержит сведения о принципах построения нечетких экспертных систем.

В шестой главе содержатся сведения о гибридных экспертных системах, включая основные термины и пример реализации.

Седьмая глава посвящена онтологиям, используемым для описания крупных проектов с отношениями между объектами.

В восьмой главе содержатся сведения о построении экспертных системах на базе прецедентов, которые нашли свое применение в юриспруденции и других отраслях народного хозяйства с дефицитом фактического материала, необходимого для создания ЭС.

ОСНОВНЫЕ ПОНЯТИЯ И НАЗНАЧЕНИЕ ЭКСПЕРТНЫХ СИСТЕМ

1.1. Понятие экспертной системы

В 60-70 годах прошлого столетия в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название *«экспертные системы»* (ЭС).

Искусственный интеллект представляет собой область информатики, которая занимается разработкой интеллектуальных компьютерных систем, т.е. систем, обладающих возможностями, которые традиционно связаны с человеческим разумом: пониманием языка, обучением, способностью рассуждать, решать проблемы и т.д.

Цель исследований по экспертным системам состоит в разработке программ, предназначенных для решения трудно формализуемых задач, не уступающих по качеству и эффективности решениям, которые получает эксперт.

Исследователи в области ЭС для названия своей дисциплины часто используют также термин «*инженерия знаний*» как «привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов» [1,5,6]. Процесс организации знаний в базу знаний и построения экспертной системы называют *инженерией знаний*.

Технология экспертных систем используется для решения различных типов задач: интерпретации предсказания, диагностики, планирования, конструирования, контроля, отладки, инструктажа, управления.

Решения экспертных систем обладают «прозрачностью» и могут быть объяснены пользователю на качественном уровне. Это свойство экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях. Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом. В литературе приводятся разные

определения ЭС. В настоящее время отсутствует единое определение ЭС. Наиболее популярными являются следующие.

Экспертная система – это программа для компьютера, которая оперирует со знаниями в определенной предметной области с целью выработки рекомендаций или решения проблем [1].

Экспертная система - это сложный программно-аппаратный комплекс, аккумулирующий знания специалистов в конкретной предметной области и тиражирующий этот эмпирический опыт для консультаций менее квалифицированных пользователей [1,5,6].

Экспертная система - это система, основанная на знаниях (knowledge-based system).

Строго говоря, экспертная система – это более широкое понятие. Система, основанная на знаниях, это любая система, процесс работы которой основан на применении правил отношений к символическому представлению знаний. Например, программа, способная рассуждать о погоде, будет системой, основанной на знаниях даже в том случае, если она не способна выполнить метеорологическую экспертизу. Программа, способная давать прогноз погоды, имеет право называться метеорологической ЭС.

Огромный интерес к ЭС со стороны пользователей вызван, по крайней мере, тремя причинами.

1. Пользователи ориентированы на решение широкого круга задач в неформализованных областях, малодоступны для вычислительной техники.

2. С помощью ЭС специалисты, не знающие технологии программирования, могут самостоятельно разрабатывать интересующие их приложения, что позволяет резко расширить сферу использования вычислительной техники.

3. ЭС при решении практических задач достигает результатов, не уступающих, а иногда и превосходящих возможности людей-экспертов.

Экспертные системы предназначены для решения так называемых неформализованных задач, обладающих одной или несколькими характеристиками:

- задачи не могут быть заданы в числовой форме (символьный вывод);
- цели не могут быть выражены в терминах точно определенной целевой функции;
- не существует алгоритмического решения задач;
- применение эвристического поиска решений в условиях, когда алгоритмическое решение существует, но его нельзя использовать из-за ограниченности ресурсов (параметры времени и памяти).

Неформализованные задачи обычно характеризуются аспектами неполной информации: *неточностью, неопределенностью, нечеткостью* проблемной области [3]. Кроме того, наблюдается большая размерность пространства решений (перебор при поиске решения весьма велик) и динамически изменяющиеся данные. Неточные данные задаются в интервальной форме [D-E; D+E]. Неопределенность - это неизвестное значение истинности высказывания. Нечеткость данных связана с заданием функции принадлежности элементов множества.

1.2. Компоненты типовой статической экспертной системы

Типовая структура статической ЭС состоит из следующих основных компонентов, приведенных на рис. 1.1: модуля интерфейса пользователя; базы знаний (БЗ); машины логического вывода; модуля объяснений; интеллектуального редактора базы знаний.

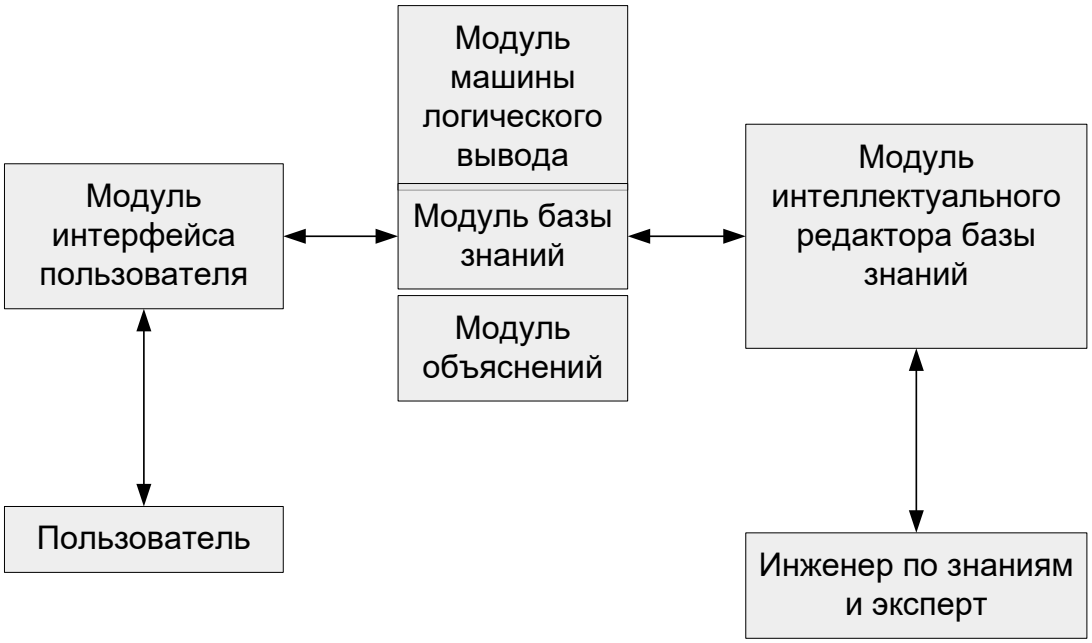


Рис. 1.1. Структура статической экспертной системы

Пользователь - специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям - специалист по искусственному интеллекту, выступающий в роли промежуточного буфера между экспертом и БЗ.

Модуль интерфейса пользователя - программа, реализующая диалог пользователя с ЭС как на стадии ввода информации, так и получения результатов.

Модуль базы знаний - ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному языку). Параллельно такому «человеческому» представлению существует БЗ во внутреннем «машинном» представлении.

Модуль машины логического вывода – программа, моделирующая ход рассуждения эксперта на основании знаний, имеющихся в БЗ [1].

Модуль объяснений - программа, позволяющая пользователю получить ответы на вопросы: "Как была получена та или иная рекомендация?" и "Почему система приняла такое решение?"

Модуль интеллектуального редактора БЗ - программа, представляющая инженеру по знаниям возможность создавать и редактировать БЗ в диалоговом режиме.

В разработке ЭС участвуют *эксперт проблемной области, инженер по знаниям и программист*.

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

В режиме *приобретения знаний* общение с ЭС осуществляет (через посредничество инженера по знаниям) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области [1].

В режиме *консультации* общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения [4]. Необходимо отметить, что в зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (в этом случае он обращается к ЭС за результатом, не умея получить его сам) или быть специалистом (в этом случае пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу).

В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память.

1.3. История развития экспертных систем

Наиболее известные ЭС, разработанные в 60-70-х годах, стали в своих областях уже классическими. По происхождению, предметным областям и по преемственности применяемых идей, методов и инструментальных программных средств ЭС можно разделить на несколько семейств [1, 5, 6].

1. META-DENDRAL. ЭС DENDRAL позволяет определить наиболее вероятную структуру химического соединения по экспериментальным данным. Она автоматизирует процесс приобретения знаний и генерирует правила построения фрагментов химических структур.

2.MYCIN-EMYCIN-TEIREIAS-PUFF-NEOMYCIN. Это семейство медицинских ЭС и сервисных программных средств для их создания.

3. PROSPECTOR-KAS. Экспертная система PROSPECTOR предназначена для поиска (предсказания) месторождений на основе геологических анализов. KAS - система приобретения знаний для PROSPECTOR.

4. CASNET-EXPERT. Система CASNET - медицинская ЭС для диагностики выдачи рекомендаций по лечению глазных заболеваний. На ее основе разработан язык инженерии знаний EXPERT, с помощью которой создан ряд других медицинских диагностических систем.

5. HEARSAY-HEARSAY-2-HEARSAY-3-AGE. Первые две системы этого ряда являются развитием интеллектуальной системы распознавания слитной человеческой речи, слова которой берутся из заданного словаря. Эти системы отличаются оригинальной структурой, основанной на использовании доски объявлений - глобальной базы данных, содержащей текущие результаты работы системы. В дальнейшем на основе этих систем были созданы инструментальные системы HEARSAY-3 и AGE (Attempt to Generalize - попытка общения) для построения ЭС.

6. Системы AM (Artifical Mathematician- искусственный математик) и EURISCO были разработаны в Станфордском университете для исследовательских и учебных целей.

В экспертную систему AM первоначально было заложено около 100 правил вывода и более 200 эвристических алгоритмов обучения, позволяющих строить произвольные математические теории и представления. Дальнейшее развитие системы замедлилось, и было отмечено, что, несмотря на проявленные на первых порах «математические способности», система не может синтезировать новых эвристических правил, т.е. ее возможности определяются только теми эвристиками, что были в нее изначально заложены.

При разработке системы EURISCO была предпринята попытка преодолеть указанные недостатки системы AM. Как и в начале эксплуатации AM, первые результаты, полученные с помощью EURISCO, были эффективными. Сообщалось, что система EURISCO может успешно участвовать в очень сложных играх. С ее помощью в военно-стратегической игре, проводимой ВМФ США, была разработана стратегия, содержащая ряд оригинальных тактических ходов. Например, предлагалось взрывать свои корабли, получившие повреждения. При этом корабли, оставшиеся неповрежденными, получает необходимое пространство для выполнения маневра.

Однако через некоторое время обнаружилось, что система не всегда корректно переопределяет первоначально заложенные в нее правила. Так, например, она стала нарушать строгое предписание обращаться к программистам с вопросами только в определенное время суток. Экспертная система EURISCO, так же, как и ее предшественница, остановилась в своем развитии, достигнув предела, определенного ее разработчиком.

Некоторые примеры успешного применения экспертных систем:

- только в США ежегодный доход от продаж инструментальных средств разработки ЭС составлял в начале 90-х годов 300 - 400 млн. долларов, а от применения ЭС – от 80 до 90 млн. долларов [1];
- XCON (США) экономит 70 млн. долларов в год [1] благодаря экспертной системе XCON/XSEL, которая по заказу покупателя составляет конфигурацию вычислительной системы VAX. Использование экспертной системы сократило количество ошибок от 30% (допускал человек) до 1% (допускает экспертная система);
- фирма Sira (США) сократила затраты на строительство трубопровода в Австралии на 40 млн. долларов [1] за счёт экспертной системы, управляющей трубопроводом. Экспертная система реализована на базе инструментальных средств G2 (фирма Gensym);
- фирма Monsanto (США) ежегодно экономит от 250 до 500 тыс. долларов благодаря экспертной системе выявления и блокирования неисправностей в нефтехимической промышленности. Экспертная система реализована на базе инструментальных средств G2 (фирма Gensym);
- фирма Aetha Insurance (США) уже сэкономила более 5 млн. долларов, а общий планируемый эффект составит 15 - 20 млн. долларов благодаря экспертной системе, используемой для моделирования страховых исков, обрабатываемых компанией. Экспертная система, реализованная на базе инструментальных средств G2 (фирма Gensym), позволяет находить в деятельности компании неэффективные процессы и рабочие потоки и производить оперативные изменения для увеличения продуктивности работы.

Среди отечественных разработок можно отметить следующие экспертные системы [1, 11]:

1. Экспертная диагностическая система, предназначенная для помощи лицам, принимающим решения при мониторинге и управлении сложными

объектами и процессами различной природы в условиях жестких временных ограничений.

2. ЭС ДИАГЕН для диагностики наследственных болезней.

3. ЭС ДИН для диагностики неотложных состояний у детей в условиях ограничений на проведение дополнительных исследований, обусловленных тяжестью состояния или недостатком диагностической аппаратуры.

4. ЭС ВЕСТ-СИНДРОМ для диагностики эпилепсии.

5. ЭС поддержки проектирования процесса нормализации пленочных резисторов.

Большинство публикаций по отечественным ЭС приходится на медицину. Использование консультирующих ЭС, основанных на знаниях, особенно важно для начинающих врачей, не обладающих личным опытом. Такие ЭС не пропустят следующие ситуации:

- часто встречающаяся болезнь с нетипичными симптомами;
- симптомы-миражи, которые не имеют никакого отношения к заболеванию;
- напрасный поиск «зебр» (редких болезней) с самого начала диагностического процесса.

В последнее время наметилась тенденция разработки ЭС на базе нейронных сетей, нечеткой логики, которые способны обучаться и использовать интуитивный опыт экспертов [3,9,10]. Также отмечается в качестве перспективного направления создание гибридных экспертных систем [9].

1.4. Классификация экспертных систем

Наиболее популярная классификация, предложенная Т.А. Гавриловой и В.Ф. Хорошевским в [1], подразделяет ЭС на четыре основных класса: по типу решаемой задачи; по связи с реальным временем; по типу используемой ЭВМ; по степени интеграции.

Классификация по типу решаемой задачи: интерпретация данных; диагностика; мониторинг; проектирование; прогнозирование; планирование; обучение, управление.

Интерпретация в переводе с латинского понимается как истолкование, объяснение, перевод на более понятный язык или процесс определения смысла данных, например, определение основных свойств личности по результатам тестирования в системах АВАНТЕСТ и МИКРОЛЮШЕР; идентификация типов океанских судов по результатам аэрокосмического сканирования (SIAP).

Диагностика – это процесс соотнесения объекта с некоторым классом объектов и(или) обнаружение неисправности в объекте, например, диагностика и терапия сужения коронарных сосудов (ANGY).

Мониторинг – это непрерывная интерпретация данных в реальном масштабе времени, и сигнализация о выходе параметров за допустимые пределы, например, мониторинг за работой атомного реактора (REACTOR), контроль аварийных датчиков на химическом заводе (FALCON).

Проектирование – это процесс создания проекта и подготовка спецификаций на объект. Под спецификацией понимается комплект разрабатываемой документации, например, проектирование СБИС (CADHELP).

Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных, например, предсказание погоды (WILLARD), прогнозы в экономике (ECON).

Планирование - это нахождение плана действий, относящихся к объектам, способным выполнять некоторые функции, например, планирование поведения робота (STRIPS), планирование промышленных заказов (ISIS).

Обучение - это использование персонального компьютера для обучения какой-либо дисциплине, например, обучение языку программирования LISP с помощью программы «Учитель ЛИСП».

Управление - это функция организованной системы, поддерживающая определенный режим деятельности, например, управление системой календарного планирования Project Assistant.

Классификация по связи с реальным временем подразделяет ЭС на статические, квазидинамические и динамические [1].

Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени, например, ЭС диагностики состояния автомобиля.

Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Например, микробиологическая ЭС, которая считывает показания датчиков с технологического процесса один раз в 4-5 часов.

Динамические ЭС работают в режиме реального времени и производят непрерывную интерпретацию поступающих в систему данных, например, ЭС мониторинга за состоянием больного в реанимации.

Классификация по типу используемой ЭВМ выделяет ЭС, построенные с использованием: суперЭВМ (например, GRAY); мейнфреймов (например, System 390 серии G5); рабочих станций (SUN); миниЭВМ (VAX); персональных компьютеров; карманных персональных компьютеров.

Классификация по степени интеграции с другими программами подразделяет ЭС на автономные и интегрированные.

Автономные ЭС работают непосредственно в режиме консультаций с пользователем, при этом для решения задач не требуется привлекать традиционные методы обработки данных (расчеты, моделирование, статистическая обработка).

Интегрированные ЭС представляют собой программный комплекс, агрегирующий стандартные пакеты программ (например, математическую статистику) или системы управления базами данных и средства манипулирования знаниями.

1.5. Классификация инструментальных средств создания экспертных систем

В настоящее время известны три основные направления проектирования экспертных систем [1,5,6]:

1. Экспертные системы, выполненные в виде отдельных программ, на некотором алгоритмическом языке, база знаний которых является непосредственно частью этой программы. Как правило, такие системы предназначены для решения задач в одной конкретной предметной области. При построении таких систем применяются как традиционные процедурные языки PASCAL, C++ и т.д., так и специализированные языки искусственного интеллекта LISP, PROLOG.

2. Экспертные системы, построенные с использованием оболочек ЭС, которые представляют собой программный продукт, обладающий средствами представления знаний для определенных предметных областей. Задача пользователя заключается в формализации и вводе знаний с использованием специальной оболочки. Недостатком этих систем можно считать невозможность охвата одной системой всех существующих предметных областей. Примером могут служить ИНТЕР-ЭКСПЕРТ, PC+, VP-Expert, EMYCIN.

3. Экспертные системы, построенные с помощью генератора экспертных систем, который представляет собой мощный программный комплекс, предназначенный для получения оболочек, ориентированных на то или иное представление знаний в зависимости от рассматриваемой предметной области. Примеры этой разновидности - системы KEE, ART.

Система «Expert» является скелетным языком инженерии знаний, которая использует схему представления знаний, основанную на правилах, и имеет ограниченный механизм вывода, организованный по принципу прямой цепочки рассуждений и делающий его пригодным для задач типа диагностики и классификации. В этой системе имеются встроенные блоки построения объяснений, приобретения знаний и контроля непротиворечивости, которые ускоряют разработку. Блок контроля непротиворечивости хранит базу данных репрезентативных случаев с известными заключениями и использует их для тестирования экспертной системы после того, как инженер знаний добавляет новые правила. Если в каком-то случае не удаётся получить правильных рассуждений, то система «Expert» представляет процесс рассуждений до этого случая, чтобы инженер знаний мог понять, какое новое правило привело к неожиданным результатам [11,12].

ИНТЕР-ЭКСПЕРТ – это интегрированная система ведения баз данных и баз знаний. Она представляет собой среду для создания экспертных систем в области делопроизводства и экономики, создания и ведения электронных ведомостей, использование деловой графики, составление отчётов и электронных таблиц.

В системе реализован принцип синергизма (каждой части ИНТЕР-ЭКСПЕРТ доступны все данные, имеющиеся в системе). Экспертная система, созданная с помощью ИНТЕР-ЭКСПЕРТ, имеет доступ к информации, хранимой в базе данных, электронной ведомости, другом наборе правил, таблице, файле графиков.

ACQUIRE - это законченная среда для разработки и поддержки интеллектуальных прикладных программ. Система содержит в себе методологию пошагового представления знаний, что позволяет специалистам в проблемной области непосредственно участвовать в процессе приобретения, структурирования и кодирования знания. Прямое участие специалиста в проблемной области улучшает качество, законченность и точность приобретенного знания, снижает время разработки и эксплуатационные расходы.

Особенностью оболочки является структурированный подход к приобретению знаний. Модель приобретения знаний основана на распознавании образов. Знания представлены как объекты, а продукционные правила - в табличной форме. Оболочка позволяет выполнять обработку неопределенных качественных знаний, содержит средства вывода и документацию баз знаний в среде гипертекста.

Инструментальная система G2 фирмы Gensym предлагает графическую, объектно-ориентированную среду для создания интеллектуальных прикладных программ, которые контролируют, диагностируют и управляют динамическими событиями в сетевых и моделируемых средах [http://www.gensym.com].

Инструментальная система G2 предназначена для создания ЭС на базе продукционных правил и других моделей (процедур) с использованием структурированного естественного языка. Инструментальная экспертная система G2 является основой всех прикладных программ фирмы Gensym. Она позволяет использовать визуальную среду программирования для создания интеллектуальных прикладных программ управления. NeurOn-Line и другие программы фирмы позволяют пользователям создавать нейросетевые прикладные программы. G2 совмещает выполнение правил и процедур в текущий момент времени со способностями рассуждений спустя некоторое время. Руководство по G2 позволяет пользователям создавать графические интерфейсы и системы диагностирования в реальном

масштабе времени. Компания Telewindows Gensym's создала более мощную универсальную среду клиент/сервер, которая позволяет пользователям совместно использовать прикладные программы на основе G2.

Фирма Gensym также предлагает мосты (программы) для связи с другими программам (на языках С и АДА) и системы передачи и обработки данных о движущихся объектах в реальном времени, включая реляционные базы данных, распределенные системы управления и программируемые логические системы.

COMDALE/C - экспертная система реального времени, предназначенная для наблюдения и контроля над процессами в условиях производства [http://www.comdale.com].

+COMDALE/C позволяет вырабатывать рекомендации, заключения об управляющих воздействиях в непрерывном процессе принятия решения. Она обрабатывает неопределенные знания и данные и имеет открытую архитектуру. Её основные характеристики - объектно-ориентированная конфигурация; возможность организации работы в сети; обработка прерываний; хранение и обработка данных; поддержка работы с базой данных в реальном масштабе времени; интерфейсы с системами передачи данных.

+COMDALE/X - консультационная экспертная система, которая работает в режиме реального времени. Для принятия решения система организует диалог с пользователем. COMDALE/X совместно с системой COMDALE/C используется как инструмент разработки экспертных систем реального времени. COMDALE/X позволяет включить гипертекст в экспертную систему, что позволяет создавать hyper-справочники с удобным интерфейсом.

FLEX - гибридная экспертная система, работающая на различных платформах [http://vvv.com/ai/]. Система предлагает фреймовое, процедурное и продукционное представление знаний.

FLEX чередует прямой и обратный методы поиска решений, множественное наследование свойств, присоединенные процедуры, автоматическую систему вопросов и ответов. Правила, фреймы и вопросы написаны на естественном англо-подобном языке. Язык спецификаций (KSL) позволяет разрабатывать легко читаемые и простые в поддержке базы знаний. Экспертная система FLEX разработана на языке Пролог и использовался в многочисленных коммерческих финансовых экспертных системах.

EXSYS 3.0 (Exsys Inc.) - оболочка, предназначенная для создания экспертных прикладных систем классификационного типа. Использует прямую и обратную цепочки вывода, организует базу продукционных правил, имеющих вид “ЕСЛИ-ТО-ИНАЧЕ”. EXSYS позволяет использовать данные из других программ, электронных таблиц, баз данных, поддерживает математические вычисления, строковые и числовые переменные [11, 12].

NEXPERT OBJECT (Neuron Data Corp.) - это мощная система, базирующаяся на использовании правил. Она поддерживает различные типы правил и комбинации прямого и обратного выводов [4, 12]. Система может автоматически строить сеть правил в графическом изображении, что позволяет наблюдать, каким образом правила связаны друг с другом. NEXPERT OBJECT имеет возможности представления обоих типов фреймов (с наследованием и без наследования) и механизм сопоставления с образом. В этой системе большое внимание уделено графическому представлению баз данных и объясняющих возможностей, что делает возможным организацию естественных и дружественных интерфейсов разработчика и конечного пользователя. Средство написано на языке С.

Все ведущие производители инструментальных средств экспертных систем (Gensym, Inference, Intellicorp, Neuron Data) признали и реализуют проблемно/предметно-ориентированные инструментальные средства.

Наибольшего успеха в этом направлении добилась фирма Gensym со своими продуктами: G2, GDA, DSP, NeurOn-Line, Rethink [1,5,6].

Развитие ориентированных (предметно-ориентированных) инструментальных средствах проводится в следующих направлениях:

- инструментальные средства для динамических экспертных систем реального времени, используемых в управлении технологическими процессами и имитационном моделировании [1, 4];
- инструментальные средства для систем-советчиков;
- инструментальные средства для систем, основанных на прецедентах.

В области инструментальных средств и динамических экспертных систем доминирующие позиции занимает фирма Gensym (G2, GDA, DSP, NOL), затем идут Talarian (RTworks) и Comdale Technologies (Comdale/C, Comdale/X).

MYCIN – экспертная система для диагностики и лечения инфекционных заболеваний [1].

Разработан скелетный язык, иначе - оболочка ЭС EMYCIN [1]. Декларативные знания системы MYCIN описываются в виде «объект-атрибут-значение». Каждой тройке приписывается коэффициент уверенности, определяющий степень надежности знаний. Процедурные знания описаны в виде классического правила продукции. Механизм логического вывода основан на обратной цепочке рассуждений. Поиск производится в иерархически упорядоченном пространстве состояний.

В оболочке EMYCIN [1] по отношению к MYCIN усилена функция редактирования БЗ, доведена до высокого уровня система объяснения хода решения задачи, а также аппарат обучения системы. Оболочка разработана на языке Фортран.

OPS-5 - универсальный язык инженерии знаний, предназначенный для разработки ЭС, используемых в коммерческих приложениях.

Декларативные знания в системе описаны в виде «объект-атрибут-значение». Процедурные знания описаны в виде классических правил продукции.

В Российском научно-исследовательском институте информационных технологий и систем автоматизированного проектирования (РосНИИ ИТ и АП) разработан комплекс ЭКО [1]. Наиболее успешно комплекс применяется для создания экспертных систем, решающих задачи диагностики (технической и медицинской), эвристического оценивания (риска и надёжности и т.д.), качественного прогнозирования, а также обучения.

На основе комплекса ЭКО было разработано более 100 прикладных экспертных систем. Среди них можно отметить следующие:

- поиск одиночных неисправностей в персональном компьютере;
- оценка состояния гидротехнического сооружения (Чарвакская ГЭС);
- подготовка деловых писем при ведении переписки с зарубежными партнёрами;
- проведение скрининговой оценки иммунологического статуса;
- оценка показаний микробиологического обследования пациента, страдающего неспецифическими хроническими заболеваниями легких;
- психодиагностика в психосоматике, а также другие системы.

Структура комплекса ЭКО включает три компонента. Ядром комплекса ЭКО является интегрированная оболочка экспертных систем, которая обеспечивает быстрое создание эффективных приложений для решения задач анализа.

Оболочка функционирует в двух режимах: в режиме приобретения знаний и в режиме консультации (решения задач). В первом режиме разработчик экспертных систем средствами диалогового редактора вводит в базу знаний описание конкретного приложения в терминах языка

представления знаний оболочки. Во втором режиме оболочка решает конкретные задачи пользователя в диалоговом или пакетном режиме.

Для расширения возможностей оболочки по работе с глубинными знаниями комплекс ЭКО может быть дополнен компонентом К-ЭКО (конкретизатором знаний), который позволяет описывать закономерности в проблемных средах в терминах общих объектов и правил. К-ЭКО используется на этапе приобретения знаний вместо диалогового редактора оболочки для преобразования общих описаний в конкретные сети вывода, допускающие эффективный вывод решений средствами оболочки ЭКО.

Третий компонент комплекса ЭКО - система ИЛИС, позволяющая создавать экспертную систему в проблемных средах за счёт индуктивного обобщения данных (примеров) и предназначенная для использования в тех приложениях, где отсутствие правил, отражающих закономерности в проблемной среде, возмещается экспериментальными данными.

Средства представления знаний в оболочке ЭКО представляют собой совокупность нескольких моделей в базе знаний, каждая из которых описывает отдельное конкретное приложение или его компонент. Отдельная модель включает описание проблемной среды и знания о порядке решения задач. Описание проблемной среды состоит из описаний атрибутов и правил вывода. Атрибуты используются для описания состояний предметной области.

Оболочка работает со статистическими проблемными средами (значения атрибутов не изменяются в ходе решения задачи), в которых предметная область может быть описана с помощью априорно заданного набора атрибутов. Не допускается динамическое создание атрибутов во время решения задачи. Средства комплекса позволяют представлять качественные (символьные) и количественные (числовые) характеристики предметной области.

Высказывания типа «А есть В» называются утверждениями о состоянии предметной области. В этом высказывании посылка А представляет атрибут (качественную характеристику), а заключение В - одно из возможных значений высказывания. Решение задачи сводится к получению значений некоторых целевых атрибутов и определению истинности некоторых целевых утверждений. Оболочка позволяет работать с неточно и нечётко определёнными знаниями. С каждым утверждением о состоянии предметной области связывается коэффициент определённости, который характеризует степень уверенности в его истинности. Вывод решения заключается в нахождении коэффициентов определённости некоторых целевых утверждений, указанных разработчиком экспертных систем. Для построения вывода в условиях неопределённости может использоваться байесовский подход. Коэффициенты определённости утверждений - это действительные числа, принимающие значения от минус 5,00 до 5,00. Коэффициенту определённости D(H) утверждения H можно дать следующую интерпретацию [18,19]:

- если точно известно, что H истинно, то D(H) = 5,00;
- если точно известно, что H ложно, то D(H) = -5,00;
- если H может быть с одинаковой уверенностью истинно или ложно, то D(H) = 0,00;
- если H скорее истинно, чем ложно, то 0,00 < D(H) < 5,00, причём D(H) тем больше, чем больше уверенность в истинности H;
- если H скорее ложно, чем истинно, то -5,00 < D(H) < 0,00, причём D(H) тем меньше, чем больше уверенность в ложности H.

Утверждения и числовые атрибуты модели называются целями, а символьные атрибуты, представляющие собой множество утверждений, называются сложными целями. Значения целей определяются с помощью простых и сложных правил вывода:

- простой вопрос позволяет получать либо значение числового

атрибута, либо коэффициент определённости отдельного утверждения;

- сложный вопрос позволяет получить распределение коэффициентов определённости по всем возможным значениям символьного атрибута;
- альтернативный вопрос используется в тех случаях, когда известно, что символьный атрибут имеет точно одно значение из множества возможных значений;
- дистрибутивный вопрос используется в тех случаях, когда символьный атрибут может иметь одновременно несколько значений или ни одного;
- арифметические правила предназначены для вычисления значений числовых атрибутов, а также получения коэффициентов определённости утверждений;
- логические правила предназначены для вычисления коэффициентов определённости утверждений по формулам нечёткой логики. При этом значение логического выражения (в условии правила) присваивается коэффициенту определённости целевого утверждения правила;
- байесовские правила предназначены для вычисления коэффициентов определённости тех утверждений, об истинности которых можно судить по выполнению ряда факторов (симптомов), имеющих разную значимость.

Для определения значения одной цели разработчик экспертной системы может задавать несколько правил, образующих в модели упорядоченный список правил вывода данной цели.

Сценарий консультации представляет собой последовательность предложений, каждое из которых может иметь условие применимости. В

рамках каждого предложения возможно выполнение одного из следующих действий:

- вывести значение цели;
- выдать сообщение пользователю;
- выдать сообщение внешней программе;
- сбросить выведенные результаты (СБРОС);
- перейти к выполнению другого предложения;
- принять информацию от внешней программы;
- передать информацию о результатах решения внешней программе;
- создать контрольную точку консультации;
- загрузить контрольную точку;
- обратиться к подмодели, решающей некоторую частную подзадачу, передать ей параметры и получить выведенные в подмодели результаты;
- закончить консультацию с сообщением (СТОП).

Рассмотрим построение сети вывода на основе содержащихся в модели описаний правил и атрибутов. В процессе построения сети из числовых атрибутов, утверждений и правил строится сеть вывода, в явном виде включающая все связи между атрибутами и утверждениями, обусловленные правилами вывода. Сеть вывода образует граф с вершинами двух типов: первые вершины соответствуют простым целям; вторые вершины соответствуют простым правилам. Дуги представляют собой связи между простыми целями и простыми правилами. Особенность таких простых правил состоит в том, что применение одного влечёт применение остальных.

Стратегии управления в оболочке ЭКО характеризуются следующими моментами. В начале решения задачи выбирается первое предложение сценария, затем проверяется условие его применимости; если условие выполнено, выполняется указанное в нём действие. Если в ходе проверки

возникает потребность в значении некоторой цели, то анализ условия приостанавливается и требуемое значение выводится из сети вывода. После обработки первого предложения сценария осуществляется переход к следующему предложению и т.д., пока не будет обнаружено действие СТОП или пока не будет исчерпан сценарий. В оболочке ЭКО используется стратегия обратного рассуждения от целей к данным в глубину: при рассмотрении некоторой цели делается попытка найти в сети вывода путь от вершин, представляющих исходные данные консультации, к вершине соответствующей выбранной цели. Путь считается найденным, если выполнены условия применимости всех правил, соответствующих дугам этого пути. В том случае, когда оказалось сразу несколько применимых правил, используется первое применимое правило. Значения целей вычисляются один раз и не могут быть изменены иначе, как по команде СБРОС.

Ввод знаний в базу знаний ЭКО осуществляется средствами диалогового редактора, предоставляющего:

- навигацию по базе знаний;
- шаблоны ввода всех конструкций языка для представления знаний ЭКО;
- синтаксический и семантический контроль вводимой информации;
- тестирование и компиляцию моделей;
- генерацию текстовых и гипертекстовых отчётов по базе знаний модели.

Помимо базы знаний разработчик экспертных систем может сформировать контекстно-зависимую помощь к приложению в виде иллюстрированного гипертекста.

Решение задач осуществляется в режиме консультации, при этом предоставляются следующие возможности:

- решение конкретной задачи на основе выбранной модели с формированием объяснений;
- просмотр информации о правилах в моделях;
- сброс значений всех целей либо значений всех выведенных целей
- (отмена всех значений, не являющихся исходными данными);
- получение трассы решения задачи;
- запись протокола консультации в файл и создание и загрузка контрольных точек.

Решение осуществляется в ходе диалога экспертной системы с пользователем. На экран выдаются сообщения в соответствии со сценарием консультации, а также задаются вопросы, описанные в применяемых правилах. Если пользователь не имеет информации, позволяющей ему ответить на вопрос, то он может ответить НЕ ЗНАЮ. Тогда будут применяться другие правила вывода искомого значения, если такие правила предусмотрены в модели.

Контрольные вопросы

1. Понятие экспертной системы и ее отличие от другой программы.
2. Мотивация интереса пользователей к ЭС.
3. Характеристика неформализованной задачи.
4. Классификация ЭС.
5. Классификация инструментальных средств разработки ЭС.
6. Назначение основных модулей ЭС.
7. Понятие инженерии знаний и инженера знаний.
8. Примеры создания ЭС.
9. Характеристика ЭС ЭКО.
10. Отличие ЭС от системы, основанной на знаниях.

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ

2.1. Данные и знания

Данными называют информацию фактического характера, описывающую объекты, процессы и явления предметной области, а также их свойства.

Данные - это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

В процессе компьютерной обработки данные проходят следующие этапы преобразований:

- исходная форма существования данных (результаты наблюдений и измерений, таблицы, справочники, графики и т.п.);
- представление на специальных языках описания данных, предназначенных для ввода в ЭВМ;
- базы данных на машинных носителях.

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

D1 - данные как результат измерений и наблюдений;

D2 - данные на материальных носителях информации (таблицы, протоколы, справочники);

D3 - модели (структуры) данных в виде диаграмм, графиков, функций;

D4 - данные в компьютере на языке описания данных;

D5 - базы данных на машинных носителях информации.

Знание - это совокупность сведений, образующих целостное описание, соответствующее определенному уровню осведомленности об описываемой проблеме.

Основное отличие знаний от данных в том, что данные описывают лишь конкретное состояние объектов или группы объектов в текущий

момент времени, а знания кроме данных содержат сведения о том, как оперировать этими данными.

Знания - это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

Единой принятой терминологии не выработано. Большинство исследователей в области искусственного интеллекта предлагают следующее определение.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

При обработке на ЭВМ знания формируются аналогично данным:

Z1 – знания в памяти человека как результат мышления;

Z2 – материальные носители знаний (учебники, методические пособия);

Z3 – поле знаний - условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

Z4 – знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы и другие модели);

Z5 – база знаний на машинных носителях информации.

В базе знаний ЭС знания должны быть обязательно структурированы и описаны терминами одной из модели знаний. Выбор модели знаний - это наиболее сложный вопрос в проектировании ЭС, так как формальное описание знаний оказывает существенное влияние на конечные характеристики и свойства ЭС.

В рамках одной БЗ все знания должны быть однородные и описаны простыми для понимания терминами. Однородность описания диктуется тем, что в рамках ЭС должна быть разработана единая процедура

логического вывода, которая манипулирует знаниями на основе стандартных типовых подходов. Простота понимания определяется необходимостью постоянных контактов с экспертами предметной области, которые не обладают достаточными знаниями в компьютерной технике.

Знания подразделяются, с точки зрения семантики, на факты и эвристики. Факты, как правило, указывают на устоявшиеся в рамках предметной области обстоятельства, а эвристики основываются на интуиции и опыте экспертов предметной области.

По степени обобщенности описания знания подразделяются на поверхностные и глубинные [1,5,6].

Поверхностные знания описывают совокупности причинно-следственных отношений между отдельными понятиями предметной области.

Глубинные знания характеризуют абстракции, аналогии, образцы, которые отображают глубину понимания всех процессов, происходящих в предметной области.

Введение в базу глубинных представлений позволяет сделать систему более гибкой и адаптивной, так как глубинные знания являются результатом обобщения разработчиком или экспертом первичных примитивных понятий.

По степени отражения явлений знания подразделяются на «жесткие» и «мягкие». «Жесткие» - позволяют получить однозначные четкие рекомендации при задании начальных условий. «Мягкие» - допускают множественные, расплывчатые решения и многовариантные рекомендации [9].

Мягкими называют знания, которые используют для представления теорию нечетких множеств, нейронных сетей, вероятностных и приближенных рассуждений, генетических алгоритмов.

Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и

взаимосвязи между ними. Поэтому знания называют структурированными данными. Они могут быть поучены на основе обработки эмпирических данных. Они представляют результат мыслительной деятельности человека, направленной на обобщение опыта. Для того чтобы ввести знания в базу знаний любой интеллектуальной системы, их необходимо представить в определенной форме.

Знания в экспертных системах существуют в следующих формах:

- исходные знания (правила, выведенные на основе практического опыта, математические и эмпирические зависимости, отражающие взаимосвязи между фактами, описывающие изменение фактов с течением времени, графы и т.п.);
- описание исходных знаний средствами выбранной модели представления знаний (множество логических формул, продукционных правил, фреймов, семантических и нейронных сетей);
- представление знаний структурами данных;
- базы знаний на машинных носителях.

2.2. Классификация знаний

Существует множество классификаций знаний. По своей природе знания подразделяются на декларативные и процедурные.

Декларативные знания представляют собой описания фактов и явлений, фиксируют наличие или отсутствие таких фактов.

Процедурные знания – это описания действий, которые возможны для манипулирования фактами и явлениями при достижении поставленных целей. К ним относятся методики, правила и т.п.

Существует множество способов определять понятия [1]. Один из широко применяемых способов основан на идее интенсионала. *Интенсионал* понятия - это определение его через соотнесение с понятием

более высокого уровня абстракции с указанием специфических свойств. Интенсионалы формулируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому объекту. Это определение через данные, или *экстенсионал понятия*.

Пример 1. Понятие «персональный компьютер». Его интенсионал: персональный компьютер - это дружественная и относительно недорогая ЭВМ, которую можно разместить на столе.

Экстенсионал этого понятия: «Персональный компьютер - это IBM PC». Для хранения данных используются базы данных (для них характерны большой объем и относительно небольшая удельная стоимость информации), для хранения знаний - базы знаний (небольшого объема, но исключительно дорогие информационные массивы).

Знания могут быть классифицированы по следующим категориям:

- *поверхностные* - знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области;
- *глубинные* - абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Пример 2. Поверхностные знания: «Если нажать на кнопку звонка, раздастся звук». «Если болит голова, то следует принять аспирин». Глубинные знания: «Принципиальная электрическая схема звонка и проводки». «Знания физиологов и врачей высокой квалификации о причинах, видах головных болей и методах их лечения».

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Исторически первичными были процедурные знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы.

Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знаний сосредоточилась в структурах, данных (таблицы, списки, абстрактные типы данных).

В результате повысилась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму, т.е. знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному языку и понятных специалистам.

По способу приобретения знания можно разделить на факты и эвристические правила, которые позволяют сделать выбор при отсутствии точных теоретических обоснований.

2.3. Логическая модель представления знаний

Распространенными моделями представления знаний в ЭС являются: логическая, продукционная, фреймовая, семантическая, нейронная.

Существуют множество моделей (или языков) представления знаний для предметных различных областей.

Логическая модель представления знаний основана на системе исчисления предикатов первого порядка. Предикатом называется некоторая связь, которая задана на наборе констант и переменных. Применение предикатов всеобщности, существования и операций "И", "ИЛИ", "НЕ", "Импликация", "Эквивалентность" позволяют описать многие знания в предметной области. Недостатками логической модели, основанной на системе исчисления предикатов первого порядка, являются следующие [1,5,6,9]:

– невозможность выразить через переменные другие предикаты;

– сложность логического вывода при больших массивах данных.

В основе языка предикатов первого порядка лежит понятие предикатов, т.е. логическая функция от одной или нескольких нелогических переменных. Функция может принимать значения истина (t) или ложь (f). В рамках логики утверждение считается истинным, если и относящееся к нему предположение считается истинным и заключение самого утверждения тоже истина.

Синтаксис языка предикатов включает: предикативные символы, символы переменных, константы, а также разделители ( ), [ ], “, ‘.

Предикативные символы используются для обозначения отношений. Объекты отношений записываются в круглые скобки после предикативного имени и называются аргументами. Полная запись отношения называется атомарной формулой. Например, атомарная формула «Является (Иванов, специалист по компьютерам)» содержит два предикативных терма.

Термы могут быть константами и переменными. Разрешено также в качестве термов использовать функции, которые обязательно должны быть определены в рамках предметной области. Разработчик ЭС заранее определяет, как интерпретировать порядок термов в отношении.

Допустимые выражения в исчислении предикатов, в частности атомарные формулы, называются правильно построенными функциями (ППФ). В языке предикатов для каждой ППФ обязательно определяется конкретная интерпретация. Как только для ППФ определена интерпретация, то говорят, что формула имеет значение «истина», если соответствующее утверждение истинно, в противном случае ППФ имеет значение «ложь».

Из формул можно составить предложение с помощью логических связок: конъюнкция, дизъюнкция, импликация, отрицание.

Конъюнкция обозначается символом «&» и реализует функцию «И»:

Учится (Иванов, Московский государственный университет) & располагается (университет, Москва).

Дизъюнкция обозначается символом «V» (ИЛИ) и реализует функцию не исключающего “ИЛИ”. Находятся (Иванов, аудитория 113) V Находится (Иванов, библиотека).

Импликация обозначается символом «→» и используется для представления утверждения типа “ЕСЛИ, ТО”. Владеть (Иванов, автомобиль) → марка (автомобиль, “Opel”).

Левая сторона импликации называется антецедент, правая - консеквент. Импликация имеет значение “ложь” только в одном случае, если антецедент имеет значение “истина”, а консеквент имеет значение “ложь”.

ППФ со знаком отрицания «~» перед ней называется отрицанием.

В языке предикатов атомарная формула может принимать только истинные значения или только ложные значения в зависимости от значений переменных, которые в нее входят. Для того чтобы при исчислении предикатов можно было манипулировать значениями переменных, потребовалось ввести понятие «квантор».

Квантор - это операция, в которой участвуют все значения переменной одного предиката. Квантор служит для указания меры, в какой экземпляры переменной, т.е. константы должны быть истинными, чтобы все значения в целом были истинными.

Различают квантор общности ∀(x) и квантор существования. Если перед предикатом записан квантор ∀x для какой-то переменной, например, ∀(x), то это означает, что значение предиката будет истинным только в том случае, если все значения переменной x будут истинными.

∀ (x) (специалист_по_ЭВМ (x) → программист(x))

Например, квантор существования ∃(x) означает, что для истинности предиката достаточно, чтобы только некоторые значения переменной или одно значение были истинными.

∃(x) (специалист_по_ЭВМ(x) & оптимист(x))

В рамках одного предиката можно использовать и кванторы общности, и кванторы существования, но для разных переменных.

Машинная реализация языка предиката первого порядка имеет ряд серьезных проблем, которые связаны с универсальностью аппарата логического вывода.

Первая проблема — *монотонность рассуждений*. В процессе логического вывода нельзя отказаться от промежуточного заключения, если становятся известными дополнительные факты, которые свидетельствуют о том, что полученные на основе этого заключения решения не приводят к желаемому результату.

Вторая проблема — *комбинаторный взрыв*. В процессе логического вывода невозможно применять оценочные критерии для выбора очередного правила. Бессистемное применение правил в расчете на случайное доказательство приводит к тому, что возникает много лишних цепочек ППФ, активных в определенный момент времени. Это чаще всего приводит к переполнению рабочей памяти.

Наиболее эффективной разработкой этого подхода является язык PROLOG. В нем принята обратная стратегия вывода, полностью реализованы все средства описания знаний с помощью предикатов. Для порождения новых высказываний используется операция резолюции. В качестве процедуры поиска решения, позволяющей устранить монотонность и комбинаторный взрыв, используют поиск в иерархически упорядоченном пространстве состояний.

К достоинствам логической модели можно отнести наличие теоретического материала по методам логического вывода и наличие стандартной типовой процедуры логического вывода (доказательства теорем).

Недостатками логической модели являются: сложность использования эвристик в процессе логического вывода, монотонность

логического вывода, возможность комбинаторного взрыва, слабая структурированность описаний.

2.4. Продукционная модель представления знаний

Продукционная модель или модель, основанная на правилах, позволяет представить знания в виде предложений типа" ЕСЛИ (условие), ТО (действие)".

Под «условием» (антецедентом) понимается некоторое предложение - образец, по которому осуществляется поиск в базе знаний, а под «действием» (консеквентом) - действия, выполняемые при успешном исходе поиска. Они могут быть промежуточными, выступающими далее, как условия, и терминальными или целевыми, завершающими работу системы.

Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения к данным). Существуют также системы с двунаправленными выводами. Данные - это исходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, выбирающий правила из продукционной базы знаний [1, 4, 5, 6, 9, 11].

Продукционная модель чаще всего применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения дополнений и изменений и простотой механизма логического вывода.

Основные достоинствами продукционных систем связаны с простотой представления знаний и организации логического вывода. К недостаткам продукционных систем можно отнести следующее:

- отличие от структур знаний, свойственных человеку;
- неясность взаимных отношений правил;

- сложность оценки целостного образа знаний;
- низкая эффективность обработки знаний;
- возникновение конфликтных ситуаций и необходимость их разрешения;
- при большом количестве правил возможность появления противоречивых правил требует разработки специальных механизмов контроля вводимых и редактируемых правил.

Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; " оболочки " или " пустые " ЭС - EXSYS Professional, Kappa, ЭКСПЕРТ, ЭКО, инструментальные системы ПИЭС), а также промышленных ЭС на его основе (например, ЭС, созданных средствами G2 и др.).

2.5. Фреймовая модель представления знаний

Фреймовая модель основана на теории Минского и представляет систематизированную психологическую модель памяти человека и его сознания. Термин *фрейм* происходит от английского слова frame, которое означает «каркас» или «рамка». Он был предложен Марвином Минским (одним из пионеров искусственного интеллекта) в 1979 году.

Фрейм – это абстрактный образ для представления некоего восприятия. В психологии известно понятие абстрактного образа. В искусственном интеллекте фреймом называют структуру данных для представления некоторого концептуального объекта. Из понятия фрейма ничего нельзя выбросить. Например, фрейм комнаты. Из описания комнаты нельзя ничего выбросить. Если удалить окна, то образ комнаты превращается в образ чулана [1, 5, 6].

(ИМЯ ФРЕЙМА:

(имя 1-го слота: значение 1-го слота),

(имя 2-го слота: значение 2-го слота),

…

(имя N-го слота: значение N-го слота)).

Структура фрейма может быть представлена в виде табл. 2.1.

Таблица 2.1

Структура фрейма

Имя фрейма			
Имя слота	Значение слота	Способ получения значения	Присоединенная процедура
Рост	175		

В данной таблице дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур. Например, слот ВОЗРАСТ может содержать имя процедуры, которая вычисляет возраст человека по дате рождения, записанной в другом слоте, и текущей дате.

Процедуры, располагающиеся в слотах, называются связанными или присоединенными процедурами. Вызов связанной процедуры осуществляется при обращении к слоту, в котором она помещена. Заполнителями слота могут быть правила продукций, используемые для определения конкретного значения. В слоте может содержаться не одно, а несколько значений, например, массивы, списки, фреймы и т.п. Так, в слоте БРАТ может содержаться список имен, если объект, описываемый данным фреймом, имеет нескольких братьев.

Значение слота может содержать перечень возможных значений, арифметическое выражение, фрагмент текста и т.д.

Пример фрейма РУКОВОДИТЕЛЬ

Имя слота	Значение слота	Тип значения слота
Имя	Иванов И.И.	Строка символов
Рождение	01.01.1965	Дата
Возраст	Age(дата, рождение)	Процедура
Специальность	Юрист	Строка символов
Адрес	Домашний адрес	фрейм

Совокупность данных предметной области может быть представлена множеством взаимосвязанных фреймов, образующих единую фреймовую систему, в которой объединяются декларативные и процедурные знания. Такая система имеет иерархическую структуру, в которой фреймы соединены друг с другом родовидовыми связями. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов. Например, фрейм УССУРИЙСКИЙ ТИГР наследует от фрейма ТИГР значение слота цвет - полосатый. Над фреймами можно совершать такие операции, как объединение и пересечение. При объединении фреймов в результирующем фрейме будут присутствовать все слоты, которые встречались в исходных фреймах. При пересечении фреймов в результирующем фрейме будут присутствовать только те слоты, которые имелись во всех исходных фреймах.

Фреймовые системы подразделяются на статические и динамические. Динамические допускают изменения фреймов в процессе решения задачи.

Имя фрейма служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Одни слоты являются системными и служат для выполнения специфических функций, например, слот-указатель родителя данного фрейма (IS-A), слот-указатель дочерних фреймов, слот для ввода имени пользователя, слот для ввода даты определения фрейма, слот для ввода даты изменения фрейма и т.п.

Имя слота. Оно должно быть уникальным в пределах фрейма. В качестве имени слота может выступать произвольный текст. Например, ИМЯ СЛОТА = Первый космонавт, а ЗНАЧЕНИЕ СЛОТА = Гагарин.

Имена системных слотов обычно зарезервированы, например, IS-A, HASSPART и т.п. Системные слоты служат для редактирования базы знаний и управления выводом во фреймовой системе.

Указатели наследования. Они показывают, какую информацию об атрибутах слотов фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. В конкретных системах указатели наследования могут быть организованы различными способами:

U(Unique) – значение слота не наследуется;

S(Same) – значение слота наследуется;

R(Range) – значение слота должны находится в пределах интервала значений, указанных в одноименном слоте родительского фрейма;

O(Override) – при отсутствии значений в текущем слоте оно наследуется из фрейма верхнего уровня. Однако, в случае определения значения текущего слота оно может быть уникальным. Этот тип указателя выполняет одновременно функции указателя U и S.

Указатель типа данных. Он показывает тип значения слота:

Frame – указатель на фрейм;

Real – вещественное число;

Integer – целое число;

Boolean – логический тип;

Text – фрагмент текста;

List – список;

Table – таблица;

Expression – выражение;

Lisp – связанная процедура и т.п.

Значение слота. Оно должно соответствовать типу данных и условию наследования.

Демоны. Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. Демоны автоматически запускаются при обращении к соответствующему слоту. Типы демонов связаны с условием запуска процедуры.

Демон IF-NEEDED запускается, если в момент обращения к слоту его значение не было установлено.

Демон IF-ADDED запускается при попытке изменения значения слота.

Демон IF-REMOVED запускается при попытке удаления значения слота.

Присоединенная процедура. В качестве значения слота может использоваться процедура, называемая служебной в языке ЛИСП или методом в языках объектно-ориентированного программирования. Она запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными знаниями в единую систему.

Во фреймах наследование происходит по связям АКО(IS-A). Слот АКО(IS-A) указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются значения аналогичных слотов. Например, в сети фреймов рис. 2.1 понятие «ученик» наследует свойства «ребенок» и

«человек», которые находятся на более высоком уровне иерархии.

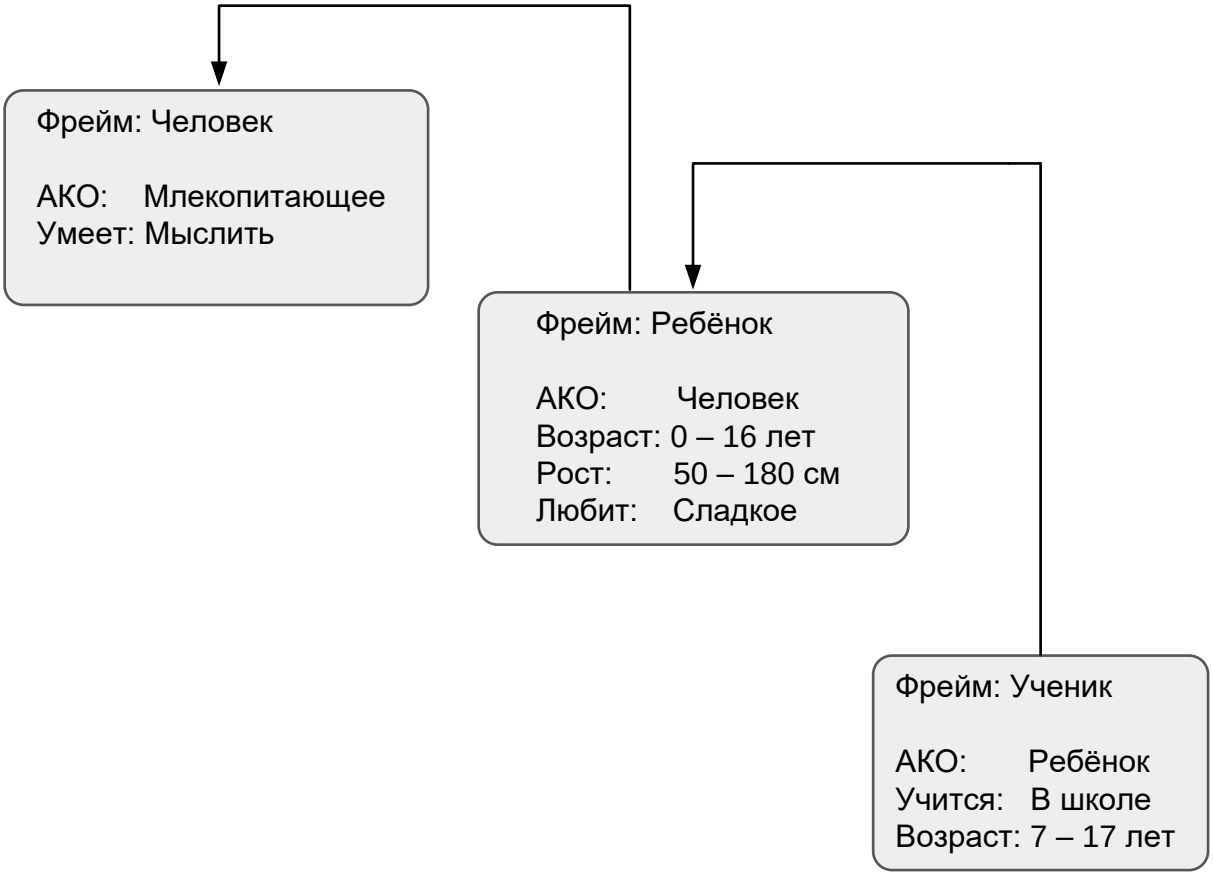


Рис. 2.1. Пример сети фреймов

Существует несколько способов получения слотом значений во фрейме-экземпляре:

- по умолчанию от фрейма – образца;
- через наследование свойств от фрейма, указанного в слоте АКО;
- по формуле, указанной в слоте;
- через присоединенную процедуру;
- через диалог с пользователем;
- из базы данных.

На вопрос «любит ли ученик сладкое» следует ответ «да», так как этим свойством обладают все дети, что указано во фрейме «ребенок». Наследование свойств может быть частичным, так как возраст для учеников

не наследуется из фрейма «ребенок», поскольку указан явно в своем собственном фрейме.

В качестве значения слота может быть имя другого фрейма, так образуется сеть фреймов.

Различают фреймы-образцы (фреймы-прототипы), хранящиеся в базе знаний и фреймы-экземпляры (фреймы-примеры), которые создаются для отображения реальных ситуаций на основе поступающих данных.

Можно выделить различные типы моделей фреймов:

- фреймы-структуры, применяемые для отображения объектов и понятий (заем, вексель);
- фреймы-роли (менеджер, кассир, клиент);
- фреймы-сценарии (банкротство, собрание акционеров и т.п.);
- фреймы – ситуации (тревога, авария, рабочий режим устройства и т.п.).

Основное преимущество модели фреймов состоит в том, что такая модель отражает концептуальную основу организации памяти человека и является наглядной и гибкой. Разработаны специальные языки представления знаний в сетях фреймов. Это FRL (Frame representation Language), KRL (Knowledge Representation Language, оболочка Kappa и другие [1]. Известны фрейм - ориентированные экспертные системы ANALYST, МОДИС, TRISTAN [1].

2.6. Семантическая модель представления знаний

Термин *семантическая* означает «смысловая». Семантика - это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т.е. наука, определяющая смысл знаков [1].

Семантическая сеть - это ориентированный граф, вершины которого - понятия, а дуги - отношения между ними.

В качестве понятий обычно выступают абстрактные или конкретные объекты, а *отношения* - это связи типа: «это» («АКО - A- Kind-Of», «is»), «имеет частью», «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

- класс – элемент класса (цветок-роза);
- свойство – значение (цветок–желтый);
- пример элемента класса (роза-чайная).

Семантическая сеть называется однородной, если содержит только один тип отношений, и неоднородной – в противном случае.

По типам отношений семантическая сеть подразделяется на бинарную (связывающую два объекта) и N-арную (для связи нескольких объектов).

Наиболее часто в семантических сетях используются следующие отношения:

- связи типа «часть-целое», «класс-подкласс», «элемент-множество»;
- функциональные связи, определяемые глаголами «производит», «влияет»;
- количественные (больше, меньше, равно);
- пространственные (далеко от, близко от, над, под и т.п.);
- временные (раньше, позже и т.п.);
- атрибутивные (иметь свойство, иметь значение);
- логические связи (И, ИЛИ, НЕ) и др.

Поиск решения в базе знаний, представленной в виде семантической сети, сводится к задаче поиска подграфа. В семантической сети (рис. 2.2) в качестве вершин выступают понятия «человек», «Иванов», «ВАЗ», «автомобиль», «вид транспорта», «двигатель».

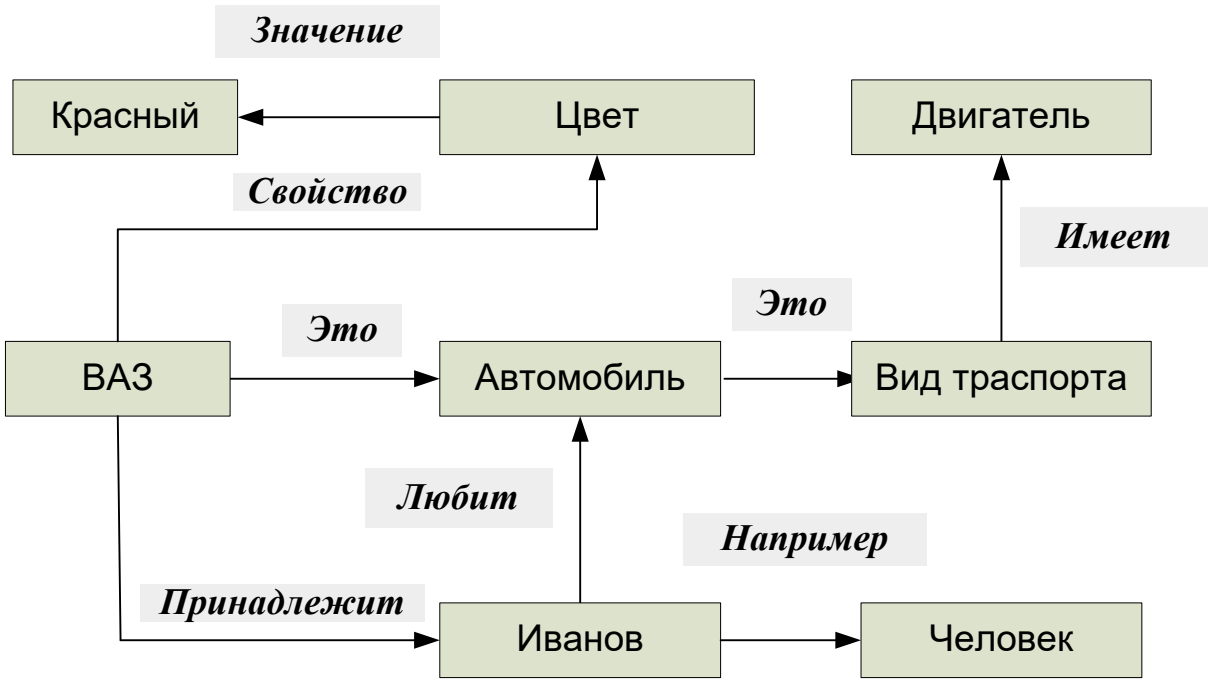


Рис. 2.2. Пример семантической сети

Данная модель была предложена американским психологом Куиллианом. Семантическая сеть соответствует современным представлениям об организации долговременной памяти человека. Здесь, как и во фреймах, декларативные и процедурные знания не разделены. К недостаткам следует отнести сложность поиска подграфа и вывода на семантической сети. Для реализации семантических сетей разработаны специальные языки (NET, SIMER+MIR и др.). Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний: PROSPECTOR, CASNET, TORUS [1].

2.7. Нейронная модель представления знаний

В последние годы бурно развиваются *нейронные экспертные системы*. В основе их построения лежит принцип обучения нейронной сети на известных примерах с последующим тестированием по любому входному вектору.

Нейронные сети могут выступать в качестве модели представления знаний. В первую очередь это обучающая выборка, которая представляет неявную базу знаний (до обучения нейросетевой экспертной системы).

Во-вторых, это синаптическая карта, сформированная по результатам выбора оптимальной архитектуры и обучения нейронной сети. Однако при дополнении обучающей выборки требуется дообучение или переобучение нейронной сети.

Дообучение сети не требует изменения архитектуры нейронной сети, а переобучение может привести к изменению архитектуры нейронной сети.

Достоинства: не требуется разрабатывать множество продукционных правил и реализовывать механизм разрешения конфликтов. Обучающее множество можно оптимизировать по количеству примеров.

Недостатки: отсутствует методика выбора оптимальной архитектуры нейронной сети, и методика определения оптимального размера обучающей выборки.

2.8. Представление знаний в виде нечетких правил

Математическая теория нечетких множеств (fuzzy sets) и нечеткая логика (fuzzy logic) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия были впервые предложены американским ученым Лотфи Заде (Lotfi Zadeh) в 1965 году. Основной причиной появления новой теории стало наличие нечетких и приближенных рассуждений при описании человеком процессов, систем, объектов [3,9,10]. Следует использовать общепринятую терминологию.

Нечеткое множество (fuzzy sets) – множество, элементы которого принадлежат ему в той или иной степени.

Нечеткая логика (fuzzy logic) – умозаключение с использованием нечетких множеств или нечетких правил.

Нечеткое правило (fuzzy rule) – условное высказывание вида "ЕСЛИ X есть A, ТО У есть B", где A и B нечеткие множества. Нечеткое правило образует связь между нечеткими множествами.

Нечеткая система (fuzzy system) – множество нечетких првил, преобразующих входные данные в выходные.

В нечеткой логике основополагающим является понятие *лингвистической переменной,* значениями которой являются не числа, а слова естественного языка, называемые *термами.*

Характеристикой нечеткого множества выступает функция принадлежности (Membership Function). MFc(x) – степень принадлежности значения «x» к нечеткому множеству C, представляющая собой обобщение понятия характеристической функции обычного множества. Тогда нечетким множеством C называется множество упорядоченных пар вида C={MFc(x)/x}, MFc(x) принадлежит интервалу от 0 до 1. Значение MFc(x)=0 означает отсутствие принадлежности к множеству, 1 – полную принадлежность.

Задание лингвистической переменной можно осуществить в дискретной или непрерывной форме.

Дискретная форма:

A(x)={M(x1)/x1, M(x2)/x2, …, M(xn)/xn}.

Непрерывная форма в виде функциональной зависимости.

$M_A(x)=\exp(-(x-3)/0,2)^2$.

Пример. Для множества чисел от x1=7, x2=8, …, x7=13 степень принадлежности их к числам, близким к 10, выглядит таким образом:

A(x)={0,1/7; 0.3/8; 0.8/9; 1.0/10; 0.8/11; 0.3/12; 0.1/13}.

Примером нечеткого множества может служить формализация неточного определения ГОРЯЧИЙ ЧАЙ. В качестве x (область рассуждений) будет выступать шкала температуры в градусах Цельсия. Очевидно, что она будет изменяется от 0 до 100 градусов. Нечеткое множество для понятия ГОРЯЧИЙ ЧАЙ может выглядеть следующим образом:

C={0/0; 0/10; 0/20; 0,15/30; 0,30/40; 0,60/50; 0,80/60; 0,90/70; 1/80; 1/90; 1/100}.

Например, чай с температурой 60 градусов Цельсия, принадлежит к множеству ГОРЯЧИЙ ЧАЙ со степенью принадлежности 0,80. Для одного человека чай с температурой 60 градусов Цельсия может оказаться горячим, для другого – не слишком горячим. Именно в этом и проявляется нечеткость задания соответствующего множества.

Нечеткая переменная описывается набором (N, X, A), где N – это название переменной, X - универсальное множество (область рассуждений), A – нечеткое множество на X.

Значениями лингвистической переменной могут быть нечеткие переменные, т.е. лингвистическая переменная находится на более высоком уровне, чем нечеткая переменная.

Каждая *лингвистическая переменная* состоит из следующих компонентов:

- наименование переменной;
- множество своих значений, которое также называется базовым терм-множеством T. Элементы базового терм-множества представляют собой названия нечетких переменных;
- универсальное множество X;
- синтаксические правила G, по которым генерируются новые термы с применением слов естественного или формального языка;

– семантические правила P, которые каждому значению лингвистической переменной ставят в соответствие нечеткое подмножество множества X.

Например, *нечеткое понятие* как ЦЕНА АКЦИИ представляет наименование лингвистической переменной. Для нее базовое терм-множество будет состоять из трех нечетких переменных: НИЗКАЯ, УМЕРЕННАЯ, ВЫСОКАЯ. Область рассуждений определяется в виде X=[100;200] единиц.

Для каждого лингвистического терма из базового терм-множества T должна быть построена функция принадлежности. Количество термов в лингвистической переменной обычно не превышает семи.

Например, в случае управления мобильным роботом можно ввести две лингвистические переменные: ДИСТАНЦИЯ (расстояние до помехи) и НАПРАВЛЕНИЕ (угол между продольной осью робота и направлением на помеху).

Рассмотрим *лингвистическую переменную* ДИСТАНЦИЯ. Значениями ее можно определить термы ДАЛЕКО, СРЕДНЯЯ, БЛИЗКО и ОЧЕНЬ БЛИЗКО. Для физической реализации лингвистической переменной необходимо определить точные физические значения термов этой переменной. Пусть переменная ДИСТАНЦИЯ может принимать любое значение из диапазона от нуля до бесконечности. Согласно положениям теории нечетких множеств, каждому значению расстояния из указанного диапазона может быть поставлено в соответствие некоторое число от нуля до единицы, которое определяет степень принадлежности данного физического расстояния (допустим 40 см) к тому или иному терму лингвистической переменной ДИСТАНЦИЯ.

Степень принадлежности определяется так называемой функцией принадлежности M(d), где d - расстояние до помехи. В нашем примере расстоянию 40 см можно задать степень принадлежности к терму ОЧЕНЬ

БЛИЗКО, равную 0,7, а к терму БЛИЗКО – 0,3 (рис.2.3). Конкретное определение степени принадлежности может проходить только при работе с экспертами.

Переменной НАПРАВЛЕНИЕ, которая может принимать значения в диапазоне от 0 до 360 градусов, зададим термы ЛЕВОЕ, ПРЯМО И ПРАВОЕ.

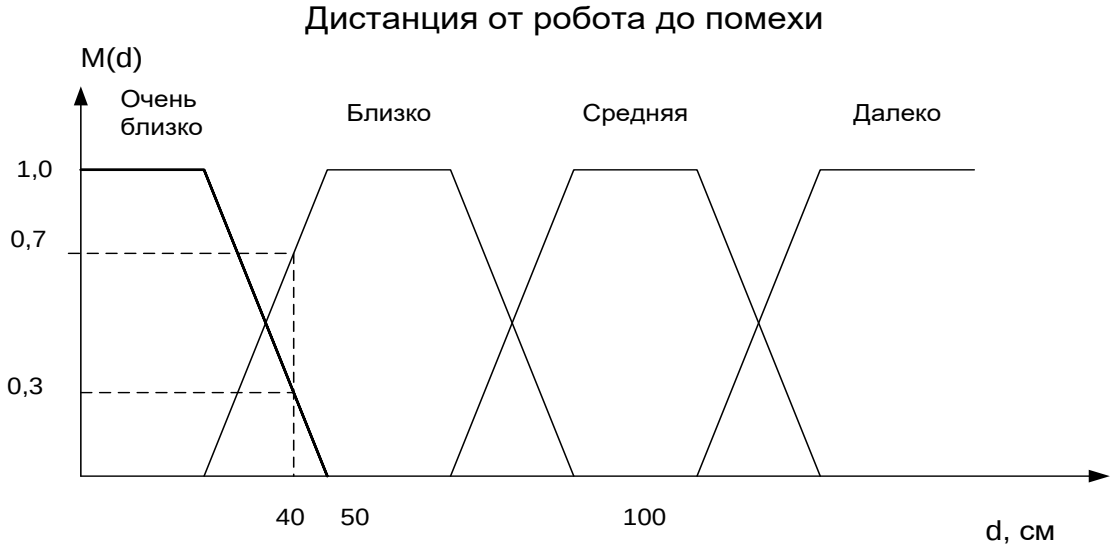
Теперь необходимо задать выходные переменные. В рассматриваемом примере достаточно одной, которая будет называться РУЛЕВОЙ УГОЛ. Она может содержать термы: РЕЗКО ВЛЕВО, ВЛЕВО, ПРЯМО, ВПРАВО, РЕЗКО ВПРАВО. Связь между входом и выходом запоминается в таблице нечетких правил (рис.2.4).

Каждая запись, представленная на рис.2.4, соответствует своему нечеткому правилу: «Если ДИСТАНЦИЯ БЛИЗКО и НАПРАВЛЕНИЕ ПРАВОЕ, тогда РУЛЕВОЙ УГОЛ РЕЗКО ВЛЕВО».

Таким образом, мобильный робот с нечеткой логикой будет работать по следующему принципу: данные с сенсоров о расстоянии до помехи и направлении на нее будут фаззифицированы, обработаны согласно табличным правилам, дефаззифицированы, и полученные данные в виде управляющих сигналов поступят на привод робота.

Все системы с нечеткой логикой функционируют по одному принципу: показания измерительных приборов фаззифицируются (переводятся в нечеткий формат), обрабатываются (см. ниже), дефаззифицируются и в виде привычных сигналов подаются на исполнительные устройства.

Фаззификация – это сопоставление множества значений x ее функции принадлежности M(x), т.е. перевод значений x в нечеткий формат (пример с термином молодой). Дефаззификация представляет собой процесс, обратный фаззификации [3,9,10].



Лингвистическая переменная «ДИСТАНЦИЯ» включает 4 терма
«ДИСТАНЦИЯ»= {очень близко, близко, средняя, далеко }

Рис.2.3. Лингвистическая переменная «ДИСТАНЦИЯ»

Направление	Дистанция			
	Очень близко	**Близко**	Средняя	Далеко
Правое	Резко влево	**Резко влево**	Влево	Прямо
Прямо	Резко влево	Влево	Влево	Прямо
Левое	Резко вправо	Резко вправо	Вправо	Прямо

Пример нечеткого правила «ЕСЛИ дистанция **БЛИЗКО**
И направление **ПРАВОЕ**, ТО рулевой угол **РЕЗКО ВЛЕВО»**

Рис. 2.4. База нечетких правил

В нечеткой логике есть возможность строить выражения, на основе которых можно получать новые значения нечетких (следовательно, и лингвистических) переменных.

Применение правил нечеткого вывода позволяет получить описание предметной области на основе имеющихся у эксперта знаниях о ней.

Благодаря операциям фаззификации и дефаззификации имеется возможность обработки данных, общая схема которой представляется так:

- по имеющемуся набору данных, с помощью оператора фаззификации провести перевод этих данных в нечеткий формат;
- пользуясь имеющейся базой нечетких правил, преобразовать нечеткие переменные;
- получившийся набор нечетких переменных подвергнуть дефаззификации и получить четкое значение.

Часть выходных данных можно передать через обратную связь на вход, если того требует алгоритм управления.

Проведение таких преобразований называется нечетким логическим выводом. Для реализации ЭС на базе нечетких правил разработано множество алгоритмов нечеткого вывода. Например, правила нечеткого вывода заданы следующим образом:

П1: если x есть A, то w есть D,
П2: если y есть B, то w есть E, (2.1)
П3: если z есть C, то w есть F,

где x, y, z – имена входных переменных (четкого формата);
w – имя переменной вывода;
A, B, C, D, E, F – заданные функции принадлежности.

Иллюстрация к алгоритму нечеткого вывода представлена на рис.2.5.

Пример реализации алгоритма Мамдани с правилами П1 и П2:

П1: если x есть A1 и y есть B1, то z есть C1,
П2: если x есть A2 и y есть B2, то z есть C2, (2.2)

где x, y – имена входных переменных (четкого формата);
z – имя переменной вывода;
A1, B1, C1, A2, B2, C2 – заданные функции принадлежности.

Далее следует этап, называемый «введение нечеткости».

Находятся степени истинности для предпосылок каждого правила:

$$A_1(x_0), A_2(x_0), B_1(y_0), B_2(y_0), \tag{2.3}$$

где x_0, y_0 – имена входных переменных (четкого формата).

Находятся уровни отсечения для предпосылок каждого из правил.

$$\begin{aligned} a_1 &= A_1(x_0) \wedge B_1(y_0) \\ a_2 &= A_2(x_0) \wedge B_2(y_0), \end{aligned} \tag{2.4}$$

где a1, a2 – уровни отсечения;

$\wedge$ - оператор минимума.

Производится объединение усеченных множеств

$$\mu_{\Sigma}(z) = (a_1 \wedge C_1(z)) \vee (a_2 \wedge C_2(z)), \tag{2.5}$$

где C(z) – функция принадлежности для элемента z.

Для нахождения значения z_0 необходимо провести дефаззификацию, например, центроидным методом [26].

Дефаззификация – приведение к четкости. Существует несколько популярных методов [3,9,10], некоторые из которых приведены ниже.

Центроидный метод. Для дискретного варианта представления множеств:

$$Z_0 = \frac{\sum_{i=1}^{n} \mu_i z_i}{\sum_{i=1}^{n} \mu_i}. \tag{2.6}$$

где z – элемент эталонного множества;

z_i – элемент эталонного множества;

μ_i – функция принадлежности для элемента z_i;

Z_0 – четкое значение.

Метод первого максимума (first-of-maxima). Четкая величина находится как наименьшее значение, при котором достигается максимум функции принадлежности.

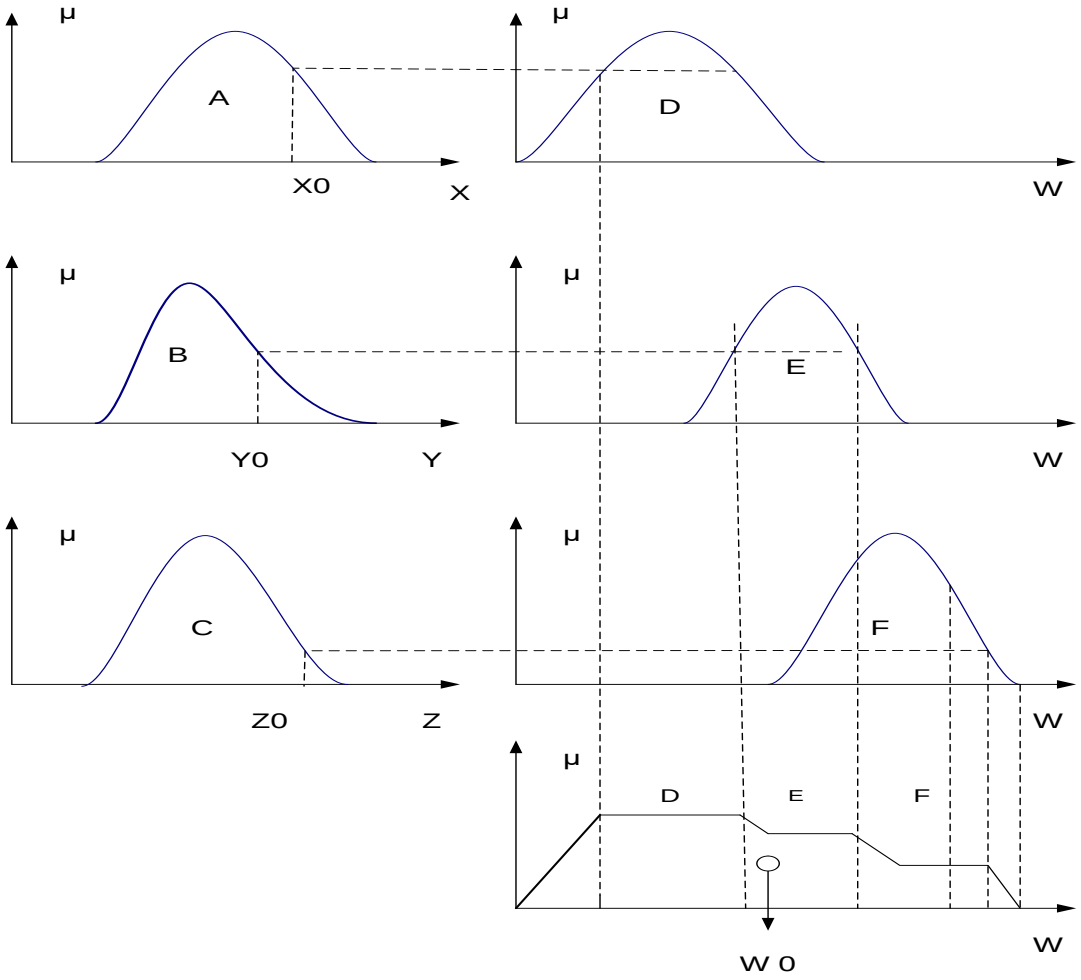


Рис. 2.5. Процедура нечеткого вывода

Метод среднего максимума. Для дискретного варианта представления множеств:

$$Z_0 = \frac{1}{n}\sum_{i=1}^{n} z_i \ . \qquad (2.7)$$

где z_i – элемент эталонного множества;

n – число элементов – максимумов одного уровня

Z_0 – четкое значение.

Кроме описанных алгоритмов, можно отметить такие алгоритмы как Tsukamoto, Sugeno, Larsen [10, 11, 12].

Контрольные вопросы

1. Отличие знаний от данных.
2. Классификация знаний.
3. Примеры декларативных, процедурных, глубинных и поверхностных знаний.
4. Понятие интенсионала и экстенсионала.
5. Примеры продукционных правил.
6. Понятие фрейма, характеристика его слотов и пример фрейма.
7. Понятие семантической сети и пример.
8. Способы представления знаний в нейронной сети.
9. Пример нечетких правил.
10. Пример лингвистической переменной.
11. Понятие фаззификсации и дефаззификации.
12. Основные функции фаззификации.
13. Способы дефаззификации.
14. Метод нечеткого вывода Мамдани.

РАЗРАБОТКА ИНСТРУМЕНТАЛЬНОЙ ПРОДУКЦИОННОЙ ЭКСПЕРТНОЙ СИСТЕМЫ

3.1. Концепция построения инструментальной экспертной системы

Вопросы проектирования экспертных систем достаточно полно освещены в литературе [1, 5, 9] и на сайте www.mari-el.ru/mmlab/home/AI/.

При создании экспертных систем одним из основных критериев является трудоемкость создания экспертных систем. В условиях сжатых сроков разработки и ограничения на ресурсы проектирование ЭС на базе инструментальной системы является наиболее предпочтительным вариантом решения проблемы.

Инструментальная экспертная система предназначена для создания проблемно-ориентированных или предметно-ориентированных экспертных систем и демонстрации возможностей, которые предоставляют продукционные правила при их создании.

Часто к инструментальным экспертным системам предъявляют требование по обеспечению работы в условиях неопределённости и неполноты информации. Сведения о поставленной задаче могут быть неполными, и отношения между объектами предметной области могут быть приближёнными. Например, может не быть полной уверенности в наличии у пациента некоторого симптома или в том, что данные, полученные при измерении, верны. В таких случаях необходимы рассуждения с использованием вероятностного подхода.

Психологические исследования процессов принятия решений человеком показали, что, рассуждая, человек использует правила, аналогичные продукциям, которые называются «условие→действие».

При использовании продукционной модели база знаний состоит из набора правил. Основным модулем любой ЭС является модуль машины

логического вывода, который управляет перебором правил, принятием решения и объяснением хода рассуждения [1].

Машина логического вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр правил из базы знаний, во-вторых, применения правил.

Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила оказывается недостаточно данных.

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой программу и включает в себя два компонента: первый реализует собственно вывод, другой управляет этим процессом.

Действие компонента вывода основано на применении правила, называемого modus ponens: «*если известно, что истинно утверждение A и существует правило вида «ЕСЛИ A, ТО B», тогда утверждение B также истинно».*

Правила срабатывают, когда находятся истинные факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение.

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

Управляющий компонент определяет порядок применения правил и выполняет четыре функции:

– сопоставление - образец правила сопоставляется с имеющимися фактами;

- выбор - если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта);
- срабатывание - если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает;
- действие - рабочая память подвергается изменению путём добавления в неё 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 продукций работает циклически [1]. В каждом цикле он просматривает все правила, чтобы выявить те посылки, которые совпадают с известными и истинными на данный момент фактами. После выбора правило срабатывает, его заключение заносится в рабочую память, а затем цикл повторяется сначала.

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то такая ситуация называется конфликтной. Интерпретатор выполняет разрешение конфликтов путём выбора по определённому критерию единственного правила, которое срабатывает в данном цикле. Цикл работы интерпретатора схематически представлен на рис. 3.1.

Информация из рабочей памяти последовательно сопоставляется с посылками правил для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое конфликтное множество. Для разрешения конфликта в интерпретаторе разработчиком предусматривается блок разрешения конфликтов, который выбирает единственное правило, после чего оно 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 название какого-нибудь действия, то оно

выполняется. Способы разрешения конфликтов достаточно подробно рассмотрены в [5]. По мнению автора книги, [5], в настоящее время не существует эффективного способа разрешения конфликтов и выбор способа полностью определяется предметной областью, для которой разрабатывается ЭС.

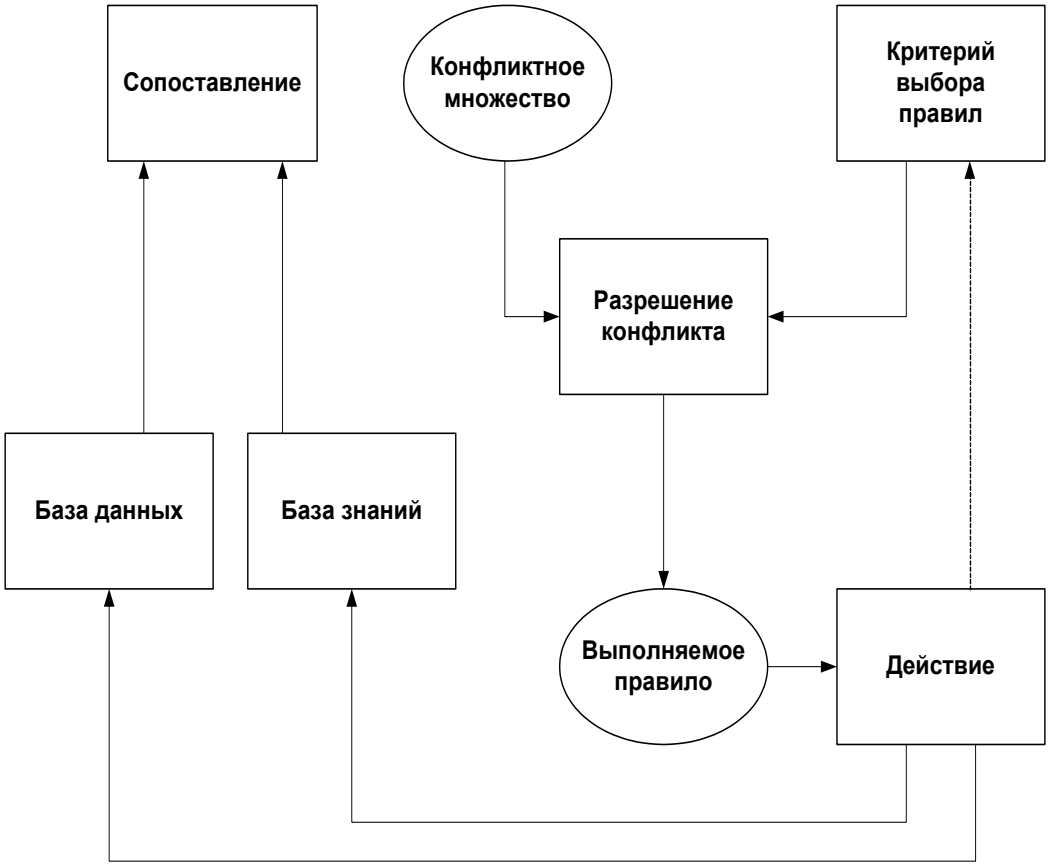


Рис. 3.1. Цикл работы интерпретатора машины логического вывода

При использовании теории приближенных рассуждений сравнительно корректно решается проблема функционирования ЭС в условиях неопределенности [12] и отпадает необходимость реализации блока разрешения конфликтов.

Стратегии управления выводом - это порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно включаются в механизм вывода. Поэтому в

большинстве случаев инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию.

При разработке стратегии управления выводом важно определить два вопроса.

1. Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска - в прямом или обратном направлении;
2. Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора: в глубину или в ширину, по подзадачам или иначе.

Прямой и обратный вывод. При *обратном порядке вывода* вначале выдвигается некоторая гипотеза, а затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу. Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчинённой гипотезы. Вывод такого типа называется управляемым целями, или управляемым консеквентами.

Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

В системах с *прямым выводом* по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удаётся найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами.

Например, имеется фрагмент базы знаний из двух правил.

Правило 1 - ЕСЛИ «отдых - летом» И «человек - активный», ТО «ехать - в горы».

Правило 2 - ЕСЛИ «любит - солнце», ТО «отдых - летом».

Предположим, в систему введены истинные факты - «любит - солнце» И «человек - активный».

Прямой вывод - исходя из фактических данных позволяет получить рекомендацию.

Первый проход

Шаг 1. Попробуем правило 1, не работает (не хватает данных «отдых - летом»).

Шаг 2. Попробуем правило 2, работает, в базу поступает факт «отдых - летом».

Второй проход

Шаг 3. Попробуем правило 1, работает, активизируется цель «ехать - в горы», которая и выступает как совет, который дает экспертная система.

Обратный вывод позволяет подтвердить выбранную цель при помощи имеющихся правил и данных.

Первый проход

Шаг 1. Цель - «ехать - в горы»: попробуем правило 1 - не хватает данных «отдых - летом», они становятся новой целью и выполняется поиск правила, у которого цель находится в левой части.

Шаг 2. Цель - «отдых - летом»: правило 2 подтверждает цель и активирует её.

Второй проход

Шаг 3. Попробуем правило 1, подтверждается искомая цель.

Идея двунаправленного поиска основывается сразу на двух стратегиях – прямого поиска от корневой вершины и обратного от целевой вершины. Процесс поиска прекращается, когда оба эти процесса встречаются.

Принцип двунаправленного вывода отражён в методе деления дизъюнктов.

В системах, база знаний которых насчитывается сотни правил, желательным является использование стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вывода. К числу таких стратегий относятся: «поиск в глубину», «поиск в ширину», разбиение на подзадачи и альфа-бета алгоритм.

При *«поиске в глубину»* в качестве очередной подцели выбирается та, которая соответствует следующему, более детальному уровню описания задачи. Например, диагностирующая система, сделав на основе известных симптомов предложение о наличии определённого заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не опровергнет выдвинутую гипотезу.

При *«поиске в ширину»*, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдёт к симптомам следующего уровня детальности.

Разбиение на подзадачи подразумевает выделение подзадач, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Примером, подтверждающим эффективность разбиения на подзадачи, является поиск неисправностей в компьютере: сначала выявляется отказавшаяся подсистема (питание, память и т.д.), что значительно сужает пространство поиска. Если удаётся правильно понять сущность задачи и оптимально разбить её на систему иерархически связанных целей, то можно добиться того, что путь к её решению в пространстве поиска будет минимальным.

Альфа-бета алгоритм позволяет уменьшить пространство состояний путём удаления ветвей, неперспективных для успешного поиска. Поэтому просматриваются только те вершины, в которые можно попасть в

результате следующего шага, после чего неперспективные направления исключаются.

3.2. Основы теории приближенных рассуждений

Существуют следующие проблемы, которые необходимо учитывать в понятии неопределенности систем логического вывода [1].

Как количественно выразить степень определенности при установлении истинности (или ложности) некоторой части данных?

Как отразить степень поддержки заключения конкретной посылкой?

Как использовать совместно несколько посылок, влияющих на заключение?

При разработке ЭС могут использоваться методы точного вероятностного и приближенного рассуждений [1, 5, 6].

Использование рассуждений на основе вероятностей становится все более трудным и неудобным. По этой причине многие ЭС применяют специальные методы приближенных рассуждений. Именно такой механизм использован в экспертных системах EMYCIN, FUZZYNET, которые демонстрируют эффективность многоступенчатых приближенных рассуждений. Импликация с одной посылкой имеет вид «ЕСЛИ (е), ТО (с)».

Обычное правило комбинирования, позволяющее вычислить коэффициент определенности заключения в случае, когда известен коэффициент определенности посылки, лежащей в его основе, и связи в импликации, записывается следующим образом:

ct (заключение) = ct (посылка) · ct (импликация).

Если в примере истинность посылки определена с уверенностью 0,8, а импликация выполняется в большинстве случаев, но не всегда (например, с коэффициентом определенности 0,9), тогда коэффициент определенности заключения вычисляется следующим образом:

ct (заключение) = 0,8 · 0,9 = 0,72.

Логические комбинации посылок в одном правиле. Основной вычислительный прием, который можно использовать для нахождения коэффициента определенности заключения, сводится к следующему:

ct (заключение) = ct (посылка) · ct (импликация).

Прежде всего нужно суметь оценить коэффициенты определенности посылок. Посылкой считаются все логические выражения в правиле между словами «ЕСЛИ» и «ТО». Исключение составляет простая импликация, в которой выражение состоит из атомарных посылок, каждая из которых имеет свой коэффициент определенности.

Посылки могут быть связаны между собой логическими операциями, например, ЕСЛИ (e1 ИЛИ (e2 И e3)), ТО (c) или

ЕСЛИ (e1 И e2 И ((НЕ e3) ИЛИ e4)) , ТО (c).

Очевидно, требуется некоторый способ оценки коэффициентов определенности этих сложных форм в понятиях их отдельных компонент. Подход заключается в том, чтобы отбросить все сложные выражения и считать все правила простыми. Есть несколько тривиальных процедур для сведения коэффициентов определенности простых логических комбинаций в одно число.

Простой логической комбинацией является конъюнкция (И) между двумя элементарными свидетельствами. Импликация выглядит так:

ЕСЛИ (e1 И e2) , ТО (c)

Коэффициент определенности посылки равен коэффициенту определенности наименее надежной из посылок, т.е.

ct (e1 И e2) = min [ct (e1), ct (e2)].

Другой простой формой является правило, в котором используется

дизъюнкция (ИЛИ), связывающая две части свидетельств: ЕСЛИ (e1 ИЛИ e2) , ТО (c) .

Общее правило комбинирования, по которому вычисляется коэффициент определенности посылки, заключается в том, что

коэффициент определенности дизъюнкции равен коэффициенту определенности ее сильнейшей части

ct (e1 ИЛИ e2) = max [ct (e1), ct (e2)].

Хотя правила иногда и записываются с помощью дизъюнкции, но если есть выбор, то принято разбивать дизъюнкцию на две части, например:

ЕСЛИ (e1) , ТО (c),

ЕСЛИ (e2) , ТО (c).

Использование двух правил вместо дизъюнкции требует механизм, определяющий коэффициент определенности заключения при поддержке этих правил. Выбор способа представления правил определяется экспертом.

Поддержка одного заключения множеством правил. Например, используются два правила и оба они поддерживают одно и то же заключение:

правило 1: ЕСЛИ (e1), ТО (c) ct (заключение) = 0,9;

правило 2: ЕСЛИ (e2), ТО (c) ct (заключение) = 0,8.

Допустим, обе посылки верны, тогда можно вычислить вероятность заключения для каждого правила по отдельности.

Предположим, что переменная ctotal представляет общий коэффициент определенности заключения, полученный использованием всех поддерживающих его правил. Можно предложить много различных комбинаций процедур. Рассмотрим простой и эффективный механизм, приведенный в [12]: ctotal = коэффициент определенности из правила 1 + + коэффициент определенности из правила 2 минус произведение (коэффициента определенности из правила 1) и· (коэффициента определенности из правила 2).

Здесь два коэффициента определенности преобразуются в один коэффициент, который всегда меньше единицы. Подставив числа, заданные в примере, получим

ctotal = 0,9 + 0,8 – (0,9) · (0,8) = 0,98.

Рассмотренный принцип можно распространить на случай из нескольких правил, поддерживающих одно заключение, где для каждого

правила существует своя вероятность. Например, есть три правила со следующими коэффициентами определенности.

Правило 1: ЕСЛИ (e1), ТО (c) ct (заключение) = ct1.

Правило 2: ЕСЛИ (e2), ТО (c) ct (заключение) = ct2.

Правило 3: ЕСЛИ (e3), ТО (c) ct (заключение) = ct3.

Совокупный коэффициент определенности заключения с учетом всей возможной поддержки может быть вычислен следующим образом:

ctotal = ct1 + ct2 +ct3 - ct1 · ct2 - ct1 · ct3 - ct1 · ct3 + ct1 · ct2 · ct3.

Суть заключается в формировании произведений из базовых коэффициентов определенностей правил и в сложении и вычитании этих произведений соответствующим образом. Каждая двойная комбинация коэффициентов определенностей вычитается, тройная - прибавляется, четвертная - вычитается и т.п. до тех пор, пока все правила не будут использованы совместно.

Несколько правил, используемых последовательно. Механизм дополнения - это другой способ вычисления коэффициента определенности заключения, поддерживаемого несколькими правилами импликации. Он используется в том случае, когда сведения о разрешенных к применению правилах поступают последовательно, а не одновременно. Например, если система задает пользователю вопросы, то использование новых правил будет происходить по очереди.

Например, известно, что заключение поддерживается двумя правилами со следующими коэффициентами определенности.

Правило 1: ЕСЛИ (e1), ТО (c) ct (заключение)= ct1.

Правило 2: ЕСЛИ (e2), ТО (c) ct (заключение)= ct2.

При применении двух правил совокупный коэффициент определенности ctotal = ct1 + ct2 - ct1 · ct2.

Теперь предположим, что появилось третье правило, поддерживающее то же заключение:

правило 3: ЕСЛИ (e3), ТО (c) ct (заключение) = ct3.

Если все, что получено из предыдущего рассмотрения, входит в переменную ctotal и считается, что ct3 может войти в рассуждения на общих основаниях, то можно использовать стратегию дополнения для формирования измененной оценки коэффициента определенности заключения:

cnewtotal = ct3 + ctotal - ct3 · ctotal.

Перемножив все компоненты этой формулы, получается следующий результат:

cnewtotal = ct3 + (ct1 + ct2 - ct1 · ct2) - ct3 · (ct1 + ct2 - ct1 · ct2) =

= ct1 + ct2 + ct3 - ct1 · ct2 - ct1 · ct3 - ct2 · ct3 + ct1 · ct2 · ct3.

Очевидно, что это в точности тот же вывод, который был сделан ранее, комбинируя свидетельства, полученные одновременно.

Данный вывод, если его обобщить, имеет два важных приложения:

- при использовании механизма дополнения порядок поступления правил, поддерживающих заключение, не имеет значения;
- можно объединять коэффициенты определенности из поддерживающих импликаций последовательно по мере их поступления или сохранять информацию, а затем использовать ее всю сразу - результат от этого не меняется.

Практически сеть рассуждений меняется, как только поступают новые сведения. Поэтому сохранять нужно лишь совокупный коэффициент определенности для каждого заключения, что обеспечивает наиболее экономный способ поддержки информационного обеспечения ЭС.

Биполярные схемы для коэффициентов определенности. Прототипом систем, основанных на приближенных рассуждениях, являются MYCIN и ее прямой потомок EMYCIN. Эти системы используют механизм объединения коэффициентов определенностей, который был рассмотрен выше. Коэффициент определенности является грубым приближением к

вероятности. В EMYCIN [12] используется интервал от минус 1 до 1, так что это не может быть вероятностью.

Границы интервала обозначают следующее: «1» - система в чем-то полностью определена, «0» - у системы нет знаний об обсуждаемой величине, «минус 1» - высказанная гипотетическая посылка или заключение абсолютно неверно. Промежуточные величины отражают степень доверия или недоверия к указанным ситуациям. Все описанные процедуры рассуждений применимы для коэффициентов определенности, задаваемых в этих более широких границах. Однако нужно учесть, что, когда с помощью правил вы находится максимум или минимум двух величин, нужно помнить про знаки. Например, значение 0.1 должно рассматриваться как более крупная величина, чем минус 0.2.

Полная реализация идеи биполярных коэффициентов определенности требует сделать два обобщения для развиваемого нами вычислительного механизма. Во-первых, отсутствует навык работы с отрицанием атомарных посылок. Например, в посылке имеется частица «НЕ»: ЕСЛИ (e1 И (НЕ e2)), ТО (c) .

В этом случае нужно только считать (не e2) атомарным утверждением. Можно дать ему новое имя, например, e3, но какой коэффициент определенности следует приписать этому новому свидетельству? Обычно коэффициент определенности задан для e2. Для вычисления же коэффициента определенности (НЕ e2) достаточно просто поменять знак: ct (НЕ e) = - ct (e).

Этот факт является следствием применения биполярной меры определенности. В любом случае коэффициент определенности для отрицания посылки всегда может быть легко найден, а потом использован в различных манипуляциях.

Особый интерес представляет процедура получения композиции коэффициентов определенностей в условиях поддержки двумя правилами одного и того же заключения.

Необходимо сделать следующее: если оба коэффициента определенности положительны:

$$ctotal = ct1 + ct2 - ct1 \cdot ct2, \quad (3.1)$$

если оба коэффициента определенности отрицательны:

$$ctotal = ct1 + ct2 + ct1 \cdot ct2. \quad (3.2)$$

Когда отрицателен только один из коэффициентов, то

$$ctotal = \frac{ct1 + ct2}{1 - \min[|ct1|, |ct2|]}, \quad (3.3)$$

где $|ct1|, |ct2|$ - модули коэффициентов уверенности.

В том случае, если один коэффициент определенности равен 1, а другой минус 1, то ctotal = 0.

Когда два правила с небольшими коэффициентами определенности поддерживают одно заключение, коэффициент определенности заключения возрастает. Если же знаки не совпадают, то результат определяется «сильнейшим» коэффициентом, но влияние его несколько ослабляется.

Применение биполярных коэффициентов определенности может привести к нереальным результатам, если правила сформулированы неточно. Работая с одним правилом вывода, следует не забывать, что всегда используется соотношение ct (заключение) = ct (посылка) · ct (импликация).

Все правила попадают в одну из этих двух очень важных категорий. Правила первой категории будем называть *обратимыми*. Одной из характеристик такого правила является его применимость к любому значению коэффициента определенности, которое может быть связано с посылкой. Правила второй категории считаются *необратимыми*. Эти правила «работают» только при положительных значениях посылки. Если же ее значение отрицательно, правило применять нельзя (оно не имеет здравого смысла). В этом случае необратимое правило с отрицательным коэффициентом уверенности отбрасывается.

При создании правил всегда следует проверять их на обратимость, имитируя отрицание посылки и заключения, и определяя сохраняет ли правило здравый смысл.

Многоступенчатые рассуждения. До сих пор окончательное заключение отделялось от посылки одним шагом рассуждений. Более типичной является другая ситуация - сеть, в которой окончательные рассуждения отдельны от базы посылок большим числом промежуточных шагов. Такие рассуждения называются многоступенчатыми.

Можно привести следующий пример многоступенчатых рассуждений на базе правил диагностики заболевания. Число в правой части каждого правила указывает коэффициент определенности для конкретной импликации.

Если у вас грипп, и вы находитесь в уязвимом возрасте, то
вызовите врача ct (импликация) = 0,9

Если у вас острый фарингит, то
вызовите врача ct (имплдикация) = 1,0

Если у вас простуда, то
ложитесь в постель и примите аспирин ct (имплдикация) = 0,4

Если у вас грипп, и вы не находитесь в уязвимом возрасте, то
ложитесь в постель и примите аспирин ct (импликация) = 0,4

Если у вас лихорадка и болят мышцы, то
это грипп ct (импликация) = 0,7

Если у вас насморк, мышечные боли и нет лихорадки, то
это простуда ct (импликация) = 0,7

Если у вас в горле нарывы и есть лихорадка, то
это острый фарингит ct (импликация) = 0,8

Если вам меньше 8 или больше 60 лет, то
вы находитесь в уязвимом возрасте ct (импликация) = 0,7

Теперь вы можете просмотреть правила и в зависимости от конкретных симптомов заболевания решить, обратиться ли к врачу или достаточно лечь в постель и принять аспирин. Вы даже можете использовать комбинацию изученных правил для определения коэффициента определенности, соответствующей каждому из возможных результатов, а потом выбрать тот, который имеет наибольший коэффициент определенности. Однако данная форма представления правил удобна для компьютера, но не для человека.

Для обсуждения многоступенчатого рассуждения правила удобно преобразовывать в другую форму, позволяющую более отчетливо представить соответствующие коэффициенты. Любая система может быть отображена графически. Она называется сетью вывода и имеет вид графика с указанными связями между правилами. На графике также отчетливо видны все возможные поддерживающие структуры промежуточных рассуждений, находящиеся ниже любого более высокого уровня заключения. Такая сеть логического вывода строится, чтобы придать правилам конкретную форму (рис. 3.2).

Сеть показывает возможности многоступенчатых рассуждении в задаче в более удобном виде, чем просто список утверждений. Здесь сделана попытка представить явным образом все шаги рассуждений для некоторой гипотетической ситуации, когда пациент имеет какое-то заболевание, родственное гриппу, и хочет получить рекомендации.

В диаграммах подобного типа используются некоторые стандартные приемы, которые надо знать, чтобы уметь их прочесть (рис. 3.3). Сеть вывода и множество взаимосвязанных импликаций - это одно и то же. Обе формы содержат одинаковый объем информации.

В правилах, составляющих сеть вывода, могут быть и позитивные, и негативные утверждения. В рассмотренном выше примере встретились две фразы: «есть лихорадка» и «нет лихорадки».

Поскольку здесь речь идет об одном и том же, эти фразы удобно зафиксировать в одном узле сети вывода. Для правила, где фраза появляется в негативной форме, связь отмечается перечеркивающей полосой, проходящей через узел, отображающий лихорадку. Там, где она появляется в позитивной форме, связь имеет обычный вид.

Биполярные схемы позволяют получить коэффициент определенности негативной формы путем смены знака на противоположный.

При изображении связки «И» используется сплошная дуга (например, для вершин c1, c2, c4), а при изображении связки «ИЛИ» -штриховая дуга (например, для вершины c3). Связка «НЕ» изображается короткой чертой на выбранной дуге (например, e3-c1).

Пример, иллюстрирующий распространение коэффициентов определенности в сети, приведен на рис. 3.2. Коэффициент определенности отмечен справа от каждого узла. Отдельные первоначальные посылки, расположенные в нижней части дерева, показывают коэффициенты определенностей, которые были получены при задании необходимых вопросов и при получении данных из внешнего мира. В исходном состоянии все внутренние узлы имеют коэффициенты определенности, равные нулю, так как рассуждения пока не проводились.

Под каждым внутренним узлом стоит число, отражающее коэффициент определенности импликации, поддерживающей конкретный узел. Рядом с коэффициентом определенности импликации записывается признак rev (для обратимых правил) или признак nrev (для необратимых правил), что обозначает, будет импликация использоваться как обратимое или как необратимое правило. Обратимое правило можно применять всегда, а необратимое нужно удалить из сети, если коэффициент определенности посылки для этого правила становится отрицательным. Вычисление коэффициента определенности заключения может потребовать выполнения нескольких шагов: могут добавляться «И» «ИЛИ», «НЕ». В каждом

конкретном случае, пока не будет закончена вся эта предварительная работа, нельзя с уверенностью сказать, применимо ли правило.

Иногда правило включает отрицание некоторой посылки или заключения. На диаграммах сети вывода определенности всегда показаны для посылок или заключения до применения отрицания.

Пример расчета коэффициентов уверенности в сети логического вывода, представлен на рис. 3.3. Предлагаются следующие правила:

ЕСЛИ (e1), ТО (c1) ct (импликация) = 0,8 (nrev) ,

ЕСЛИ (e2), ТО (c2) ct (импликация) = 0,9 (rev) ,

ЕСЛИ (e3), ТО (c2) ct (импликация) = 0,7 (rev) ,

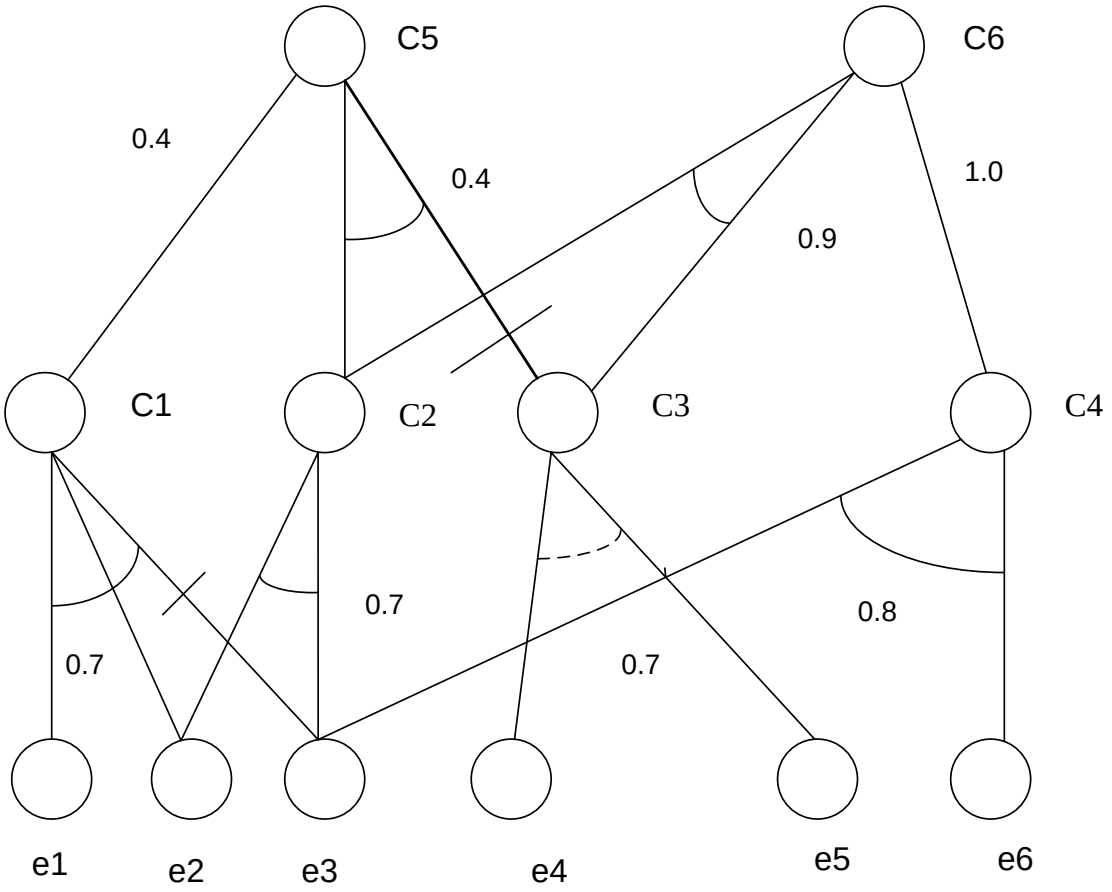
ЕСЛИ (e4), ТО (c3) ct (импликация) = 0,6 (nrev),

ЕСЛИ (НЕ e5), ТО (c3) ct (импликация) = 0,5 (nrev) ,

ЕСЛИ (c2 И c3), ТО (c4) ct (импликация) = 0,9 (rev),

ЕСЛИ (c1 ИЛИ c4), ТО (c5) ct (импликация) = 0,8 (nrev) .

Следует начать с основания и идти вверх по дереву, чтобы оценить, что же произошло.



e1 - насморк; e2 - мышечные боли; e3 - лихорадка; e4 - возраст менее 8 лет; e5 - возраст более 60 лет; e6 - нарывы в горле; c1 - простуда;

c2 - грипп; c3 - уязвимый возраст; c4 - острый фарингит; c5 - лечь в постель и принять аспирин; c6 - вызвать врача

Рис. 3.2. Медицинские правила в виде сети логического вывода

В сети содержатся импликации разных типов: простые, И, ИЛИ, импликации с отрицаниями, а также обратимые и необратимые правила.

Рассуждения начинаются с основания дерева, где все известно, а затем с помощью правил импликации находятся коэффициенты определенности для узлов, поддерживающих нижний информационный уровень. Этот процесс продолжается последовательно до тех пор, пока не будет найден коэффициент определенности для каждого заключения.

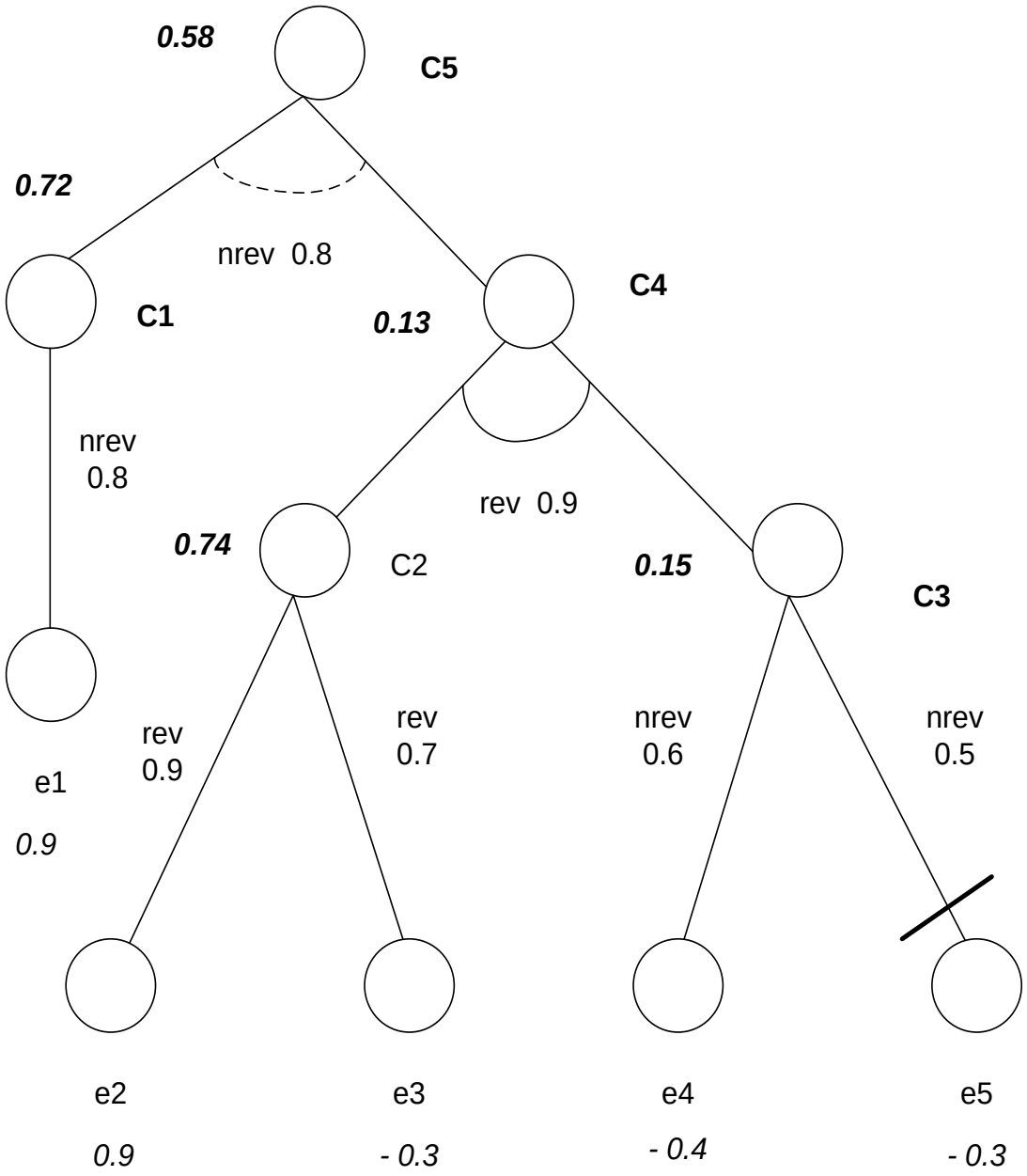


Рис. 3.3. Пример сети логического вывода

Коэффициент определенности c1 может быть вычислен следующим образом: ct (заключение c1) = 0.8 · 0.9 = 0.72.

Это простая необратимая импликация, но, поскольку коэффициент определенности посылки позитивен, правило можно применять.

Для вычисления коэффициента определенности c2 задействованы два правила и оба используются без ограничений, так как они обратимы. Правило слева даст оценку коэффициента определенности для вершины

графа c2 ct (заключение c2) = 0.9 · 0.9 = 0.81. Правило справа даст вторую оценку для вершины графа c2 ct (заключение c2) = - 0.3 · 0.7 = - 0.21.

Здесь приведены два поддерживающих правила, дающих оценку коэффициента определенности с противоположными знаками, поэтому для окончательного ответа применяется формула (3.3):

$$\text{ct (заключение c2)} = \frac{0.81 + (-0.21)}{1 - 0.21} = 0.74 .$$

Для c3 имеется два правила. Правило, связанное с левым поддеревом, не применяется, так как оно необратимо и коэффициент определенности посылки отрицателен. Правило, связанное с правым поддеревом, есть простая импликация. Она необратима и содержит отрицательную посылку. Что нужно сделать? Правило утверждает:

ЕСЛИ (НЕ e5), ТО (c3) ct(импликация) = 0,5 (nrev).

Коэффициент определенности e5 равен минус 0,3. Так как он негативен, то коэффициент определенности посылки в правиле равен 0,3. Коэффициент определенности всего предложения, поддерживающего посылку использует процедуру, предназначенную для простой импликации: ct (заключение c3) = 0,3· 0,5=0,15.

Импликация, поддерживающая c4, включает конъюнкцию (связку И) посылок. Коэффициент определенности посылки определяется следующим образом: ct (свидетельства) =min (0,15; 0,74) = 0,15.

Поскольку правило обратимо, можно использовать посылку в любом интервале определенности. Используя этот результат, вычисяется коэффициент определенности для c4: ct (заключение c4) =0,15 · 0,9=0,13.

Теперь пройден путь вверх по дереву до того места, где можно судить об узле верхнего уровня. Здесь задействовано одно правило, в котором посылки разделены с помощью связки ИЛИ, поэтому

ct (свидетельства) =max (0,72; 0,13) =0,72.

Правило необратимо, но коэффициент определенности посылки позитивен и поэтому вычисление возможно.

Последнее звено в цепочке рассуждений (коэффициент определенности для узла высшего уровня) вычисляется по формуле

ct (заключение c4) = 0,72 · 0,8 = 0,58.

Полученную сеть логического вывода можно использовать для механизма объяснения выбора конкретного заключения.

3.3. Характеристика инструментальной экспертной системы

Состав инструментальной системы включает модуль пользовательского интерфейса, модуль работы с текстом, модуль машины логического вывода, модуль базы знаний.

Инструментальная экспертная система «ANIES» является учебной программой, предназначенной для демонстрации возможностей, которые предоставляют продукционные правила при логическом выводе.

В процессе разработки базы знаний формируется файлы ЭС для конкретной предметной области. Все файлы хранятся в текстовом формате.

Одним из основных управляющих элементов интерфейса является главное меню программы, которое состоит из горизонтального меню, содержащего имена основных групп команд, и выпадающих подменю, позволяющих выбрать конкретную команду или режим работы. Такие пункты горизонтального меню, как «Файл», «Правка», «Помощь», являются стандартными для программ. Они содержат набор команд для работы с файловой системой, облегчения редактирования текста. Используя текстовый редактор, разработчик участвует в создании базы знаний с использованием продукционных правил «IF-THEN-ELSE».

Для распараллеливания работы при создании проекта ЭС можно разрабатывать разными специалистами разделы БЗ с последующим их объединением в один проект (команда «Объединение файлов в проект»).

В меню «Путь исполнения правил» устанавливается последовательность вызова правил из БЗ. По команде «Запуск» выполняется автоматическая процедура логического вывода.

По команде «Трассировка» выполняется процедура логического вывода только с остановом после вызова одного правила с анализом его выполнения. При помощи пунктов меню «Результаты» пользователь может анализировать решение задачи.

База знаний содержит правила, необходимые для логического вывода, а также факты из предметной области, введённые пользователем или выведенные машиной логического вывода.

После запуска ЭС пользователь вводит ответы на вопросы, формируемые системой с указанием коэффициента уверенности в диапазоне [-1; 1]. Отвечая на один вопрос, пользователь может указать несколько ответов или ни одного. Система, используя механизм логического вывода, производит подсчет коэффициентов уверенности всех заключений и отображает ранжированный перечень гипотез. При желании пользователь может просмотреть ход срабатывания правил экспертной системы в виде протокола решения или в режиме трассировки.

Некоторые пункты меню продублированы кнопками быстрого управления. Разработчику экспертной системы предлагается использовать панели инструментов: гипотез, параметров, ключевых слов - при формировании базы правил. Это позволяет значительно сократить количество ошибок и трудоемкость разработки ЭС.

При написании правил можно использовать обычный режим - режим редактирования (копировать, вырезать, вставить, вернуться на шаг назад).

NAME Правило1

IF на_занятиях неусидчив

THEN темперамент холерик [0,1]

ELSE

IF на_занятиях энергичен

THEN темперамент сангвиник [0,1]

ELSE

IF на_занятиях спокоен

THEN темперамент флегматик [0,1]

ELSE темперамент меланхолик [0,1]

END

Синтаксис формируемых правил:

- все правила «IF-THEN-ELSE» и «CASE» должны заканчиваться ключевым словом «END»;
- при написании гипотез, параметров, наименований правил вместо пробела необходимо использовать знак подчёркивания с целью распознавания окончания;
- при составлении циклов «IF-THEN-ELSE» необходимо проставлять все ключевые слова: IF, THEN, ELSE.

Создание и редактирование базы знаний ЭС. Сначала создается новый проект ЭС с помощью команды системы меню «Файл / Новый проект», а затем формируется новый файл базы знаний (БЗ) пользователя. Ранее созданный и сохраненный файл БЗ файл базы знаний можно загрузить с помощью команды «Файл / Открыть существующую БЗ». Пункты меню «Файл / Сохранить» и «Файл / Сохранить как» предназначены для сохранения БЗ пользователя.

Пользователю предлагается использовать следующие функции системы:

- разбиение базы знаний на разделы при создании проекта или при редактировании базы знаний;
- слияние нескольких ранее созданных баз знаний с помощью команды меню «Файл / Объединение файлов в проект». Файлы с других компьютеров должны быть переписаны в текущий каталог системы.

– *Редактор базы знаний* предусматривает несколько режимов работы:
– работа с текстом;
– копирование, удаление, вставка, перемещение блоков;
– вызов с помощью правой кнопки мыши контекстное меню с перечнем параметров и гипотез.

Подготовка баз знаний с помощью данного текстового редактора заключается в последовательном выполнении ряда этапов:

– ввод разделов и правил БЗ;
– редактирование БЗ;
– открытие ранее разработанного файла БЗ;
– отладка правил БЗ по заранее созданным примерам эксперта;
– сохранение файла БЗ на магнитном диске.

После описания ЭС на продукционных правилах пользователь может запустить ЭС на выполнение. При запуске происходит чтение файла БЗ пользователя.

Выполнение экспертной системы. Данный режим работы программы заключается в последовательном выполнении ряда этапов:

– указание пути исполнения правил;
– выбор метода логического вывода и метода поиска решений (прямой метод, обратный метод, «поиск в глубину», «поиск в ширину»);
– запуск на выполнение ЭС;
– анализ результатов работы ЭС путем просмотра протокола решений.

В инструментальной системе предусмотрены средства отладки: диагностики и трассировки. Средства трассировки позволяют пользователю следить за действиями системы и анализировать полученный результат по каждому выбранному правилу. Средства диагностики используются для обнаружения ошибок в базе знаний.

Машина логического вывода выполняет следующие основные функции:

- формирование из непрерывного потока символов (извлекаемых из текстового редактора) лексем - минимальных единиц текста. Для реализации данной функции в структуру модуля работы с текстом был включён блок формирования лексем;
- идентификация лексем (блок идентификации) - определение, является ли лексема известным для оболочки экспертной системы словом или символом;
- контроль корректности исходного текста базы знаний;
- вывод пользователю сообщений о встретившихся ошибках и указание места ошибки в тексте базы знаний;
- поиск решения задачи исходя из правил, имеющихся в базе знаний и фактов с коэффициентами уверенности, полученных от пользователя;
- вычисление коэффициентов уверенности каждой гипотезы и параметра в зависимости от степени доверия к исходным данным.
- Поиск решения можно разделить на несколько отдельных подзадач:
- выполнение процедуры диагностики;
- выполнение (решение) правила. Включает в себя проверку истинности условий правила, переход между ветвями правила в зависимости от знака коэффициента уверенности условия, вычисление коэффициента уверенности найденного заключения правила;
- подсчет результирующего коэффициента уверенности. В базе знаний могут присутствовать несколько правил, приводящих к одному и тому же заключению, поэтому нужно найти такие правила и решить их. Подсчёт коэффициентов уверенности производится в соответствии теории приближённых рассуждений [12];

– задача поиска решения задачи заключается в поиске правил, к которым можно применить полученные заключения и решении найденных правил. Результаты решения правил вновь используются для поиска и т.д., пока не будет найдено решение задачи.

Дополнительно в состав машины логического вывода была включена система управления режимом трассировки базы знаний.

Объяснение хода рассуждений системы выполняется путем протоколирования процесса рассуждений. Система записывает в файл информацию о каждом своем действии, а пользователь может просмотреть полученный протокол в любой момент решения задачи.

Тестирование базы знаний. Для проверки правильности написания базы знаний вводятся развитые средства распознавания ошибочных ситуаций, которые выдают пользователю сообщения об обнаруженных ошибках и указывают место ошибки в тексте базы знаний.

В блок диагностики включены специальные процедуры, которые автоматически исправляют некоторые виды ошибок пользователя в БЗ.

В отладке предусмотрены шесть основных механизмов управления выполнением программы:

– диагностика позволяет обнаружить и исправить некоторые классы ошибок в базе знаний;
– трассировка дает возможность покомандного выполнения выбранного правила;
– просмотр гипотезы дает возможность просмотра коэффициента уверенности гипотезы;
– просмотр параметра дает возможность просмотра коэффициента уверенности параметра;
– просмотр переменной дает возможность просмотра значения переменной;

– протокол решения дает возможность просмотра протокола решения задачи и выявить сработавшие правила.

Контрольные вопросы

1. Какие функции выполняет машина логического вывода?
2. Что такое продукционные правила?
3. Что представляет собой база знаний в продукционной ЭС?
4. Почему в зарубежных ЭС стали применять теорию приближенных рассуждений?
5. В чем состоит неопределенность продукционного правила?
6. Сформулируйте правило модус поненс.
7. Приведите пример с прямым и обратным методом вывода.
8. В чем отличие поиска «в глубину» от «поиска в ширину»?
9. Приведите формулу подсчета результирующего коэффициента для правила «Если e1 И e2 И e3, то С».
10. Приведите формулу подсчета результирующего коэффициента для правила «Если e1 ИЛИ e2 ИЛИ e3, то С».
11. Приведите формулу подсчета результирующего коэффициента для правила «Если НЕ e1 И e2 И e3 , то С».
12. Приведите формулу подсчета результирующего коэффициента для двух правил: «Если e1 И e2 И e3 , то С»; «Если Не e4 И e5 , то С»
13. Выполните расчет коэффициентов уверенности для сети, приведенной на рисунке 3.4. Значения коэффициентов уверенности для посылок: Ке1=0,5; Ке2= - 0,6; Ке3=0,7; Ке4=0,4; Ке5=0,8.

НЕЙРОСЕТЕВЫЕ ЭКСПЕРТНЫЕ СИСТЕМЫ

4.1. Представление знаний в нейросетевой экспертной системе

В последние годы бурно развиваются нейронные экспертные системы. В основе их построения лежит принцип обучения нейронной сети на известных примерах с последующим тестированием по любому входному вектору [1,2,7,8,9].

Экспертная система, реализованная с использованием нейросетевых технологий, называется нейросетевой экспертной системой (НЭС).

Существует ряд задач, в которых не представляется возможным учитывать все реально имеющиеся условия, от которых зависит ответ, а можно выделить приблизительный набор наиболее важных условий. В результате часть условий не учитывается и ответ носит неточный характер. Алгоритмы нахождения ответа не могут быть описаны точно. Такие задачи можно решить с использованием нейросетевых систем (НС).

Нейронные сети могут выступать в качестве модели представления знаний. Во-первых, знания представляются в виде цифровой обучающей выборки, которая представляет базу знаний (до обучения нейросетевой экспертной системы). Во-вторых, это синаптическая карта, сформированная по результатам обучения нейронной сети.

В процессе эксплуатации НЭС может потребоваться дообучение или переобучение нейронной сети экспертной системы.

Дообучение нейронной сети не требует изменения архитектуры нейронной сети и выполняется путем обучения нейронной сети на дополненной или модифицированной обучающей выборке.

Необходимо отметить специфику знаний нейросетевых систем, отличительные характеристики которых приведены в табл. 4.1.

Таблица 4.1

Сравнительные характеристики нейросетевых и экспертных систем

Параметр сравнения	Экспертные системы	Нейросетевые системы
Источник знаний	Формализованный опыт эксперта, выраженный в виде логических утверждений - правил и фактов	Совокупный опыт эксперта-учителя, отбирающего примеры для обучения, и индивидуальный опыт обучающейся на этих примерах нейронной сети
Характер знаний	Формально-логическое «левополушарное» знание в виде правил	Ассоциативное «право-полушарное» знание в виде связей между нейронами сети
Развитие знаний	В форме расширения сово-купности правил и фактов (базы знаний)	В форме дообучения на новой дополненной обучающей выборке
Роль эксперта	Задание полного объема знаний экспертной системы на основе правил	Отбор характерных примеров без специального обоснования своего выбора
Роль системы	Поиск цепочки фактов и правил для доказательства суждения	Формирование индиви-дуального опыта в форме категорий, получаемых на основе примеров, и катего-ризация образов

Переобучение нейронной сети может потребовать изменение архитектуры нейронной сети с последующим ее обучением.

Все неалгоритмируемые или трудноалгоритмируемые задачи, решаемые нейронными сетями, можно классифицировать на два

принципиально различающихся типа: задачи классификации и задачи предикции.

Задачи классификации - основная и очень обширная группа медико-биологических задач. Ответом в них является выбор одного из заранее известного набора вариантов. Классификация может быть бинарной (элементарная классификация). В этом случае набор возможных ответов состоит из двух вариантов (классов) или n-арной, где число классов более двух. Примерами бинарной классификации могут служить как объективные категории (пол человека - мужской или женский; характер опухоли - доброкачественный или злокачественный), так и субъективные категории (здоров человек или болен; наличие или отсутствие склонности к простудным заболеваниям). В некоторых случаях не представляется возможным отнесение ответа задачи к объективной или субъективной категории, но это не имеет принципиального значения для обучения и работы нейросетевой экспертной системы.

Важной чертой задачи классификации по определению является возможность выбора одного и только одного варианта решения (класса). Поэтому постановка диагноза не может считаться одной классификационной задачей, так как у одного человека может одновременно присутствовать несколько патологий. В случае невозможности выбирать один вариант ответа (множественности выбора) задача подразделяется на подзадачи, каждая из которых представляет собой классификационную задачу.

Другой вид задач для нейросетей - задачи предикции, или предсказания. Они подразделяются на предсказание числа (одномерная предикция) и предсказание вектора (векторная предикция, более общий случай). Отличие от классификационных задач заключается в том, что ответ в задачах предикции может быть дробным и принимать любые значения на каком-либо интервале.

Векторная предикция предполагает, что ответ может быть представлен в виде нескольких независимых друг от друга чисел, образующих точку (или вектор) в многомерном пространстве, размерность которого равна количеству предсказываемых чисел. Число координат вектора называется при этом размерностью вектора ответа.

4.2. Способность к обобщению полученных знаний

Одно из важнейших свойств НС – это способность к обобщению полученных знаний. Сеть, натренированная на обучающей выборке, генерирует ожидаемые результаты при подаче на ее вход данных, которые не участвовали в обучении.

Все множество данных можно разделить на обучающее и тестовое подмножества. Пусть R – все множество данных, которые подчиняются правилу R, L – обучающее подмножество. G – Тестовое подмножество, V – контрольное подмножество, которое входит в состав обучающего подмножества и используется для проверки качества обучения сети и решения проблемы переобучения сети [1,2,9].

Способность отображения сетью элементов L (обучающее подмножество) можно считать *показателем накопления обучающих данных*.

Способность распознавания данных, входящих в G (тестовое подмножество) и не используемых при обучении НС, характеризует возможность обобщения знаний или генерализации знаний.

Данные, входящие в G (тестовое подмножество) и в L (обучающее подмножество), должны быть типичными элементами множества R. В обучающем подмножестве не должно быть уникальных данных, свойства которых отличаются от ожидаемых значений.

Феномен обобщения возникает вследствие большого количества комбинаций входных данных, которые могут кодироваться в сети с N входами.

Подбор весов в процессе обучения имеет целью найти такую комбинацию их значений, которая наилучшим образом воспроизводила бы последовательность ожидаемых обучающих пар (X_k - входной вектор, Dk - выходной вектор). При этом наблюдается тесная связь между количеством весов сети и количеством примеров обучающей выборки.

Если бы целью обучения было запоминание всех примеров обучающей выборки, то их количество могло быть равным числу весов. В таком случае, каждый вес соответствовал бы единственной обучающей паре. Но такая сеть не обладала бы свойством обобщения.

Для обретения способности обобщать данные сеть должна быть натренирована на избыточном множестве данных, поскольку тогда веса синапсов будут адаптироваться не к уникальным выборкам, а к их усредненным совокупностям.

Следовательно, для усиления способности к обобщению необходимо не только оптимизировать структуру сети в направлении ее минимизации, но и оперировать достаточно большим объемом обучающих данных.

Истинная цель обучения состоит в таком подборе архитектуры и параметров сети, которые обеспечат минимальную погрешность распознавания тестового подмножества, не участвующего в процессе обучения. Эта погрешность называется погрешностью обобщения $E_G(W)$.

Со статистической точки зрения погрешность обобщения зависит от уровня погрешности обучения $E_L(W)$ и от доверительного интервала ε.

$$E_G(W) =< E_L(W) + \varepsilon\,(P/h, E_L). \qquad (4.1)$$

Параметр ε зависит от уровня погрешности обучения $E_L(W)$ и от отношения количества обучающих пар P к фактическому значению h, называемому *мерой Вапника –Червоненкиса* и обозначаемому VC_{dim} [17].

Эта мера VC_{dim} отражает уровень сложности НС и связана с количеством содержащихся в ней весов. Параметр ε уменьшается по мере возрастания количества обучающих пар к уровню сложности сети.

Обязательным условием хороших способностей к обобщению считается грамотное определение *меры Вапника–Червоненкиса* для сети заданной структуры**.** До настоящего времени не разработано методики определения меры Вапника–Червоненкиса. Известно, что мера зависит от количества синаптических весов [2].

Верхнюю и нижнюю границы меры можно определить в интервале

$$2\,[K/2] \cdot N =< VC_{dim} < 2 \cdot N_w (1+\lg N_n), \qquad (4.2)$$

где [ ] – целая часть числа;

N –размерность входного вектора;

K – количество нейронов скрытого слоя;

N_w – общее количество весов сети;

N_n - общее количество нейронов сети.

Из формулы (4.2) можно сделать следующие выводы:

- нижняя граница диапазона приблизительно равна числу весов, связывающих входной и скрытый слои;
- верхняя граница превышает двукратное суммарное количество всех весов сети.

В связи с невозможностью точного определения меры VC_{dim} в качестве её приближенного значения используется общее количество весов НС.

Таким образом, на погрешность обобщения оказывает влияние отношение количества обучающих выборок к количеству весов сети.

Небольшой объём обучающей выборки (ОВ) при фиксированном количестве весов вызывает хорошую адаптацию сети к его элементам, но не усиливает способности к обобщению, так как в процессе обучения

наблюдается относительное превышение числа подбираемых параметров (весов) над количеством пар фактических и ожидаемых сигналов сети.

Фактическая задача аппроксимации подменяется задачей приближенной интерполяции.

Высокие результаты обобщения достигаются в том случае, когда количество примеров в обучающей выборке в несколько раз превышает меру VC_{dim}. Например, НС, имеющая в скрытом слое 80 нейронов, адаптировала свои выходные сигналы с нулевой погрешностью обучения. Минимизация этой погрешности на слишком малом (относительно количества весов) количестве примеров обучающей выборки спровоцировала случайный характер этих весов, что при переходе к текстовым наборам стало причиной значительных отклонений фактических выходных значений от ожидаемых значений.

Уменьшение количества скрытых нейронов до пяти при неизменном объёме обучающей выборки позволило обеспечить и малую погрешность обучения, и высокий уровень обобщения сети.

Дальнейшее уменьшение количества скрытых нейронов может привести к слишком большой погрешности $E_L(W)$.

Для сокращения количества синапсов в НС применяются *методы редукции сети*, которые можно разбить на две группы:

- исследуют чувствительность целевой функции к удалению веса синапса или нейрона. С их помощью удаляются веса с наименьшим заметным влиянием и процесс обучения продолжается на редуцированной сети;
- методы второй группы связаны с модификацией целевой функции, в которую вводятся компоненты, штрафующие за неэффективную структуру сети. Чаще это элементы, усиливающие малые значения амплитуды весов. Такой способ менее эффективен, чем первый, так

как малые значения весов необязательно ослабляют влияние на функционирование сети.

Принципиально иной подход к редукции сети состоит в последовательном наращивании количества скрытых нейронов вплоть до достижения требуемого уровня нетренированной сети на исходном обучающем множестве. Добавление нейронов происходит, как правило, по результатам оценки способности сети к обобщению после определенного количества циклов обучения.

Эффект переобучения нейронной сети, или эффект бабушкиного воспитания. Погрешность обучения при увеличении количества итераций монотонно уменьшается, тогда как погрешность обобщения снижается только до определенного предела (точка А на рис.4.1), а затем снова начинает расти. Таким образом, слишком долгое обучение может привести к переобучению сети.

Эффект переобучения объясняется в слишком детальной адаптации весов к несущественным флуктуациям обучающих примеров. Такая ситуация наблюдается при использовании нейронной сети с чрезмерным (по сравнению с необходимым) количеством весов. Это особенно заметно, если НС содержит «лишние» веса, которые адаптируются к любым нерегулярностям обучающих примеров, *воспринимая их в качестве важных характеристик.* В результате они становятся источником значительных погрешностей воспроизведения. Для предупреждения этого эффекта в обучающем множестве выделяется контрольное подмножество V, которое в процессе обучения применяется для проверки уровня обобщения сети. В точке А, приведенной на рис. 4.1, обучение прекращается.

Погрешность обобщения можно вычислить по формуле среднеквадратичной ошибки

$$E_G(W)=1/2 \sum^{M} (Yk - Dk)^2, \qquad (4.3)$$

K=1

где M-размер тестового подмножества;

Yk - прогноз НС на выходе выходного слоя, полученный после предъявления сети входного вектора X_K.

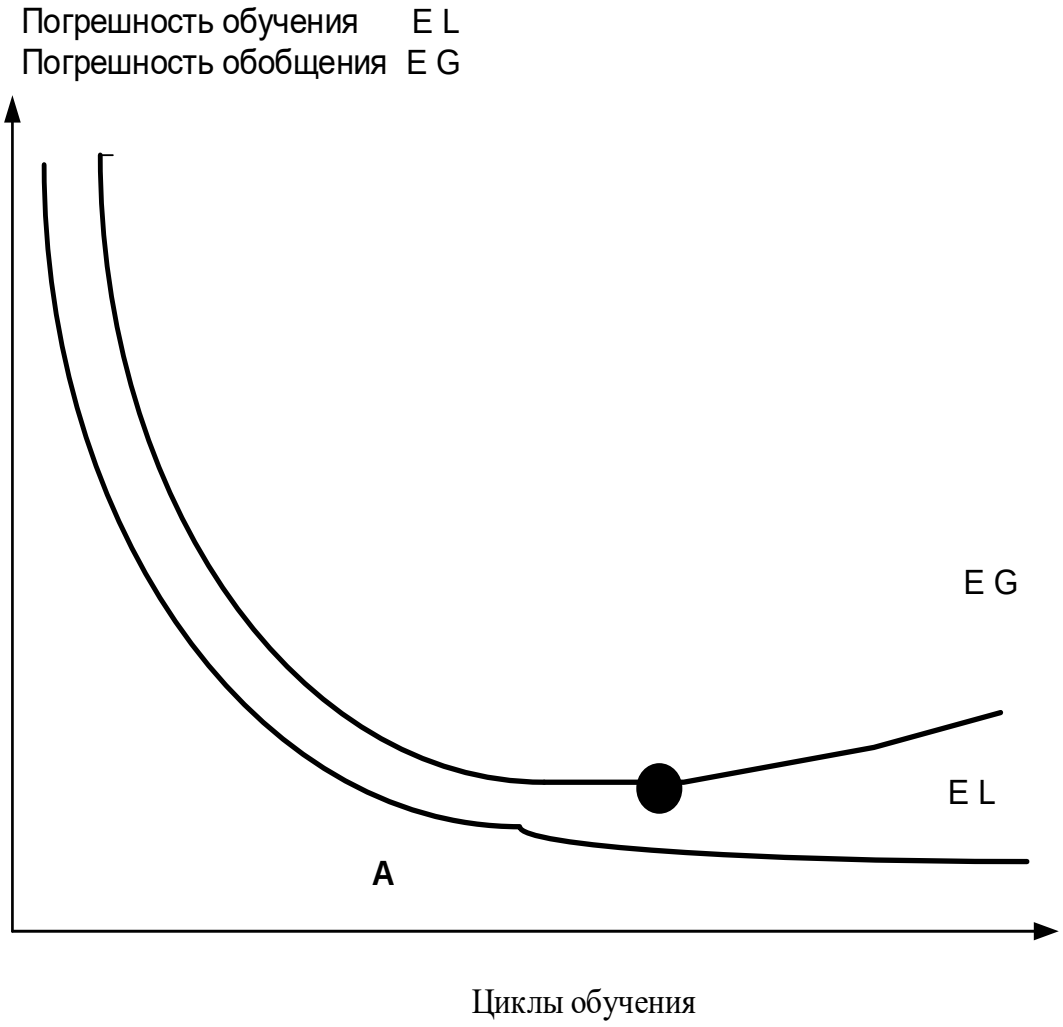


Рис. 4.1. Зависимости погрешности обучения и обобщения

4.3. Нормализация входной и выходной информации

Входная и выходная информация для нейронной сети формируется из векторов IN (входные значения обучающей выборки), OUT (выходные значения обучающей выборки).

Нормализация значений необходима в виду того, что на веса синапсов сети обычно наложены требования принадлежности некоторому диапазону значений. Это приводит к тому, что обычно нельзя подавать сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне.

Кроме того, нормирование исключает доминирование одних входных сигналов перед другими [2].

Поэтому перед подачей входных сигналов их необходимо нормировать в одном из диапазонов: от минус 1 до 1; от минус 0,5 до 0,5; от 0 до 1.

Наиболее простое нормирование сигналов можно выполнить следующим образом. Каждая компонента входного вектора данных x_i заменяется величиной, вычисляемой по формулам

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{0.5(\max\{x_i\} - \min\{x_i\})}, \quad (4.4)$$

$$x_{i0} = \frac{x_i - 0.5(\max\{x_i\} + \min\{x_i\})}{(\max\{x_i\} - \min\{x_i\})}, \quad (4.5)$$

где x_{i0} – нормализованное значение величины;

x_i – нормализуемая величина;

max x_i и min x_i – соответственно максимальное и минимальное значения для данной компоненты, вычисленные по всей обучающей выборке.

Можно нормировать и по-другому, например, пересчитывая выборку так, чтобы разброс данным был единичным, по формуле

$$x_{i0} = \frac{x_i - \min\{x_i\}}{\max\{x_i\} - \min\{x_i\}}. \qquad (4.6)$$

Здесь имеется одна сложность. Любое изменение обучающей выборки должно соответственно менять и правило нормирования данных.

Выбор способа нормализации и значений зависит от конкретной обучающей выборки и выбирается экспериментально.

Для денормализации данных используются обратные преобразования, выполняемые по формулам (4.7), (4.8), (4.9):

$$x_i = \min\{x_i\} + x_{i0}(\max\{x_i\} - \min\{x_i\}); \qquad (4.7)$$

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + 0.5x_{i0}(\max\{x_i\} - \min\{x_i\}), \qquad (4.8)$$

$$x_i = 0.5(\max\{x_i\} + \min\{x_i\}) + x_{i0}(\max\{x_i\} - \min\{x_i\}). \qquad (4.9)$$

Для обучения нейронной сети формируется задачник из обучающих векторов. Каждая обучающая пара состоит из двух векторов (X_k - входной вектор, Dk - выходной вектор). Вектор D_k представляет собой ответы, которые должна выдавать сеть после обучения.

При формировании обучающей выборки встречаются две основные проблемы. Одна проблема заключается в наличии противоречивых выборок – одним и тем же входным данным соответствуют различные выходные. Для решения данной проблемы в нейросетевых программах предусматривается проверка путем сравнения обучающих выборок. Данная проблема возникает при неполноте входной информации.

Другой проблемой является ситуация, когда в таблице данных имеются пробелы. Вместо таких отсутствующих компонент данных можно подавать нуль, можно исключать некомплектные векторы из обучающей выборки, можно перед обучением сети решать задачу заполнения пробелов в данных некоторыми правдоподобными значениями.

При разработке нейросетевой экспертной системы необходимо использовать специальный формат данных. Для этих целей используется специальный компонент, называемый предобработчиком. Разработка эффективных предобработчиков для нейронных сетей является новой, почти совсем не исследованной областью. Большинство разработчиков нейросетевого программного обеспечения склонны возлагать функции предобработки входных данных на пользователя. Это решение технологически неверно. Дело в том, что при постановке задачи для нейронной сети трудно сразу угадать правильный способ предобработки. Для его подбора проводится серия экспериментов. В каждом из экспериментов используется одна и та же обучающая выборка и разные способы предобработки входных данных сети.

Если привычный для человека способ представления входных данных непригоден для нейронной сети, то и формат ответов нейронной сети часто малопригоден для человека. Необходимо интерпретировать ответы нейронной сети. Интерпретация зависит от вида ответа. Так, если ответом нейронной сети является действительное число, то его, как правило, приходится масштабировать и сдвигать для попадания в нужный диапазон ответов. Если сеть используется как классификатор, то выбор интерпретаторов еще шире.

Например, экспертной системе, входной информацией которой служили изображения микрофотографий, предоставили спектральный состав этих изображений. При его использовании обучение нейронной сети происходит не по внешнему виду снимков, а по их цветовой насыщенности. До начала процесса обучения формируются файлы спектров, которые в

дальнейшем используются как обучающие последовательности. Файлы спектров представляют собой массивы из 256 чисел, каждое из которых показывает долю цвета палитры в анализируемом изображении.

Нейронная сеть в ходе обучения находит характерные участки спектра. На практике этот метод зарекомендовал себя как наиболее быстрый. Время обучения, по сравнению с традиционным способом, уменьшается в 15 - 20 раз. Связано это с тем, что в качестве обучающей последовательности подаются не графические массивы размером 512x512 пикселей (262144 элемента), а массивы спектров из 256 элементов. Этот метод имеет еще и то достоинство, что не зависит от ориентации изображения. Как бы ни было подано изображение (прямо, боком, под углом) будут анализироваться не контуры, а его цветовая насыщенность, которая во всех случаях окажется одинаковой.

4.4. Обоснование создания нейросетевой экспертной системы

Применение нейронных сетей (НС) оправдано только при выполнении следующих условий:

- наличие универсального типа архитектуры НС и универсального алгоритма ее обучения;
- наличие предыстории (фиксированного опыта);
- наличие экспертов в данной области, для подготовки которых требуется длительное время;
- наличие согласованной оценки разных экспертов;
- отсутствие четких алгоритмов решения проблемы отсутствуют.

Технология создания, обучения и эксплуатации нейронной сети применительно к задаче распознавания образов строится следующим образом:

– формируется набор обучающих образов;
– осуществляется первоначальная настройка нейронной сети – весовым коэффициентам присваиваются случайные значения;
– обучающие пары входных и выходных векторов, в определенном порядке, предъявляются нейронной сети, которая в соответствии с имеющейся матрицей весов классифицирует предъявленные образы;
– полученные результаты сравниваются с оценками экспертов. На основе этого сравнения происходит коррекция весовых коэффициентов;
– обучение продолжается до тех пор, пока результаты оценки нейронной сетью с заданной вероятностью не совпадут с эталонными оценками экспертов.

После этого нейронная сеть считается настроенной (обученной) на задачу распознавания предъявляемых образов и ее можно использовать для классификации других снимков. Имеется возможность производить последующее дообучение нейронной сети в процессе ее эксплуатации.

Существует большое количество областей, где целесообразно применение нейросетевых экспертных систем. Но процесс создания и обучения экспертной системы на нейронной сети является сложной задачей, решение которой занимает значительное время, а иногда ее создание вообще невозможно. Следовательно, необходима модель, на которой можно попробовать различные варианты структуры нейронной сети, методов обучения, величины и состава обучающей выборки и выявить наиболее подходящий вариант.

Программные продукты, моделирующие нейронную сеть, имеют ряд недостатков. Самый главный из них – программы не предоставляют пользователю инструментов создания нейронной сети и оптимизации обучающей выборки. Размер, состав и даже порядок следования примеров в обучающей выборке значительно влияет на скорость и качество обучения нейронной сети. Оптимизация обучающей выборки позволяет сократить

количество примеров в десятки раз, при этом качество обучения не ухудшается, а скорость обучения значительно возрастает.

Задача формирования выборки является достаточно сложной. Например, при создании экспертной системы на основе нейронной сети нет необходимости обучать сеть для всех наборов входных значений, так, как только некоторые наборы входных данный дают максимум информации для пользователя, остальные же не несут практически никакой информации. Следовательно, требуется найти методы, позволяющие управлять наполнением обучающей выборки и минимизировать количество малоинформативных примеров. Для такой оптимизации целесообразно использовать математические методы, например, кластерный и факторный анализ.

В последнее время в научной литературе все больше внимания уделяется зависимости скорости и качества обучения нейронной сети от обучающей выборки. Следовательно, необходим эффективный инструмент по созданию обучающих примеров, который позволит не только упростить формирование базы знаний нейронной сети, но и исследовать различного рода зависимости.

В некоторых пакетах нейросетевых программ имеются средства поиска оптимальной архитектуры для заданной обучающей выборки (количество слоев и количество нейронов в каждом слое, тип и параметры активационной функции и т.п.). В настоящее время наука не дает никаких однозначных указаний на то, как выбрать структуру сети. Есть два варианта.

Первый вариант: взять заведомо большее число слоев и нейронов, а потом их редуцировать.

Второй вариант: постепенно наращивать количество слоев и нейронов в них, производя обучение после каждого изменения в структуре сети. Этот вариант даст наиболее оптимальную структуру нейронной сети, но займет значительное время.

4.5. Программы моделирования искусственных нейронных сетей

В настоящее время известно более 200 нейропакетов [2], выпускаемых рядом фирм и отдельными исследователями и позволяющих конструировать, обучать и использовать нейронные сети для решения практических задач. Трудоемкость разработки НЭС сокращается в случае применения готовых нейросетевых программ.

Наиболее известные и популярные нейросистемы, и их производители следующие [2].

Пакет *Neural 10,* разработанный в 1992 году, использует одну нейросетевую парадигму – двухслойную сеть прямого распространения с алгоритмом обучения (обратного распространения ошибки). Активационная функция нейронов скрытого слоя – сигмоид, а выходных нейронов – линейная.

Программа *Neuro Pro (версия 0.25)* является свободно распространяемой альфа-версией нейросетевого программного продукта для работы с искусственными нейронными сетями и извлечения знаний из таблиц данных с помощью нейронных сетей в среде Windows. Возможности программы, следующие [2]:

- работа с файлами в форматах *.dbf и *.db;
- создание слоистых нейронных сетей с числом слоев до 10 и нейронов в слое до 100;
- использование нелинейной сигмоидной функции f(A)= A/(|A| +c);
- применение методов контрастирования (упрощения) нейронной сети;
- обучение нейронной сети по алгоритму обратного распространения ошибки с применением одного из методов оптимизации (градиентный спуск, модифицированный ParTan, метод сопряженных градиентов);
- генерация вербального описания нейронной сети.

Нейропакет QwikNet32 (версия 2.1) предназначен для работы в среде Windows и реализует один тип нейронной сети – многослойную сеть прямого распространения с числом скрытых слоев до пяти и с набором из шести алгоритмов обучения.

Возможности нейропакета QwikNet32, следующие [2]:

- применение четырех видов функций активации (сигмоидная, гиперболический тангенс, линейная, функция Гаусса);
- использование одного из шести алгоритмов обучения;
- обучение с перекрестным пересечением (обучающая выборка делится автоматически на два набора: 90% - для обучения и 10% - для тестирования и проверки качества обучения);
- рандомизация значений весов синапсов перед обучением;
- установка коэффициента скорости обучения;
- графическое представление результатов обучения;
- вывод сообщения о корректности обучения для всех выходов сети.

Нейропакет Neural Planner [2] представляет собой программную оболочку, позволяющую моделировать нейронные сети различной конфигурации.

Возможности нейропакета Neural Planner, следующие:

- добавление и удаление входного и выходного нейронов, синапса, соединение и разъединение двух слоев нейронов;
- установка параметров активационной функции и количества циклов обучения;
- выбор вида графика изменения средней ошибки при обучении сети;
- отображение информации о любом выбранном в сети нейроне;
- отображение средней и целевой ошибки;
- редактирование файлов обучающей выборки при помощи электронных таблиц MS Excel.

Пакет программ NeuralWorks Professional II Plus является одним из последних версий программного продукта NeuralWorks, разработаного фирмой NeuralWare [8]. Пакет содержит программные модели десятков архитектур нейронных сетей.

Пакет программ ExploreNet 3000 является разработкой фирмы HNC, основанной профессором Робертом Хехт-Нильсеном [8]. Пакет предоставляет широкие возможности по моделированию и управлению данными. В качестве ускорителя используется аппаратные разработки фирмы HNC - нейропроцессоры ANZA и ANZA+, являющиеся одними из первых аппаратных решений. Фирма предложила также средство для разработки прикладных программ - специализированный язык программирования AXON, основанный на языке С.

Оболочка NeuroShell 2.0. Достоинством этой программы является совместимость с популярным пакетом управления данными MicroSoft Excel, что делает продукт удобным для массового использования.

Пакет *Neuro Office* предназначен для проектирования интеллектуальных программных модулей, построенных на основе нейронных сетей с ядерной организацией. Результатом проектирования является обученная нейронная сеть с программным интерфейсом, соответствующим модели многокомпанентных объектов.

Нейропакет NeuralWorks Professional является мощным средством для моделирования нейронных сетей. В нем реализованы 28 нейронных парадигм, а также большое количество алгоритмов обучения. Имеется хорошая система визуализации данных: структуры нейронной сети, изменения ошибки обучения, изменения весов и их корреляции в процессе обучения.

Пакет NeuroShell 2 (фирма Neuron Data) использует порождающие правила для предварительной обработки информации, которая затем передаётся в нейронную сеть. Полученная на выходе нейронной сети информация также может быть обработана с помощью системы правил.

Нейропакет BrainMaker Pro является простым нейропакетом для моделирования многослойных нейронных сетей, обучаемых с помощью алгоритма обратного распространения ошибки. Основным его достоинством является большое число параметров настройки алгоритма обучения.

Пакет *CubiCalc RTC* фирмы HyperLogic, разработанный в 1990 году, представляет собой первый полнофункциональный программный пакет для разработки приложений на основе нечеткой логики. Пакет CubiCalc - первый профессиональный пакет, реализующий методы нечеткой логики. Фактически пакет CubiCalc представляет собой своего рода экспертную систему, в которой пользователь задает набор правил типа "Если-То", а система пытается на основе этих правил адекватно реагировать на параметры текущей ситуации.

Недавно вышедшая на рынок вторая версия пакета CubiCalc фирмы HyperLogic является одной из наиболее мощных экспертных систем на основе нечеткой логики. Пакет содержит интерактивную оболочку для разработки нечетких экспертных систем и систем управления, а также run-time модуль, позволяющий оформлять созданные пользователем системы в виде отдельных программ. От других пакетов CubiCalc отличает также наличие весьма мощной утилиты Rule Maker, позволяющей решать одну из основных проблем в работе с нечеткой логикой - автоматическое построение нечетких правил. Сегодня CubiCalc применяется при решении десятков различных задач - от адаптивного управления оптовыми складами до моделирования рынка фьючерсных контрактов. Большинство пользователей CubiCalc - это финансовые и политические аналитики.

Достоинством вышеперечисленных систем является простота создания и понятность процесса вывода, отсутствие проблем при внесении изменений. В качестве недостатка можно отметить высокую стоимость программного продукта.

Нейропакеты, перечисленные выше, являются относительно дорогими и предназначены для профессионального использования.

4.6. Примеры реализации нейронных экспертных систем

Приведенные ниже материалы, подобранные профессором Красноярской государственной медицинской академии Д.А. Россиевым, содержат интересные разработки по нейросетевым экспертным системам.

В Италии разработана чрезвычайно интересная экспертная система для диагностики и лечения артериальной гипертонии. Система включает в себя три нейросетевых модуля, причем ответы одних являются входными данными для других. В начале исследования больному проводят измерение систолического и диастолического давления каждые полчаса в течение суток. Данные за каждый час усредняются. Таким образом, образуется массив из 48 величин артериального давления (по 24 для систолического и диастолического).

После этого первый модуль, состоящий из двух трехслойных нейросетей (в каждой из которых 2 входных, 4 "скрытых" и 24 выходных нейрона), на основании данных о поле и возрасте больного рассчитывает аналогичные "должные" величины и сравнивает их с реальными.

Параллельно второй модуль (двухслойная нейросеть с 17 входными и 4 выходными нейронами) на основании клинических данных (симптоматика, анамнез) рассчитывает возможные сочетания гипотензивных лекарственных средств, которые могут быть использованы для лечения данного больного.

Данные, снятые с выходов обоих модулей, вместе с клиническими данными подаются на вход последнего, третьего модуля (6-слойная нейросеть). Этот модуль оперирует 4 группами гипотензивных препаратов (диуретики, бетаадреноблокаторы, ингибиторы ангиотензина, блокаторы

кальциевых каналов). Цель - назначить суточный (почасовой) график приема больным лекарств каждой (если требуется) из 4 групп. Поэтому этот модуль имеет 96 выходных нейронов (4 препарата на 24 часа). С каждого выходного нейрона снимается доза, соответствующая одному препарату, назначаемому на данный час суток. Естественно, что в реальной ситуации большинство выходных данных равны нулю.

Таким образом, создается оптимальная для пациента схема лечения гипертонии. Нужно отметить, что система учитывает некоторые особенности приема препаратов больными, например, затруднение приема препаратов ночью (назначает ночной прием только в крайних случаях), запрет на назначение мочегонных лекарств на ночь.

Отличительной чертой системы является возможность пользователя (врача) передавать нейронной сети свой опыт. Для этого создателями программы предусмотрен специальный блок, который выводит на экран компьютера суточные кривые артериального давления и предлагает врачу ввести в компьютер суточную схему приема гипотензивных препаратов в необходимых, по его мнению, дозах. Введенный пример помещается в базу данных. В любое время можно инициировать доучивание нейронных сетей с новыми примерами.

В одной из работ приводится метод выявления атеросклеротических бляшек в артериях с использованием нейронных сетей. Для этого применяется нейросеть, интерпретирующая флюоресцентные спектры, получаемые при исследовании тканей с помощью лазера.

Аналогичным образом проводится диагностика заболеваний периферических сосудов, например, определение форм артериита.

Проводится комплекс исследований по использованию нейросетей для диагностики инфаркта миокарда. Автор приводит данные по чувствительности (77,7%) и специфичности (97,2%) нейросетевого теста. С помощью нейронной сети устанавливается диагностическую значимость клинических параметров при диагностике инфаркта миокарда.

Нейросетевой анализ акустических сигналов позволяет проводить диагностику клапанных шумов сердца и оценивать систолическую и диастолическую фазы сердечного сокращения с постановкой предварительного диагноза.

Нейросети используются терапевтами для диагностики заболеваний печени по лабораторным данным исследования функций печени, а также для дифференциальной диагностики заболеваний печени и желчного пузыря.

Нейропрограммы могут с успехом работать с медицинскими данными, относящимися к субъективным категориям, например, в психиатрии. Оценка субъективных данных дает возможность распознавания психических симптомов и диагностики некоторых психиатрических симптомов.

Актуальная проблема диагностики злокачественных новообразований, возможно, получит новый уровень осмысления с началом применения нейроалгоритмов (80%-ая точность ранней диагностики меланом кожи - одного из самых злокачественных заболеваний).

Одним из серьезных направлений применения нейронных сетей является интерпретация медицинских данных. В последние годы идет бурное развитие новых средств диагностики и лечения. При этом наблюдается «вторая волна» изучения и использования древних, старинных методов и, наоборот, применение последних технических новшеств. При этом встает проблема их грамотной и корректной интерпретации. Поиск глубинных закономерностей между получаемыми данными и патологическими процессами начинает отставать от разработки все новых и новых методов, поэтому применение для этой цели нейросетей может оказаться чрезвычайно выгодным.

Классической проблемой в кардиологии является интерпретация электрокардиограмм, требующая значительного опыта врача. Сотрудники Университета Глазго (Великобритания) ведут исследования по применению

нейросетей для диагностики инфарктов миокарда по результатам анализа электрокардиограмм. Входными данными для сетей являются избранные параметры 12-канальной электрокардиограммы и 12-канальной векторкардиограммы (длины зубцов, расстояния между зубцами).

Исследователи обучили огромное количество нейросетей (167 сетей для диагностики инфаркта миокарда передней стенки и 139 сетей для инфаркта нижней стенки) на массиве данных из 360 электрокардиограмм. Обученные сети затем тестировали отдельную выборку с заранее известными ответами (493 случая). Одновременно для получения отдельной серии ответов на тестируемой выборке был использован логический метод (с заранее заданным алгоритмом). Затем сравнивались результаты тестирования выборки лучшими нейросетями и с помощью логического алгоритма.

Сравнение показало, что во многих случаях чувствительность и специфичность нейросетевого теста оказались выше, чем у логического метода. В тех случаях, когда логический алгоритм решения задачи все-таки можно выстроить, разумно комбинировать в экспертных системах оба подхода.

Интерпретация ЭКГ с помощью нейросетей была применена для диагностики злокачественных желудочковых аритмий. Трехслойная сеть с 230 входными синапсами была обучена на 190 пациентов. Результаты тестирования сравнивались с логическим методом интерпретации данных. Показано, что нейросетевой тест обладает большей чувствительностью (73% по сравнению с 70% для логического метода) и специфичностью (83% по сравнению с 59%).

Актуальным является моделирование работы электрокардиостимуляторов (искусственных водителей ритма) на базе нейронных сетей. Выпускаемые за рубежом электрокардиостимуляторы задают ритм не жестко, а в зависимости от исходного ритма, генерируемого синусовым узлом сердца. Электрокардиостимулятор представляет собой

систему вход-преобразование-выход, где входом является ритм синусового узла, выходом - собственный ритм электрокардиостимулятора, а преобразование осуществляется по заданному логическому алгоритму. Авторы смоделировали замену логического преобразователя нейронной сетью, так как взаимоотношения между генерацией импульсов в синусовом узле и требуемым ритмом не линейны и применяемые алгоритмы на практике не всегда эффективны. Нейросеть, обученная на 27 здоровых людях в ситуациях с различной физической нагрузкой, показала гораздо лучшую способность задавать ритм, чем логический алгоритм, применяющийся в электрокардиостимуляторе.

Одной из самых сложных задач для нейросетей в практической медицине является обработка и распознавание сложных образов, например, рентгенограмм. Разработана экспертная система интерпретации рентгенограмм груди у новорожденных с выбором одного и более диагнозов из двенадцати.

Созданы нейросетевые экспертные системы для классификации опухолей молочной железы (определения, доброкачественная опухоль, или злокачественная) по данным маммографии (сканограмма молочной железы). Точность диагностики до применения нейросети составляла не более 75%. При тестировании системы, нейросеть, анализирующая сканограмму, давала правильный ответ в 100% случаев. Изображение, получаемое в результате метода, представляется в виде матрицы точек размером 1024x1024 пиксела с 10-битовой шкалой яркости. Изображение подается на нейросеть, имеющую 2 входных, 80 «скрытых» и 2 выходных нейрона. При этом один из выходных нейронов «отвечает» за доброкачественную опухоль, другой - за злокачественную опухоль. Диагноз определяется в зависимости от выходного нейрона, выдавшего больший по величине ответ.

Несколько работ посвящены нейросетевой обработке лабораторных анализов и тестов. Приводится нейросетевой метод интерпретации

лабораторных данных биохимического анализа крови. В работе показаны преимущества нейронных сетей в сравнении с линейным дискриминантным анализом, которым параллельно обрабатывались данные.

Особое место среди нейросетевых экспертных систем занимают прогностические модели, применяемые, например, для прогнозирования исходов заболеваний.

В 1990 году американская фирма «Апачи Медикл Системз Инк.» установила в реанимационном отделении одной из больниц штата Мичиган экспертную систему «Апачи – III». Ее цель - прогнозирование исхода заболевания у больных, находящихся в тяжелом состоянии. Для прогноза в компьютер необходимо ввести 27 параметров больного: первичный диагноз, симптомы, степень утраты сознания, наличие или отсутствие других заболеваний.

После этого система выдает вероятность выживания больного в диапазоне от 0 до 100 процентов. Ценность применения системы заключается в том, что она позволяет очень быстро оценить динамику изменения состояния больного, незаметную «на глаз». Например, можно получить ответ у системы до и после введения какого-либо лекарства и, сравнив ответы, выяснить, будет ли наблюдаться эффект от терапии.

Необходимо отметить, что система была обучена на данных, взятых из историй болезней 17448 пациентов, лечившихся в 40 больницах штата в 1989 году. Очевидно, что если качество работы системы обеспечивается таким большим объемом выборки, возможности перенастройки системы не слишком велики. Поэтому данная система не способна к дообучению в процессе работы, опыт «зашит» в нее жестко. Это может быть существенным недостатком при установке программы в регионы, резко отличающиеся по социально-географическим условиям от тех, где проводилось обучение. Кроме того, огромный массив примеров для обучения повышает стоимость программы.

Прогностические нейросетевые модели могут использоваться в демографии и организации здравоохранения.

Судя по литературным данным, именно биологические научные исследования являются наиболее развиваемой областью применения нейросетей. В последнее время биологи, знакомые с исследованиями в области нейроинформатики, приходят к выводу, что многие системы в живых организмах работают по принципам, сходным с алгоритмами нейронных сетей (или наоборот, нейронные сети работают по принципу биосистем). Таким образом, можно наблюдать «взаимное стимулирование» научных разработок в биологии и нейроинформатике.

Эндокринная система человека рассматривается как нейронная сеть из 30 элементов, которые представлены различными гормонами, взаимодействующими друг с другом с помощью прямых и обратных связей. Похожие исследования проводятся для иммунной системы.

Применение нейросетей для исследований в области нейрофизиологии строится на похожих принципах функционирования нейросетей и нервных структур живых организмов. С помощью нейросети осуществлена попытка моделирования простейшей нервной системы.

Сделана попытка применения нейросети для классификации живых организмов: нередко биологам, открывающим новые виды организмов, требуется определить, к какому виду (классу, типу) относится тот или иной представитель флоры или фауны (как правило, это касается микроорганизмов и растений). Система способна работать при отсутствии некоторых входных данных. Это является существенным преимуществом, так как часто при изучении живых объектов не всегда возможно получить всю необходимую информацию.

Нейросети использованы для идентификации человеческих хромосом. В биологических исследованиях, а также в криминалистике часто бывает нужно определить, к какой из 23 имеющихся у человека пар хромосом относится выделенная хромосома. Точность существующих

методов достигала от 75 до 85%. Нейроклассификатор, на вход которого подается 30 признаков изображения хромосомы, определяет ответ с точностью, приближающейся к 100%.

Анализ публикаций о применении нейросетевых технологий в медицине показывает, что практически отсутствуют какие-либо методологии разработки нейросетевых медицинских систем, о чем свидетельствует как отсутствие работ такого профиля, так и огромное разнообразие подходов к нейросетевым алгоритмам обучения и архитектурам нейронных сетей.

Нужно отметить, что все рассмотренные медицинские приложения нейронных сетей для практического здравоохранения (диагностика, лечение, прогнозирование) созданы зарубежными авторами. Большинство отечественных работ направлено на исследование самих нейронных сетей и моделирование с их помощью некоторых биологических процессов (в основном, функций нервной системы).

В Санкт-Петербурге (в Боткинской больнице) разработана нейросетевая ЭС по диагностике некоторых классов болезней, которые плохо диагностируются врачами. Результаты проверки качества работы НЭС свидетельствуют о высокой достоверности результатов (94%).

Данная НЭС разработана на базе нейропакета Neural Planner. Нейронная сеть была обучена при помощи 18 примеров обучающей выборки. Время обучения составляет 2 минуты при заданном пороге ошибки 0,05. Сеть обучилась за 7500 циклов.

После задания пациенту вопросов ответы записываются в бинарном виде: «Да (1)», «Нет (0)». В результате формируется бинарный входной вектор, который предъявляется НЭС, которая определяет класс заболевания у пациента.

Наиболее важным отличием предлагаемого подхода является возможность конструирования экспертных систем самим специалистом, который может передать нейронной сети свой индивидуальный опыт, опыт

своих коллег или обучать сеть на реальных данных, полученных путем наблюдений.

В литературе [2,7,8,9] приведены созданные НЭС: ADAM, GURU, NESTOR, Neural, NEXPERT Object, Smart Elements.

При использовании коммерческих пакетов прикладных нейросетевых программ трудоемкость создания НЭС сокращается в десятки раз.

Преимущества НЭС

1. Нейросетевые ЭС принимают решения на основе опыта и позволяют моделировать ситуацию принятия решения;
2. Создателю НЭС не требуется устанавливать взаимосвязи между входными данными и необходимым решением, затрачивая время на статистическую обработку, подбор математического аппарата и проверку моделей;
3. Применение нейропакетов позволяет упрощать процесс создания НЭС;
4. Решение, принимаемое НЭС, не является категоричным, так как она выдает решение со степенью уверенности и оставляет лицу, принимающему решение (ЛПР) критически оценивать ответ;
5. Ответ НЭС дает очень быстро (доли секунды), что позволяет использовать их в динамических ЭС для незамедлительного принятия решения (например, применение НЭС в реанимации).

Недостатки НЭС

1. Программные коммерческие нейропакеты сравнительно дорогие (тысячи долларов);
2. проблема выбора оптимальной архитектуры НС может быть решена только при использовании мощных нейросетевых программ;
3. отсутствует механизм оптимизации обучающей выборки.

4.7. Пример учебной нейросетевой экспертной системы

Разработанная на кафедре ЭВМ ВятГУ учебная нейросетевая экспертная система (НЭС) предназначена для создания студентами в учебном процессе демонстрационного прототипа экспертной системы (ЭС) с целью изучения влияния параметров нейронной сети на скорость и качество ее обучения.

Учебная НЭС включает в себя следующие программные модули:

- модуль базы знаний, осуществляющий взаимодействие нейронной сети с данными базы знаний;
- модуль данных нейронной сети, обеспечивающий хранение данных нейронной сети в памяти и на носителе информации;
- модуль обучения нейронной сети, выполняющий обучение нейронной сети на основании алгоритма обучения и дополнительных параметров;
- модуль тестирования обученной нейронной сети;
- модуль настройки, предназначенный для настройки параметров обучения нейронной сети;
- модуль пользовательского интерфейса, осуществляющий взаимодействие пользователя с НЭС.

Программные модули созданы с использованием средств визуального программирования Borland Delphi 6.0.

Обучение нейронной сети осуществляется в точном соответствии с алгоритмом обратного распространения ошибки [2].

В программе имеется возможность установить два вида активационной функции (сигмоидальная или гиперболический тангенс) и Параметры нейронной сети (выбор числа слоев до 10 и нейронов в слоях до 100, а также задание точности и коэффициента скорости обучения).

Выбор этих параметров возлагается на пользователя с помощью модуля настройки. В программе реализована возможность изменения

параметров обученной нейронной сети и редактирования обучающей выборки, после чего процесс обучения нейронной сети требуется повторить. Диалоговое окно выбора параметров нейронной сети для обучения приведено на рис. 4.2.

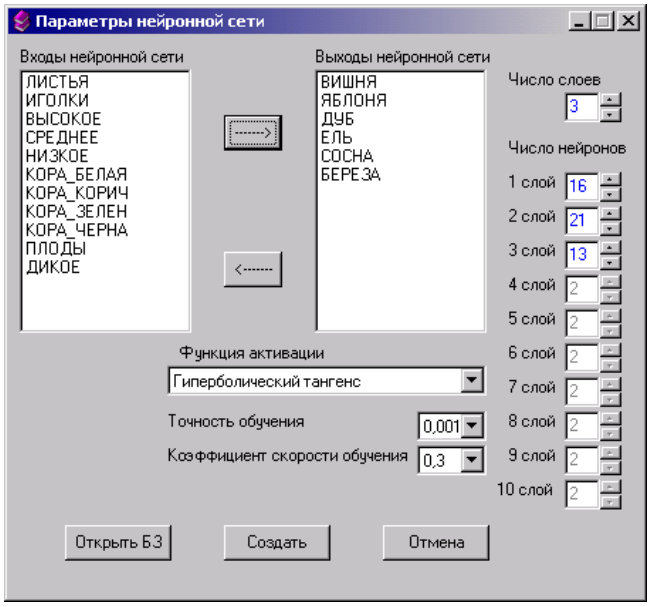


Рис. 4.2. Диалоговое окно выбора параметров нейронной сети

Разработанная учебная НЭС предоставляет студентам возможность быстрого создания демонстрационного прототипа ЭС с выбором параметров обучения нейронной сети и последующим анализом влияния различных параметров нейронной сети на скорость и качество обучения.

Контрольные вопросы

1. Представление знаний в нейросетевой экспертной системе.
2. Понятие дообучения и переобучения нейронной сети.
3. Сравнительные характеристики нейросетевых и экспертных систем.
4. Сравнительные характеристики нейропакетов.
5. Выбор оптимальной конфигурации нейронной сети.

6. Назначение меры Вапника-Червоненкиса.
7. Практические рекомендации по выбору архитектуры нейронной сети.
8. Необходимость использования разных видов активационных функций.
9. Необходимость нормализации входных и выходных данных обучающей выборки.
10. Принципы контрастирования нейронных сетей.
11. Назначение обучающего, контрольного и тестового множества.
12. Принцип вычисления погрешности обобщения.
13. Объяснение эффекта переобучения нейронной сети.
14. Примеры нейросетевых экспертных систем.
15. Проблемы создания нейросетевой экспертной системы.

НЕЧЕТКИЕ ЭКСПЕРТНЫЕ СИСТЕМЫ

5.1. Краткий обзор нечетких экспертных систем

Для создания нечетких экспертных систем используются коммерческие инструментальные программные комплексы [3, 9, 10].

CubiCalc 2.0 RTC – одна из наиболее мощных коммерческих экспертных систем на основе нечеткой логики, позволяющая создавать собственные прикладные экспертные системы.

CubiQuick – дешевая «университетская» версия пакета CubiCalc.

RuleMaker – программа автоматического извлечения нечетких правил из входных данных.

FuziCalc – электронная таблица с нечеткими полями, позволяющая делать быстрые оценки при неточно известных данных без накопления ошибки.

OWL – пакет, содержащий исходные тексты всех известных видов нейронных сетей, нечеткой ассоциативной памяти.

Fuzzy Logic Toolbox – пакет прикладных программ, входящих в состав среды MatLab.

ExSys – программный комплекс для проектирования экспертной системы в интерактивном режиме.

Maple V – пакет символьных вычислений.

FuzzyCLIPS – оболочка экспертной системы, основанная на правилах, используемая для представления и управления нечеткими фактами и правилами.

Таким образом, нечеткие экспертные системы нашли широкое применение во многих областях. Их достоинством является простота создания и понятность процесса вывода, отсутствие проблем при внесении изменений. Все это позволяет сделать вывод о том, что разрабатываемая инструментальная система может применяться для создания экспертных систем.

Базовым понятием Fuzzy Logic Toolbox является FIS-структура – система нечеткого вывода. Пакет состоит из трех частей, предназначенных для создания нечетких систем, гибридных систем и кластеризации.

В первой части создаются классические нечеткие системы вывода на основе алгоритмов Мамдани и Сугено [3].

Во второй части для вывода используется гибридная (нечеткая) нейронная сеть. При этом формируемые правила системы представляются обучающей выборкой для сети. В данном случае используется только один алгоритм нечеткого вывода – Сугено (нулевого или первого порядков),

может быть задана только одна выходная переменная и всем правилам приписывается один и тот же единичный вес.

В третьей части происходит выявление центров кластеров, т.е. точек в многомерном пространстве данных, около которых группируются (скапливаются) экспериментальные данные. Выявление подобных центров является значимым этапом при предварительной обработке данных, поскольку позволяет сопоставить с этими центрами функции принадлежности переменных при последующем проектировании системы нечеткого вывода.

Для задания функций принадлежности используется редактор функций принадлежности (Membership Function Editor).

В программе просмотра правил (Rule Viewer) можно задавать числовые параметры на входе нечеткой системы и получать значения выходных параметров. При этом программа демонстрирует на экране весь процесс вывода в виде графиков.

Программа просмотра поверхности (Surface Viewer) позволяет просматривать двух- или трехмерный график, где функция по оси Z является выходной функцией системы, а функции (X и Y) входными. Это позволяет сразу просмотреть работу системы вывода на всех вариантах входных данных.

Программный комплекс ExSys. ExSys [11] – универсальный программный комплекс для проектирования экспертной системы, позволяющего исследователям проходить все этапы создания системы в интерактивном режиме.

В основе экспертной системы ExSys лежит общая теория построения нечетких экспертных систем, а также ряд специальных методов автоматического формирования структуры нечетких множеств и генерации нечетких правил. Общая структура программного комплекса представлена на рис. 5.1. Система состоит из нескольких модулей, каждый из которых осуществляет свою функцию. Ядро является основным модулем, который

осуществляет связь всех модулей и обмен данных между ними. Таблицы названий признаков, нечетких множеств и нечетких правил представляют собой стандартные таблицы базы данных типа Paradox. Посредством графической оболочки эксперт контролирует все этапы проектирования системы.

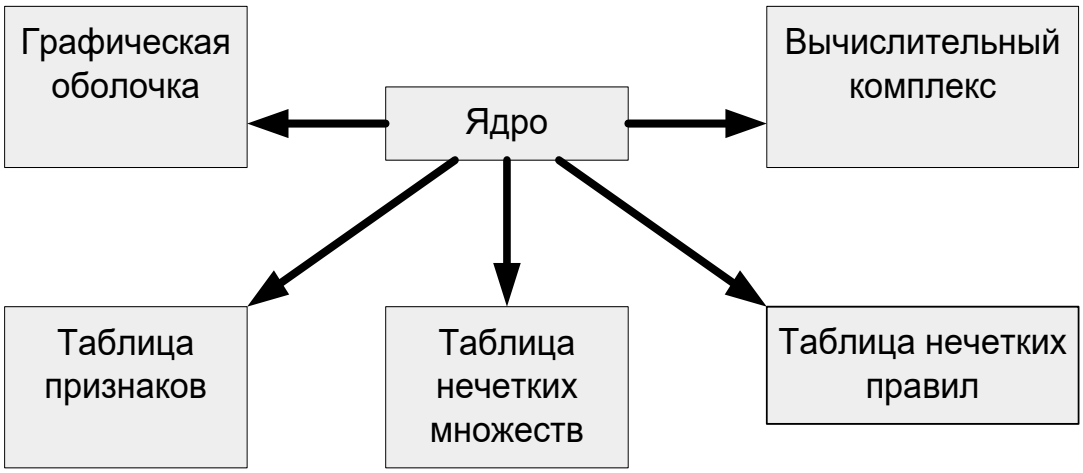


Рис. 5.1. Общая схема программы экспертной системы ExSys

Данная система может работать в нескольких режимах. В том случае, когда отсутствует какая-либо априорная информация об исследуемой области, все необходимые шаги могут быть проведены в автоматическом режиме. Пользователь должен лишь указать число нечетких множеств для каждого признака и присвоить им названия. В случае необходимости эксперт может задавать примерные границы между множествами.

Совокупность нечетких правил вывода (база знаний) генерируется автоматически с использованием таблицы прецедентов или/и вводится экспертом. Он также в случае необходимости может изменить посылки или веса некоторых правил. Форма нечетких множеств определяется путем процедуры оптимизации. Различные режимы работы системы представляются от полностью автономного режима до случая, когда все нечеткие множества и правила вывода задаются экспертом. В последнем случае система функционирует как классический нечеткий контроллер.

Нечеткая экспертная система ExSys позволяет:

– задавать регулярное разбиение признакового пространства в интерактивном режиме либо генерировать его автоматически;
– определять форму нечетких множеств ручным способом либо путем оптимизации по прецедентам;
– создавать базу знаний системы через интерактивный интерфейс, а также путем генерации методом эффективных сужений с различным набором параметров;
– отображать все данные внутри системы через стандартные таблицы базы данных типа Paradox.

5.2. Реализация обучающей нечеткой экспертной системы

На кафедре ЭВМ ВятГУ разработана нечеткая система для оценки знаний студентов на базе нечетких множеств, которая включает подсистему формирования БЗ (конструктор тестов) и подсистему функционирования (проигрыватель тестов). Конструктор тестов используется экспертом-преподавателем, формирующим базу тестов. Преподаватель может ввести пять типов вопросов:

- один правильный ответ;
- несколько правильных ответов;
- указать последовательность шагов в предложенном алгоритме;
- ввести число или слово;
- создать в графическом редакторе фрагмент схемы, рисунка, чертежа и текстовый вопрос к этому фрагменту.
- установить параметры фаззификации и дефаззификации (рис. 5.2), которые являются основой для оценивания знаний студента с использованием нечетких множеств.

С помощью конструктора тестов можно пополнять и редактировать базу тестов, устанавливать коэффициент значимости раздела дисциплины на интервале [0,1], количество вопросов для тестирования знаний студента из общего количества вопросов базы тестов.

Для определения параметров фаззификации и дефаззификации нечеткой системы можно использовать автонастройку параметров или подобрать точки A, B, C на функции фаззификации и выбрать метод дефаззификации в ручном режиме [3]. Наиболее эффективными для задач тестирования знаний является треугольная, трапецеидальная, Z-функция, S-функция фаззификации [3]. Простым и удобным может быть автоматический подбор этих функций на базе контрольного набора специальных тестов. Увеличив число контрольных точек, можно строить функции фаззификации произвольной формы.

Функция принадлежности для лингвистической переменной «экзаменационная оценка» определяется на шкале [0; 100] с нечеткими значениями F={неудовлетворительно, удовлетворительно, хорошо, отлично}. В [3] предлагается лингвистическую переменную «экзаменационная оценка» определять на шкале [0; 200].

Затем на этапе агрегации формируется «график ответов» с учетом коэффициента значимости раздела и вопроса в разделе. В качестве метода дефаззификации может быть использован центроидный метод [3,9,10].

В результате преподавателем создается автоматически шифруемая база вопросов и ответов, которая затем устанавливается в компьютерном классе для тестирования знаний студентов.

С подсистемой проигрывателя тестов работают студенты в процессе тестирования своих знаний. Экспертная система выбирает вопросы из базы тестов случайным образом и предлагает студенту выбрать правильные ответы.

Например, если студент выбрал три правильных ответа из пяти правильных ответов по 100-балльной шкале, то на блок фаззификации подается значение 60 при установленной значимости (важности) вопроса и формируется степень принадлежности к соответствующему терму.

Затем блоком агрегации из всех ответов формируется «график ответов», который подается на блок дефаззификации, который переводит нечеткое значение «графика ответов» в четкий формат. Например, на рис. 5.3 представлен результирующий ответ нечеткой обучающей экспертной системы со значением 4,46 балла.

Одновременно с оценкой 4,46 программой проигрывателя тестов формируется протокол испытаний, содержащий результаты тестирования.

Дальнейшее развитие обучающей нечеткой экспертной системы состоит в создании специальной БЗ обучаемого, что позволит выбирать вопросы для тестирования не случайным образом, а исходя из результатов ответов студента. Для этого можно разработать продукционную БЗ или

использовать нейронную сеть с целью выбора следующего вопроса при тестировании знаний студента.

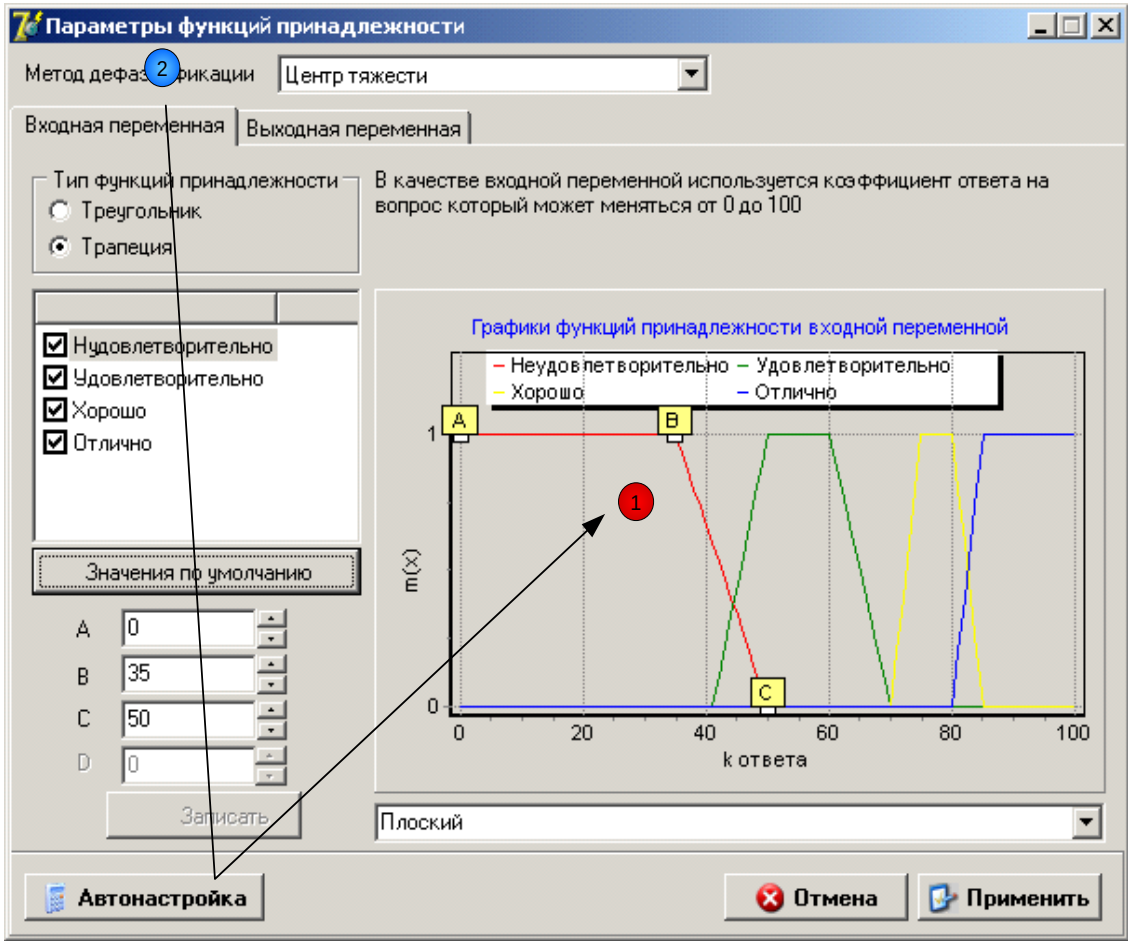


Рис. 5.2. Меню настройки параметров фаззификации и дефаззификации

Развитие обучающей нечеткой экспертной системы можно проводить в следующих направлениях:

- автоматический выбор формы функции фаззификации с помощью генетических алгоритмов;
- автоматический настройка параметров функций фаззификации с помощью генетических алгоритмов;
- автоматический выбор метода дефаззификации.

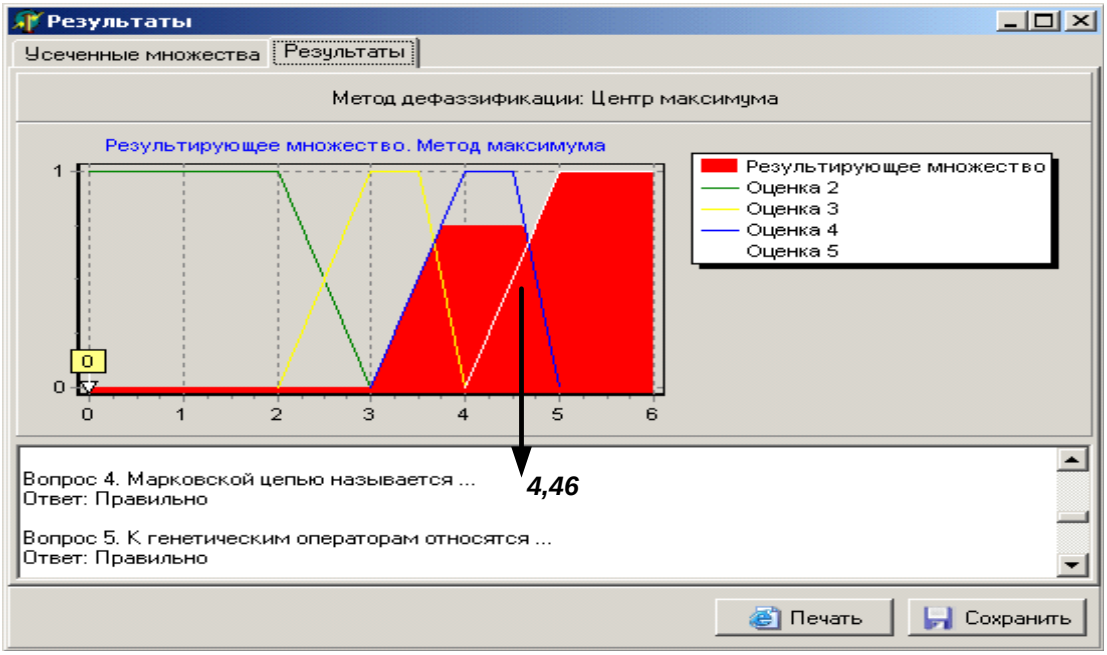


Рис. 5.3. Результаты дефаззификации «графика ответов»

5.3. Подбор параметров нечеткой экспертной системы

При разработке структуры учебной инструментальной экспертной системы нечеткого вывода можно выделить подсистему приобретения знаний и подсистему нечеткого вывода.

Подсистема приобретения знаний должна позволять хранить, создавать, редактировать, сохранять в файл и загружать из файла структуры внутреннего представления данных. Она взаимодействует с экспертом в диалогах ввода имен переменных и термов, их ограничений, диалоговых окон ввода тестовых примеров, коррекции системы, ввода имен файлов.

Подсистема нечеткого вывода должна преобразовывать точные входные значения переменных в точные выходные значения. Точные значения могут быть введены пользователем или считаны из файла. Считывание из файла удобно в случае большого списка переменных, а также большой серии значений для одной или нескольких переменных.

Для хранения настроек системы был выбран формат текстового файла по той причине, что он позволяет пользователю выбрать альтернативные средства его редактирования. Они могут быть удобнее тех, что предоставляет программа, для некоторых областей применения нечеткой логики.

В подсистеме приобретения знаний выделены следующие блоки, приведенные на рис. 5.4:

- блок загрузки нечеткой системы и структуры из текстового файла (взаимодействует с блоком синтаксического разбора и сообщает пользователю, если в файле обнаружены ошибки);
- блок сохранения нечеткой системы и структуры в текстовом файле (обеспечивает сохранение в файле для дальнейшего использования без дополнительных настроек);
- блок синтаксического разбора (производит разбор текста с обнаружением места и вида ошибки и преобразует его во внутреннюю структуру нечеткой системы, если ошибок нет);
- блок редактирования нечеткой системы и структуры экспертом (предоставляет эксперту удобный способ редактирования настроек нечеткой системы);
- блок генерации конфигурации нечеткой системы для логического вывода (автоматически генерирует оптимальную конфигурацию ЭС, а именно параметры входных и выходных функций принадлежности, метод логического вывода, метод дефаззификации).

Подсистема нечеткого вывода состоит из блока подготовки входных данных и сохранения выходных данных (блок использования ЭС конечным пользователем).

Нахождение оптимальной конфигурации экспертной системы. Процесс нахождения оптимальной конфигурации экспертной системы происходит без участия эксперта. Под конфигурацией нечеткой ЭС

понимается выбор параметров функции фаззификации и выбор метода дефаззификации.

Это позволяет освободить эксперта от трудоемкого и утомительного подбора функций принадлежности, метода логического вывода, метода дефаззификации для каждой функции и других сопутствующих проблем.

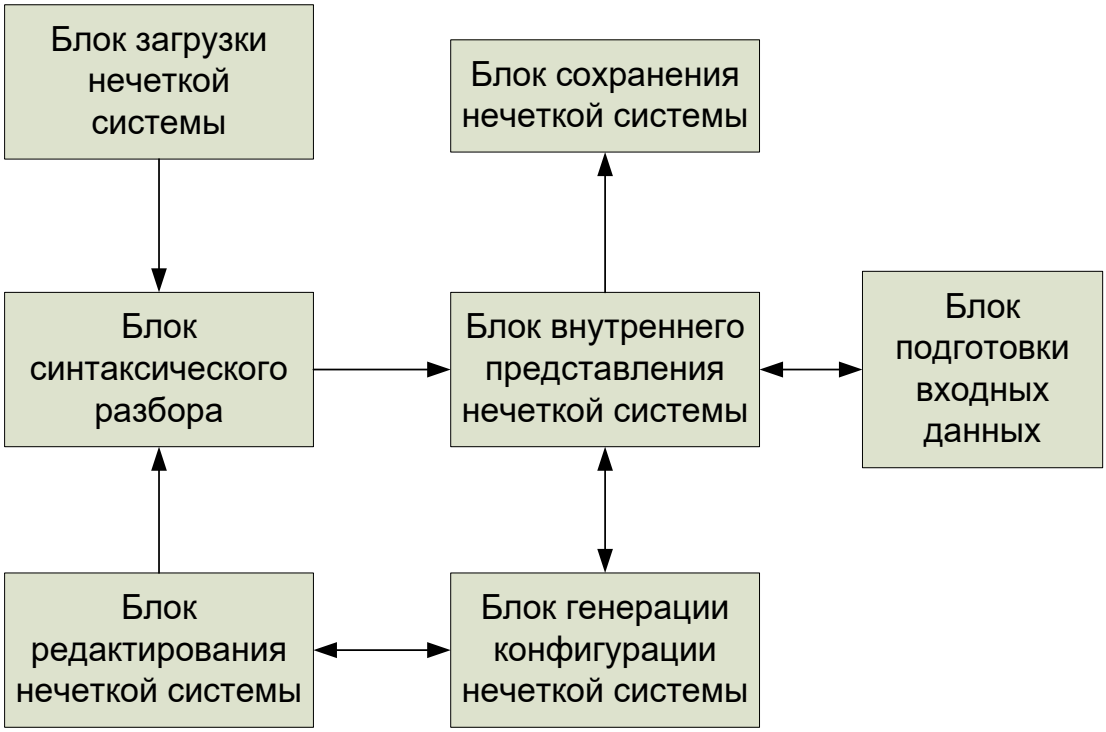


Рис. 5.4. Структура подсистемы приобретения знаний

Автоматизированный подбор снимает с эксперта необходимость изучения принципов и алгоритмов нечетких множеств. Все это в значительной степени снижает трудоемкость разработки экспертной системы. Для подбора оптимальной конфигурации инструментальная система использует данные, полученные от эксперта в процессе диалога.

Подбор необходимой функции фаззификации сводится к определению параметров функции Гаусса. Такой выбор был сделан вследствие того, что функция Гаусса является универсальной, т.е. из нее можно построить любую другую функцию, изменяя ее параметры, в том

числе треугольную и близкую к трапецеидальной функции (рис. 5.5). Например, меняя параметры функции принадлежности Гаусса (*c, σ, b)*, можно сформировать по формуле (5.1) разные виды функций.

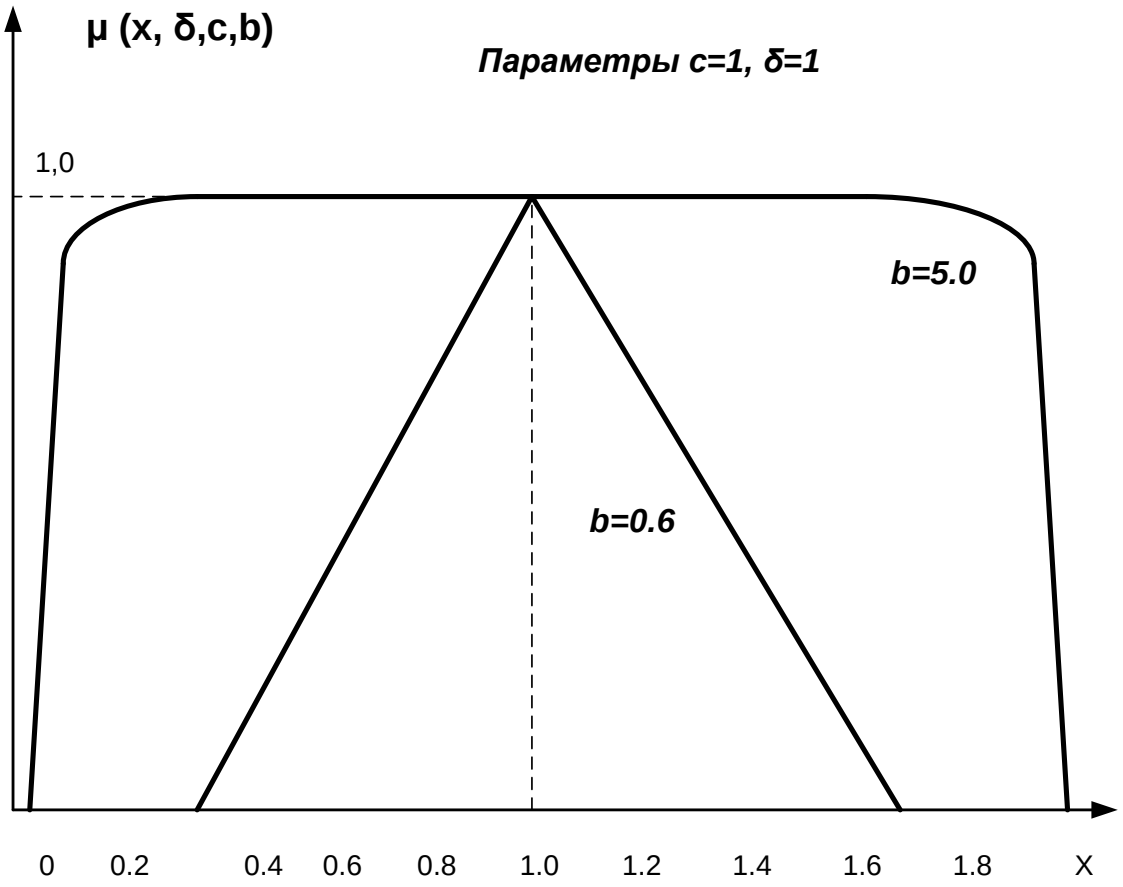


Рис. 5.5. Различные виды функции Гаусса

Это позволяет упростить внутренний алгоритм и уменьшить вероятность возникновения ошибок. Подбор функций принадлежности Гаусса осуществляется на основе тестового множества четких значений по формуле [3, 7, 9].

Параметры данной функции (*c, σ, b)* могут входить в состав особи генетического алгоритма, который подбирает их для конкретных статистических данных, представленных экспертом.

$$\mu_A(x,\delta,c,b) = \exp\left(-\left(\frac{(x-c)}{\delta}\right)^{2b}\right), \qquad (5.1)$$

где μ_A - функция принадлежности;

x - четкое значение, взятое из примера и характеризующее конкретный параметр;

δ, c, b - подбираемые значения функции.

Процесс нахождения оптимальной конфигурации нечеткой ЭС может быть реализован с помощью генетических алгоритмов. Экспертом формируется несколько наборов входных и выходных функций. Выбираются наилучшие сочетания наборов входных и выходных функций по тестовому множеству четких значений, т.е. находятся пары наборов входных и выходных функций, которые наиболее близки к тестовому множеству четких значений.

Набор функций принадлежности с тройкой или четверкой параметров для всех термов лингвистической переменной представляет особь генетического алгоритма. Выбор наилучшей особи производится по каждому тестовому примеру путем вычисления расстояния по формуле (5.2). Это разность между эталонным значением результата, введенного экспертом в тестовом примере, и вычисленным значением по данному набору:

$$\Delta_j = \left| y_j - d_j \right|, \tag{5.2}$$

где d_j – ожидаемое значение, выбранное экспертом;

y_j – полученное значение выходной величины в ходе нечеткого вывода.

Затем вычисляется сумма всех расстояний Δ по формуле

$$\Delta_{общ} = \sum_{j=1}^{n} \Delta_j , \tag{5.3}$$

где n – количество тестовых примеров.

Чем меньше величина $\Delta_{общ}$, тем лучше выбранный набор функций.

Таким образом, из множества наборов выбираются те, у которых значение $\Delta_{общ}$ принимает минимум.

При выборе метода Мамдани [17, 25] используются различные методы дефаззификации: центроидный; первого максимума; среднего максимума последнего максимума и т.п.

В модели вывода Такаги-Сугено-Канга [17, 25] нахождение оптимальной конфигурации нечеткой ЭС может быть выполнено с использованием генетического алгоритма, с помощью которого формируется особь из набора коэффициентов p_{ij}, определяемых системой линейных уравнений

$$\begin{aligned} y_1 &= p_{10} + p_{11} \cdot x_1 + p_{12} \cdot x_2 + \ldots + p_{1n} \cdot x_n \\ y_2 &= p_{20} + p_{21} \cdot x_1 + p_{22} \cdot x_2 + \ldots + p_{2n} \cdot x_n \\ &\ldots \\ y_m &= p_{m0} + p_{m1} \cdot x_1 + p_{m2} \cdot x_2 + \ldots + p_{mn} \cdot x_n \end{aligned} \qquad (5.4)$$

В процессе отладки нечеткой системы эксперту предлагается протестировать полученную систему на своих примерах: эксперт вводит входные параметры и предполагаемый выходной результат. Если расхождения между собственными оценками и оценками системы не устраивают эксперта, то инженер знаний корректирует параметры нечеткой системы и предлагает ему вновь протестировать модифицированную ЭС.

Создание нечеткой экспертной системы состоит из следующих этапов:

- определение критериев и градаций (лингвистических переменных и их термов);
- создание списка правил вывода;

- заполнение сгенерированных тестовых примеров отладка системы;
- сопровождение системы, которое включает модификацию уже разработанной системы под изменившуюся задачу, добавление и изменение списка правил.

Контрольные вопросы

1. Основные характеристики инструментальных средств создания нечеткой системы.
2. Характеристика экспертной системы ExSys.
3. Пример базы знаний нечеткой ЭС.
4. Этапы создания нечеткой ЭС.
5. Понятие терма лингвистической переменной.
6. Пример лингвистической переменной «оценка».
7. Функции фаззификации и их параметры.
8. Методы дефаззификации.
9. Параметры функции Гаусса, определяющие ее форму.
10. Принцип автоматического подбора параметров нечеткой системы на базе генетических алгоритмов.

ГИБРИДНЫЕ ЭКСПЕРТНЫЕ СИСТЕМЫ

6.1. Представление знаний в мягкой экспертной системе

Американским ученым Лотфи Заде (Lotfi Zadeh) в 1994 году введен термин «мягкие вычисления», представляющий сложную компьютерную методологию [25], компонентами которой являются:

– нечеткая логика (приближенные вычисления, грануляция информации, вычисление на словах);
– нейрокомпьютинг (обучение, адаптация, классификация, системное моделирование и идентификация);
– генетические вычисления (синтез, настройка и оптимизация с помощью систематизированного случайного поиска и эволюции);
– вероятностные вычисления (управление неопределенностью, сети доверия, хаотические системы, предсказание).

Интерпретация «мягких вычислений» может быть следующая:

мягкие вычисления = нечеткие системы + нейронные сети + +генетические алгоритмы.

Свойства мягкой интеллектуальной системы обеспечиваются ее компонентами [25]:

мягкая интеллектуальная система (МЭС) = управление неопределенностью + обучаемость + самоадаптация.

Мягкие системы, такие, как нечеткие нейронные сети с генетической настройкой параметров, демонстрируют взаимное усиление достоинств и нивелирование недостатков отдельных методов [17, 21, 25].

Нечеткие экспертные системы используют представление знаний в форме нечетких продукций и лингвистических переменных. Каждый терм лингвистической переменной определяется функцией принадлежности. Способ обработки знаний в нечеткой ЭС – это логический вывод на нечетких продукциях. Особенностью нечеткой ЭС является способ построения функций принадлежности, который сводится либо к статистическим методам построения, либо к методу экспертных оценок.

Мягкая интеллектуальная система обладает следующими особенностями:

– использует статистические данные, которые интерпретируются как обучающие выборки для нечетких нейронных сетей;

– представляет знания в виде лингвистических переменных (функций принадлежности), нечетких продукций и обученных нейронных сетей. Редукция множества нечетких продукций, настройка и подбор параметров функций принадлежности выполняется с помощью генетических алгоритмов.

Мягкая экспертная система должна представлять инструментальную среду для экспертной деятельности [3, 9], содержащую совокупность различных программ с общей логикой работы. Например, если рассматривать управление объектом, то инструментарий можно рассматривать как нечеткий контроллер (рис. 6.1).

Нечеткая нейронная сеть представляет собой средство для извлечения знаний для МЭС. Она извлекает знания из статистических выборок, интерпретируя их как обучающие выборки для обучения нейронной сети.

В [9] приведён пример извлечения нечетких знаний для ЭС анализа тенденций развития предприятия и для задачи конструирования стендов изделий.

Представление знаний в МЭС. Если использовать нечеткие нейронные сети на этапе извлечения знаний, то, кроме функций принадлежности и нечетких продукций, порождается совокупность обученных НС, которые входят в базу знаний МЭС. Оптимизация (редукция) множества извлеченных правил выполняется на основе генетического алгоритма.

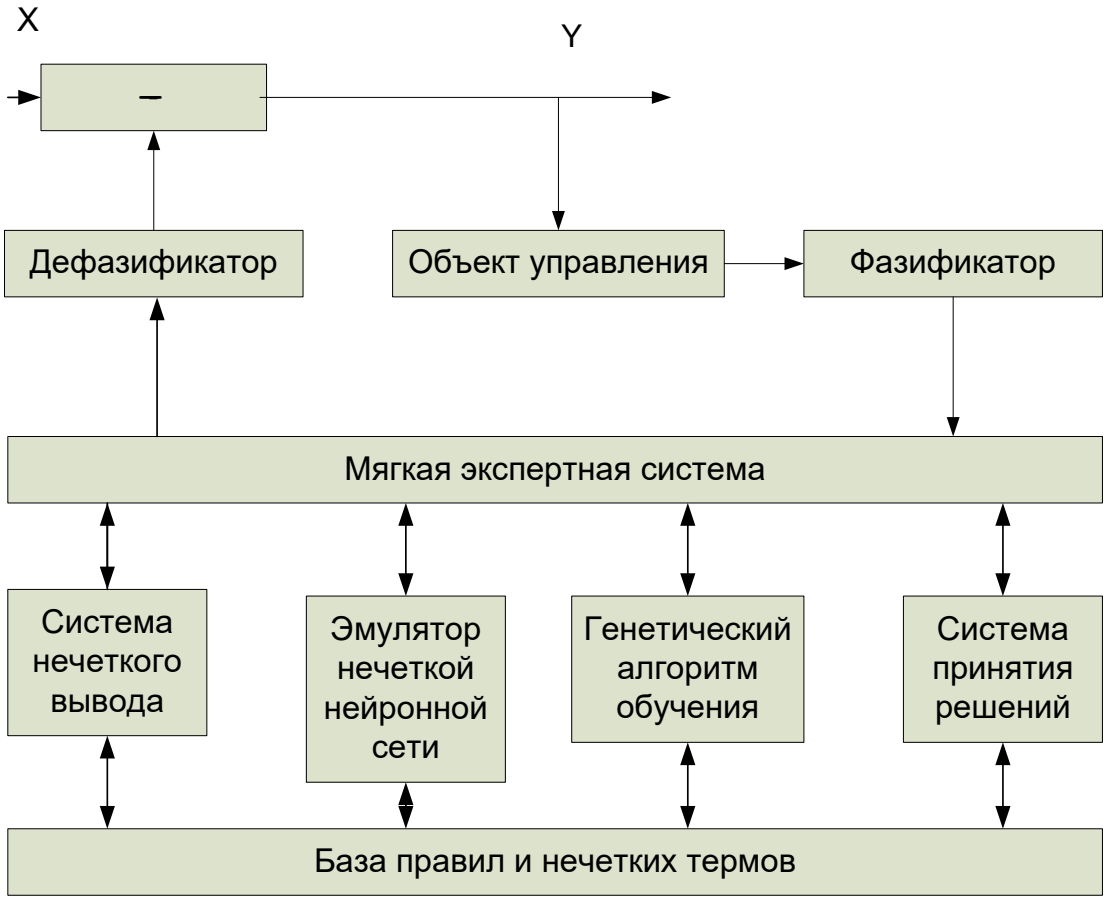


Рис. 6.1. Структурная схема мягкой экспертной системы

База знаний МЭС содержит следующие компоненты:

– функции принадлежности;

– нечеткие продукции;

– обученные нечеткие нейронные сети;

– процедуры интерпретации хромосом генетического алгоритма;

– критерий оптимальности.

Функция принадлежности может быть представлена в виде треугольной, трапециевидной, колоколообразной и сигмоидальной функцией. Представление нечетких продукций упрощается, так как порядок их обработки не влияет на ход вывода результата. Представление нечеткой нейронной сети является более сложной задачей. Для организации хранения знаний МЭС можно использовать как системы управления базами данных, так и специальные форматы.

По мнению автора настоящего учебного пособия, целесообразно рассмотренные мягкие ЭС [25] относить к классу гибридных экспертных систем по следующим соображениям:

- гибридные ЭС является более широким понятием по сравнению с мягкими ЭС;
- в гибридных ЭС применяются различные информационные технологии (нечеткие множества, нейронные сети и генетические алгоритмы);
- в научно-технической литературе принято понятие гибридной ЭС [9].

6.2. Пример гибридной экспертной системы

В основе гибридной экспертной системы, разработанной на кафедре ЭВМ ВятГУ, использована гибридная нейронная сеть (ГНС), структура которой представлена на рис. 6.2.

Гибридная нейронная сеть состоит из двух основных слоев: самоорганизующегося нечеткого слоя и многослойной сигмоидальной нейронной сети (НС), называемой многослойным персептроном. Самоорганизующийся нечеткий слой реализуется блоком кластеризации.
При обучении данной сети обучающей выборкой (x1,…, xN, y1,…, yM) вначале производится организация нечеткого слоя входных переменных (x1,…, xN) алгоритмом горной кластеризации [8, 17, 31]. В результате набор векторов (x1,…, xN) преобразуется в набор параметров, характеризующих принадлежность входного вектора кластерам (u1,…, uK). Причем значение параметра K должно быть меньше числа векторов в наборе. Далее персептрон обучается на выборке (u1,…, uK, y1,…, yM).

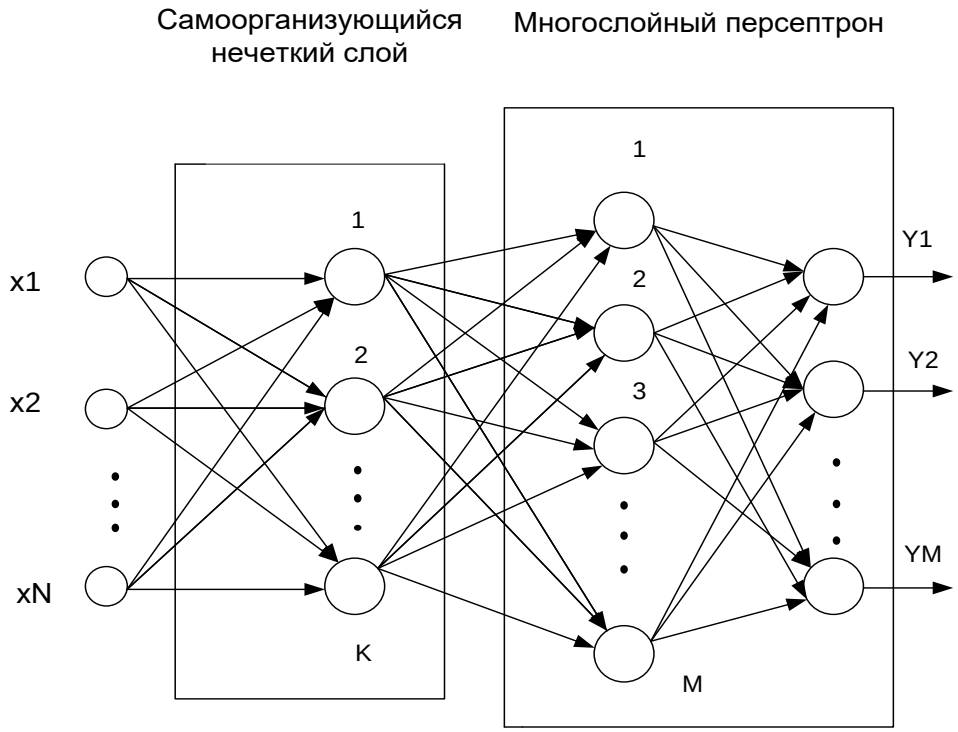


Рис. 6.2. Структура гибридной нейронной сети

Инструментальная ГЭС работает в двух режимах: режим обучения и режим предсказания.

Структура пользовательского интерфейса учебной гибридной экспертной системы (разработки кафедры ЭВМ ВятГУ) представлена на рис. 6.3.

В процессе работы студенту предлагается последовательно выполнить три шага по настройке ГЭС на конкретную предметную область: задание обучающей выборки, кластеризация данных, создание и обучение нейронной сети.

После запуска программы возможно создание нового проекта либо открытие завершенного или незавершенного сохраненного проекта из файла с расширением *.hnn.

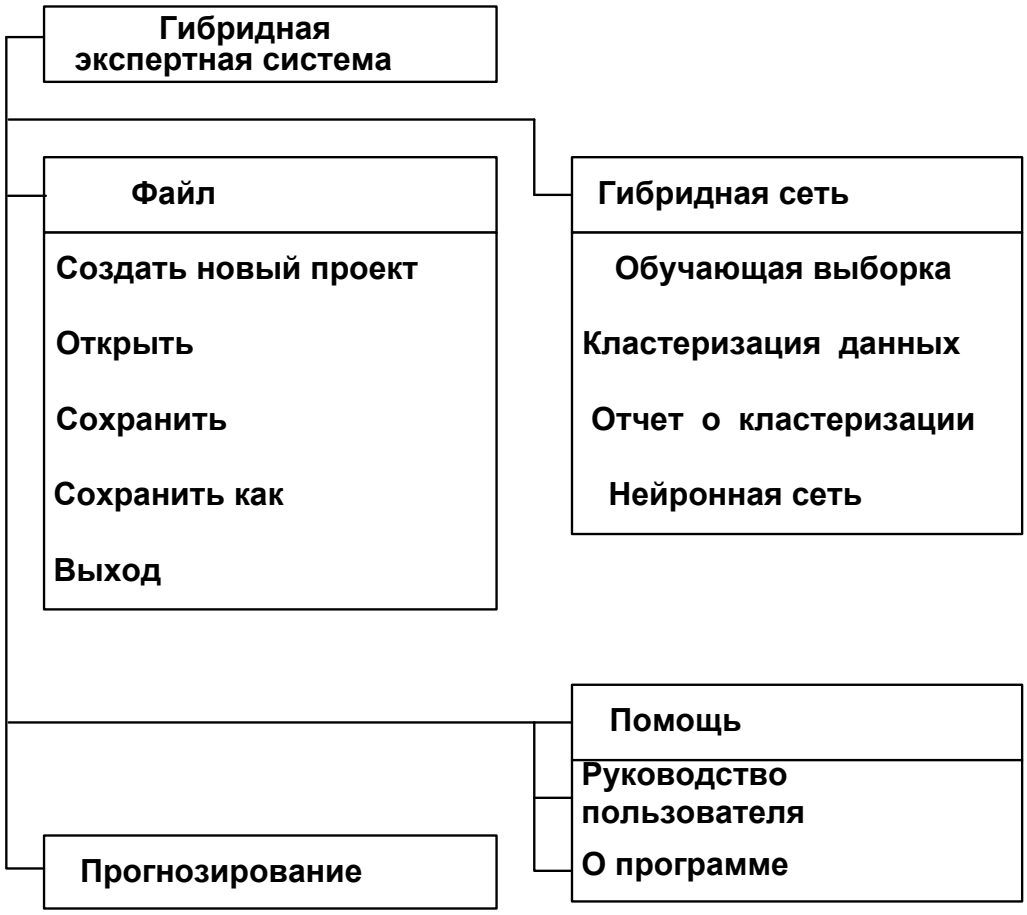


Рис. 6.3. Структура пользовательского интерфейса

Создание проекта начинается с формирования обучающей выборки (ОВ). Для создания нового проекта необходимо выбрать пункт меню «Файл» / «Создать новый проект». В появившемся окне «Задание обучающей выборки» щелкнуть по соответствующей кнопке «Загрузить из файла». Файл обучающей выборки с расширением *.txt создается в программе БЛОКНОТ, с расширением *.xls – редакторе MS Excel.

Форма окна задания обучающей выборки всегда может быть вызвана путем выбора пункта «Обучающая выборка» в меню «Гибридная сеть». После загрузки ОВ нужно щелкнуть по кнопке «Задать назначение полей» и проверить входные и выходные поля. Далее при необходимости следует упростить выборку, изменяя коэффициент степени упрощения и анализируя значение константы Липшица. На этом работа с обучающей выборкой завершается.

При нажатии кнопки «Далее» автоматически открывается окно «Кластеризация данных» (рис. 6.4), в котором необходимо задать необходимые параметры. Параметр кластеризации «σ» подбирается в соответствии с рекомендациями. Для выполнения кластеризации служит кнопка «Выполнить кластеризацию» [9].

При нажатии кнопки «Далее» откроется окно «Настройка параметров сети» (рис. 6.5), в котором устанавливаются параметры нейронной сети (количество слоев и нейронов в каждом слое, параметры активационной функции, допустимая ошибка обучения).

Форма позволяет создать сеть (или несколько сетей) с помощью кнопки «Создать сеть», задать число слоев нейронов, число нейронов в каждом слое, тип и параметр передаточной функции в каждом слое, выполнить обучение сети с помощью кнопки «Обучение». Обучение сети может быть прервано с помощью кнопки «Прервать». В правой части формы изображена таблица обучающих примеров, каждый из которых может быть использован или не использован при обучении сети. Чтобы установить или отменить использование примера для обучения сети, против данного примера в таблице необходимо соответственно установить или снять метку с помощью мыши. Вычисление ошибки обобщения определяется по примерам, которые в обучении не использовались. Возможно создание нескольких сетей и выбор текущей из них по критерию минимальной ошибки обобщения.

Имеется возможность вернуться к любому из трех шагов создания проекта с помощью пунктов из раздела меню «Гибридная сеть».

Выбор пункта меню «Прогнозирование» вызывает открытие формы, позволяющей вводить в виде таблицы произвольные значения входных параметров и получать выходные.

Проект на любой стадии создания может быть сохранен в файл с расширением .hnn с помощью пункта меню «Файл» / «Сохранить» или «Файл» / «Сохранить как».

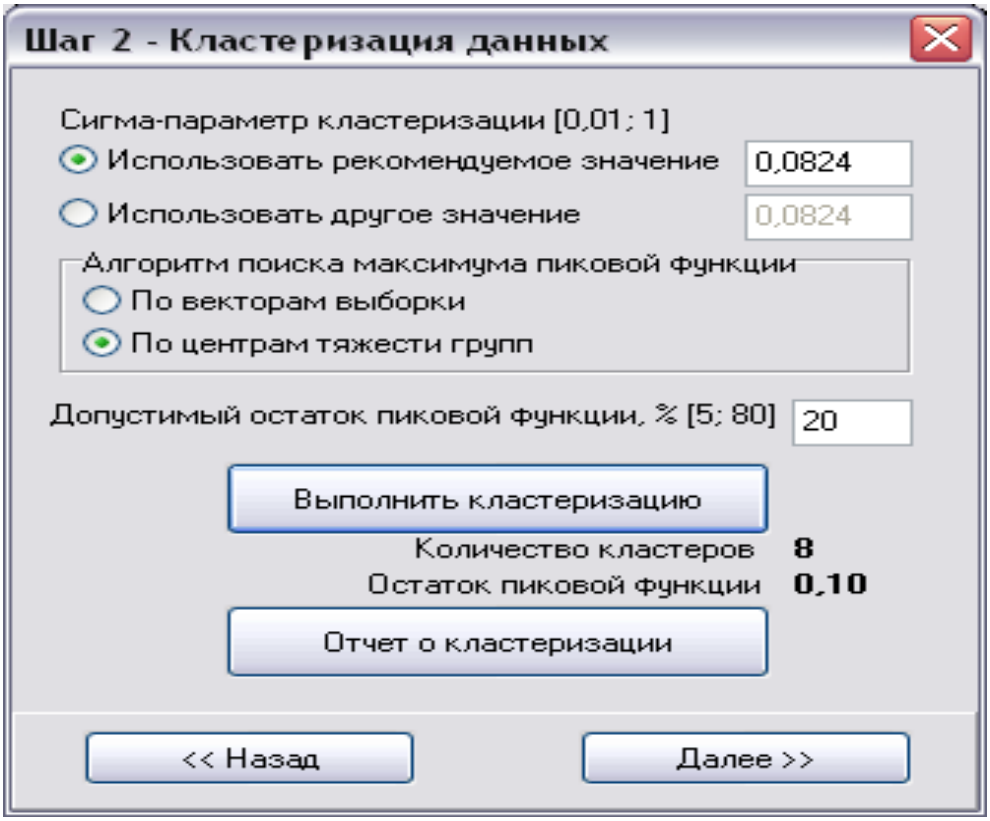


Рис. 6.4. Окно «Кластеризация данных»

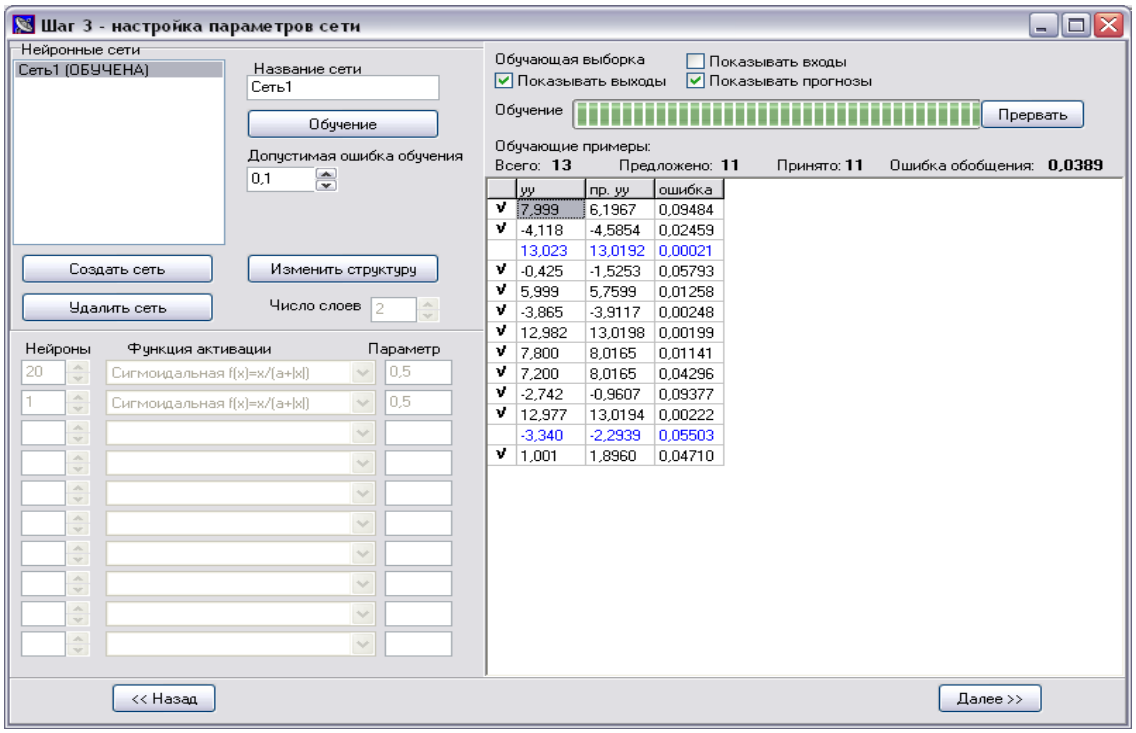


Рис. 6.5. Окно «Настройка параметров нейронной сети»

6.3. Основные понятия теории генетических алгоритмов

Генетические алгоритмы представляют собой алгоритмы поиска, построенные на принципах, сходных с принципами естественного отбора и генетики. Они имеют целью нахождение лучшего, а не оптимального решения задачи. Это связанно с тем, что для сложной системы часто требуется найти удовлетворительное решение, а проблема достижения оптимума отходит на второй план. При этом другие методы, ориентированные на поиск именно оптимального решения, вследствие чрезвычайной сложности задачи становятся вообще неприменимыми. В этом кроется причина появления, развития и роста популярности генетических алгоритмов.

Преимущества генетических алгоритмов становятся еще более прозрачным, если рассмотреть основные их отличия от традиционных методов. Основных отличий четыре [1].

1. Генетические алгоритмы работают с кодами, в которых представлен набор параметров, напрямую зависящих от аргументов целевой функции. Причем интерпретация этих кодов происходит только перед началом работы алгоритма и после завершения его работы для получения результата. В процессе работы манипуляции с кодами происходят совершенно независимо от их интерпретации. Код рассматривается просто как битовая строка или строка символов.

2. Для поиска генетический алгоритм использует несколько точек поискового пространства одновременно, а не переходит от точки к точке, как это делается в традиционных методах. Это позволяет преодолеть один из их недостатков – опасность попадания в локальный экстремум целевой функции, если она не является унимодальной, т.е. имеет несколько таких экстремумов. Использование нескольких точек одновременно значительно снижает такую возможность.

3. Генетические алгоритмы в процессе работы не используют никакой дополнительной информации, что повышает скорость работы. Единственной используемой информацией может быть область допустимых значений параметров и целевой функции в произвольной точке.

4. Генетический алгоритм использует как вероятностные правила для порождения новых точек анализа, так и детерминированные правила для перехода от одних точек к другим. Выше уже говорилось, что одновременное использование элементов случайности и детерминированности дает значительно больший эффект, чем раздельное.

Прежде чем рассматривать непосредственно работу генетического алгоритма, вводится ряд терминов, используемых в данной области.

Ранее было показано, что генетический алгоритм работает с кодами безотносительно их смысловой интерпретации. Поэтому сам код и его структура описываются понятием генотипа и представляют, по сути, точку пространства поиска. С целью максимального приближения к биологическим терминам экземпляр кода называют хромосомой, особью или индивидуумом. Далее для обозначения строки будет использоваться термин «особь».

На каждом шаге работы генетический алгоритм использует несколько точек поиска одновременно. Совокупность этих точек является набором особей, который называется популяцией. Количество особей в популяции называют размером популяции. Поскольку в данном разделе рассматриваются классические генетические алгоритмы, то можно сказать, что размер популяции является фиксированным и представляет одну из характеристик генетического алгоритма. На каждом шаге работы генетический алгоритм обновляет популяцию путем создания новых особей и уничтожения старых. Чтобы отличать популяцию на каждом из шагов и сами эти шаги, их называют поколениями и обычно идентифицируют по номеру. Например, популяция, полученная из исходной популяции после

первого шага работы алгоритма, будет первым поколением, после следующего шага - вторым, и т. д.

В процессе работы алгоритма генерация новых особей происходит на основе моделирования процесса размножения. При этом, естественно, порождающие особи называют родителями, а порожденные – потомками. Родительская пара, как правило, порождает пару потомков. Непосредственная генерация новых кодовых строк из двух выбранных происходит за счет работы оператора скрещивания, который также называют кроссинговером (от англ. crossover). При порождении новой популяции оператор скрещивания может применяться и ко всем парам родителей. Часть этих пар может переходить в популяцию следующего поколения непосредственно. Насколько часто будет возникать такая ситуация, зависит от значения вероятности применения оператора скрещивания, который является одним из основных параметров генетического алгоритма.

Моделирование процесса мутации новых особей осуществляется за счет работы оператора мутации. Основным параметром оператора мутации также является вероятность мутации.

Поскольку размер популяции фиксирован, то порождение потомков должно сопровождаться уничтожением других особей. Выбор пар родителей из популяции для порождения потомков производится оператором отбора, а выбор особей для уничтожения – оператором редукции.

К характеристикам генетического алгоритма относятся:

- размер популяции;
- оператор скрещивания и вероятность его использования;
- оператор мутации и вероятность мутации;
- оператор отбора;
- оператор редукции;

– критерий останова.

Операторы отбора, скрещивания, мутации и редукции называются *генетическими операторами.*

Критерием останова работы генетического алгоритма может быть одно из трех событий:

1. Сформировано заданное пользователем число поколений;
2. Популяция достигла заданного пользователем качества (например, значение качества всех особей превысило заданный порог);
3. Достигнут некоторый уровень сходимости. Особи в популяции стали настолько подобными, что дальнейшее их улучшение происходит чрезвычайно медленно.

Характеристики генетического алгоритма выбираются таким образом, чтобы обеспечить малое время работы, с одной стороны, и поиск как можно лучшего решения, с другой.

Схема работы генетического алгоритма приведена на рис. 6.6.

Создание исходной популяции происходит, как правило, с использованием какого-либо случайного закона, на основе которого выбирается нужное количество точек поискового пространства. Исходная популяция может быть сформирована другой программой.

Выбор особи для размножения. Вероятность участия особи в процессе размножения вычисляется по формуле

$$P_i = F_i / \sum_{i=1}^{n} F_i , \qquad (6.1)$$

где n – размер популяции;

i – номер особи;

P_i – вероятность участия i-й особи в процессе размножения;

F_i – значение целевой функции для i-й особи.

Очевидно, что одна особь может быть задействована в нескольких родительских парах. В основе оператора отбора, который служит для выбора родительских пар, лежит принцип “выживает сильнейший”.

Аналогично может быть решен вопрос уничтожения особей. Только вероятность уничтожения, соответственно, должна быть обратно пропорциональна качеству особей.

Таким образом, выбирая для размножения наиболее качественные особи и уничтожая наиболее слабые, генетический алгоритм постоянно улучшает популяцию, приводя к нахождению все лучших решений.

Оператор скрещивания призван моделировать природный процесс наследования, обеспечивая передачу свойств родителей потомкам.

Процесс скрещивания выполняется в два этапа. Например, особь представляет собой строку из n элементов. На первом этапе равновероятно выбирается число k от 1 до n-1, которое называется точкой разбиения. В соответствии с ним обе исходные строки разбиваются на две подстроки. На втором этапе строки обмениваются своими подстроками, лежащими после точки разбиения, т.е. элементами с k+1-го по n-й. В результате формируются две новые строки, которые наследовали частично свойства обоих родителей. Этот процесс проиллюстрирован следующим образом:

Строка 1	X1X2. . .XkXk+1. . .Xn	X1X2. . .XkYk+1. . .Yn
	111111111 k 111111111	**111111111000000000**
Строка 2	Y1Y2. . .YkYk+1. . .Yk	Y1Y2. . .YkXk+1. . .Xn
	000000000 k 000000000	*000000000111111111*

Вероятность применения оператора скрещивания обычно выбирается достаточно большой, в пределах от 0,8 до 1, чтобы обеспечить постоянное появление новых особей, расширяющих пространство поиска. При значении вероятности менее единицы часто используют *элитизм*. Это особая стратегия, которая предполагает переход в популяцию следующего поколения элитных особей, т.е. лучших особей текущей популяции, без всяких изменений.

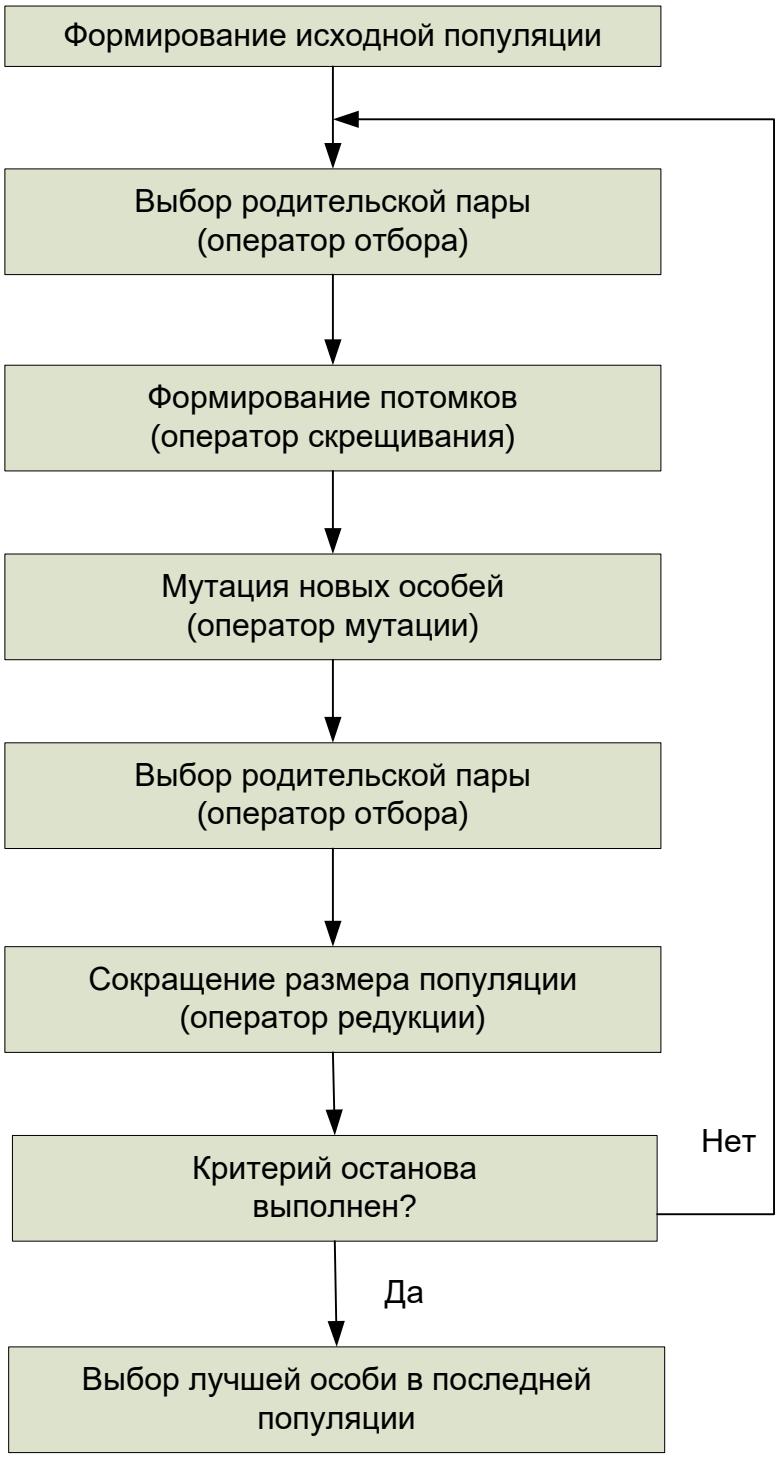


Рис. 6.6. Схема работы генетического алгоритма

Применение элитизма способствует сохранению общего качества популяции на высоком уровне. При этом элитные особи участвуют еще и в процессе отбора родителей для последующего скрещивания. Количество элитных особей, переходящих в следующее поколение, определяется по формуле

$$K=(1-P) \cdot N, \qquad (6.2)$$

где К - количество элитных особей;

Р – вероятность применения оператора скрещивания;

N - размер популяции.

Оператор мутации служит для моделирования природного процесса мутации. Его применение в генетических алгоритмах обусловлено следующими соображениями. Исходная популяция, какой бы большой она ни была, охватывает ограниченную область пространства поиска. Оператор скрещивания, безусловно, расширяет эту область, но все же до определенной степени, поскольку использует ограниченный набор знаний, заданный исходной популяцией. Внесение случайных изменений в особи позволяет преодолеть это ограничение и иногда значительно сократить время поиска или улучшить качество результата.

Как правило, вероятность мутации, в отличие от вероятности скрещивания, выбирается достаточно малой. Сам процесс мутации заключается в замене одного из элементов строки на другое значение. Это может быть перестановка двух элементов в строке, замена элемента строки значением элемента из другой строки, а в случае битовой строки может применяться инверсия одного из битов и т. д.

В процессе работы алгоритма все указанные выше операторы применяются многократно и ведут к постепенному изменению исходной популяции. Поскольку операторы отбора, скрещивания, мутации и редукции по своей сути направлены на улучшения каждой отдельной особи, то результатом их работы является постепенное улучшение популяции. В этом и заключается основной смысл работы генетического алгоритма - улучшить популяцию решений по сравнению с исходной популяцией.

После завершения работы генетического алгоритма из конечной популяции выбирается та особь, которая дает максимальное (минимальное) значение целевой функции. Эта особь является результатом работы генетического алгоритма.

Пример поиска максимума одномерной функции. Пусть имеется набор натуральных чисел от 1 до 31 и функция, определенная на этом наборе чисел, f(x)=x. Требуется найти максимальное значение функции [1].
В качестве кода используется двоичное представление аргументов функции. Это положение представляет собой фенотип нашего алгоритма. Сам код будет представлять собой двоичную строку из 5 бит. Это генотип алгоритма. Целевой функцией будет непосредственно сама рассматриваемая функция, аргументом которой является число, чье двоичное представление использует алгоритм.

Принятые характеристики генетического алгоритма:

- размер популяции 4;
- вероятность мутации 0,001,
- процесс мутации заключается в инверсии одного из битов строки, выбираемого случайно по равномерному закону;
- оператор скрещивания и отбора аналогичны описанным ранее;
- элитизм не используется.

Пусть случайным образом создана исходная популяция из четырех особей, представленная табл. 6.1.

Предположим, что оператор отбора выбрал для производства потомков две пары строк (1, 2) и (2, 4). Работа оператора скрещивания проиллюстрирована в табл. 6.2. При этом в каждой паре разбиение на подстроки происходит независимо. Точка разбиения задана звездочкой.

Пропорциональный простейший отбор (рулетка) выбирает n особей после n запусков. Колесо рулетки содержит по одному сектору для каждого гена популяции. Размер сектора пропорционален вероятности участия.

Таблица 6.1

Исходная популяция

Номер строки	Код генотипа	Значение целевой функции	Вероятность участия в размножении
1	01011	11	11/43
2	10010	18	18/43
3	00010	2	2/43
4	01100	12	12/43
		Сумма 43	

Таблица 6.2

Формирование потомков

Номер строки	Родители	Потомки	Значение целевой функции для потомков
1	0*1011	00010	2
2	1*0010	11011	27
3	100*10	10000	16
4	011*00	01110	14

Пусть оператор мутации, несмотря на низкую вероятность, срабатывает для младшего бита потомка в строке 3 и данный код изменяет свое значение с 10000 на 10001.

Таким образом, популяция за счет порожденных потомков расширяется до восьми особей, представленных табл. 6.3.

Оператор редукции далее сокращает популяцию до исходного числа особей, исключив из нее особи с минимальным значением целевой

функции. В результате исключаются строки 1, 3, 4, 5, и популяция первого поколения принимает вид, представленный табл. 6.4.

На этом шаг работы генетического алгоритма закончится. Очевидно, что даже за эту одну итерацию качество популяции значительно возросло.

Если в исходной популяции среднее значение целевой функции было 10, 75, а ее минимальное значение составляло 2, то в популяции первого поколения среднее значение увеличилось до 19, а минимальное значение составило 14. Лучшее же решение увеличилось с 18 до 27 при оптимальном решении 31.

Таблица 6.3

Расширенная популяция

№ строки	Код генотипа	Значение целевой функции
Исходная популяция		
1	01011	11
2	10010	18
3	00010	2
4	01100	12
Порожденные потомки		
5	00010	2
6	11011	27
7	10001	17
8	01110	14

Таблица 6.4

Популяция первого поколения

Номер строки	Код генотипа	Значение целевой функции	Вероятность участия в процессе размножения
1	10010	18	18/ 76
2	11011	27	27/ 76
3	10001	17	17/ 76
4	01110	14	14/ 76
		Сумма 76	

Таким образом, данный пример работы генетического алгоритма наглядно иллюстрирует процесс улучшения как популяции в целом, так и поиск наилучшего решения [1]. После первого шага работы генетического алгоритма максимальное значение увеличилось с 18 до 27 при оптимальном значении 31.

Контрольные вопросы

1. Понятия мягких вычислений и мягкой экспертной системы
2. Понятие гибридной экспертной системы.
3. Структура гибридной экспертной системы.
4. Структура мягкой экспертной системы.
5. База знаний мягкой экспертной системы.
6. Преимущества генетических алгоритмов.
7. Основные термины генетических алгоритмов.
8. Этапы выполнения генетического алгоритма.
9. Применение стратегии элитизма в генетических алгоритмах.
10. Принцип работы оператора отбора.
11. Принцип работы оператора скрещивания.
12. Принцип работы оператора мутации.

13. Принцип работы оператора редукции.
14. Результат работы генетического алгоритма.

.Онтологии

7.1. Общие сведения об онтологии

Онтоло́гия в информатике (новолат. ontologia от др.-греч. ὤν род. п. ὄντος — сущее, то, что существует и λόγος — учение, наука) — это попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области. Этот термин в информатике является производным от древнего философского понятия «онтология» [13-17].

Онтологии используются в процессе программирования как форма представления знаний о реальном мире или его части. Основные сферы применения — моделирование бизнес-процессов, семантическая паутина (англ. Semantic Web), искусственный интеллект.

Термин «онтология» изначально философский, в информатике он принял самостоятельное значение. Здесь есть два существенных отличия:

– онтология в информатике должна иметь формат, который компьютер сможет легко обработать;

– информационные онтологии создаются всегда с конкретными целями — решения конструкторских задач; они оцениваются больше с точки зрения применимости, чем полноты.

Онтологии вышли за рамки библиотечной науки, философии и представления знаний. Сейчас их беспокоят отделы маркетинга, генеральные директора, основной бизнес, исследовательские аналитические компании, такие как Forrester Research.

В настоящее время рассматриваются онтологии, используемые в качестве центральных контролируемых словарей, которые интегрированы в каталоги, базы данных, веб-публикации, приложения для управления знаниями и т. Д. Большие онтологии являются важными компонентами во многих онлайн-

приложениях, включая поиск (например, Yahoo и Lycos), электронную коммерцию как Amazon и eBay), конфигурации (например, Dell и PC-Order) и т. д. Также есть онтологии, которые имеют длительный срок службы, иногда в нескольких проектах (таких как UMLS, SIC-коды и т.д.). Такое разнообразное использование создает много последствий для онтологических сред[1].

Современные онтологии строятся по большей части одинаково, независимо от языка написания. Обычно они состоят из экземпляров, понятий, атрибутов и отношений [13-17].

Экземпляры (англ. *instances*) или индивиды (англ. *individuals*) — это объекты, основные нижнеуровневые компоненты онтологии; могут представлять собой как физические объекты (люди, дома, планеты), так и абстрактные (числа, слова). Строго говоря, онтология может обойтись и без конкретных объектов, однако, одной из главных целей онтологии является *классификация* таких объектов, поэтому они также включаются.

Понятия (англ. concepts) или ***классы*** (англ. classes) — абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы, либо же сочетания и того, и другого. Пример: понятие «люди», вложенное понятие «человек». Чем является «человек» — вложенным понятием, или экземпляром (индивидом) — зависит от онтологии.

Понятие «индивиды», экземпляр «индивид».

Классы онтологии составляют таксономию — иерархию понятий по отношению вложения.

Объекты в онтологии могут иметь атрибуты. Каждый атрибут имеет по крайней мере имя и значение и используется для хранения информации, которая специфична для объекта и привязана к нему. Например, объект Автомобиль-модели-А имеет такие атрибуты, как:

Название: Автомобиль-модели-А

[1] http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-abstract.html

Число-дверей: 4

Двигатель: {2.0л, 2.6л}

Коробка-передач: 6-ступенчатая

Значение атрибута может быть сложным типом данных. В данном примере значение атрибута, который называется Двигатель, является списком значений простых типов данных.

Важная роль атрибутов заключается в том, чтобы определять ***отношения*** (зависимости) между объектами онтологии. Обычно отношением является атрибут, значением которого является другой объект.

Предположим, что в онтологии автомобилей присутствует два объекта — автомобиль Автомобиль-модели-А и Автомобиль-модели-Б. Пусть Автомобиль-модели-Б это модель-наследник Автомобиль-модели-А, тогда отношение между Автомобиль-модели-А и Автомобиль-модели-Б определим, как атрибут «isSuccessorOf» со значением «Автомобиль-модели-А» для объекта Автомобиль-модели-Б (следует заметить, что в языках описания онтологий существуют предопределенные отношения наследования).

Специализированные (***предметно-ориентированные***) онтологии — это представление какой-либо области знаний или части реального мира. В такой онтологии содержатся специальные для этой области значения терминов. К примеру, слово «поле» в сельском хозяйстве означает участок земли, в физике — один из видов материи, в математике — класс алгебраических систем.

Общие онтологии используются для представления понятий, общих для большого числа областей. Такие онтологии содержат базовый набор терминов, глоссарий или тезаурус, используемый для описания терминов предметных областей.

Если использующая специализированные онтологии система развивается, то может потребоваться их объединение. Подзадачей объединения онтологий является задача отображения онтологий.

Онтологии даже близких областей могут быть несовместимы друг с другом. Разница может появляться из-за особенностей местной культуры, идеологии или вследствие использования другого языка описания. Объединение онтологий выполняют как вручную, так и в полуавтоматическом режиме. В целом это — трудоёмкий, медленный и дорогостоящий процесс. Использование базисной онтологии — единого глоссария — несколько упрощает эту работу. Есть научные работы по технологиям объединения, но они по большей части теоретические.

Практически, создание онтологий включает следующие этапы:

1. Определение понятий (классов) в онтологии
2. Организация понятий (классов) в некоторую иерархию (базовый класс –> подкласс)
3. Определение слотов и их допустимых значений
4. Заполнение значений слотов для экземпляров классов.

Идея Грубера состояла в том, чтобы позволить интеллектуальным системам обмениваться между собой заложенными в них знаниями, прежде всего декларативными. Если внутри интеллектуальной системы знания о мире могут быть закодированы как угодно, то для обмена этими знаниями с другой интеллектуальной системой необходимо предоставить описание этих знаний. Это описание должно быть в достаточной степени формальным, чтобы быть понятным другой системе, а также должен быть известен язык этого описания. Кроме того, описание должно быть понятно также и человеку. Для этого Грубер предложил описывать знания двумя способами:

– в канонической форме, которая представляет собой описание знаний на языке логики предикатов (например, в виде фактов языка Prolog);

– в форме онтологии, которая представляет собой множество классов, связанных между собой отношением обобщения (это обратное отношение для отношения наследования).

Онтология по Груберу представляет собой описание ***декларативных знаний***, сделанное в виде классов с отношением иерархии между ними. К этому описанию, предназначенному для чтения человеком, присоединено описание в канонической форме, которое предназначено для чтения машинами. Каждая интеллектуальная система может предоставлять несколько таких описаний, соответствующих различным областям хранящихся в ней декларативных знаний и, таким образом, выступает как хранилище библиотеки онтологий. Грубер представлял, что интеллектуальные системы будут выступать как библиотеки онтологий и свободно обмениваться онтологиями между собой. При этом библиотеке онтологий уже не обязательно быть интеллектуальной системой, достаточно просто предоставлять сервис по передаче онтологий по требованию.

Составление описания декларативного знания обычно требует большой работы и определенных навыков. Для обозначения этой работы, а также ее результата, Грубер ввел в обиход специальный термин ***«концептуализация».*** Описание он называл «спецификацией». Таким образом, онтология по Груберу определяется как спецификация концептуализации [13].

В [13-16] выявлены основные причины возникновения потребности в создании онтологий.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий. К примеру, пусть, несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную

информацию для ответов на запросы пользователей или как входные данные для других приложений.

Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нужно создать большую онтологию, то можно интегрировать несколько существующих онтологий, описывающих части большой предметной области. Также можно повторно использовать основную онтологию, но расширить ее для описания интересующей предметной области.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний – это еще один вариант общего применения онтологий. Можно описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов.

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов

чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении.

Приведенные выше причины определяют основные цели создания онтологий, а области применения их чрезвычайно обширны.

Можно ввести в рассмотрение ***модель расширяемой онтологии***. Модель расширяемой онтологии является достаточно мощной для спецификации процессов формирования пространств знаний в Internet. Вместе с тем, и эта модель является неполной в силу своей пассивности даже там, где определены соответствующие процедурные интерпретации и введены специальные функции пополнения онтологии.

Введем в рассмотрение понятие онтологической системы. Под формальной моделью онтологической системы ∑o будем понимать триплет вида:

∑o=<ometa, {Od&t}, Ξinf>

Ometa – онтология верхнего уровня (метаонтология);

{Od&t} – множество предметных онтологий и задач предметной области;

Ξinf – модель машины вывода, ассоциированной с онтологической системой ∑o.

Использование системы онтологий и специальной машины вывода позволяет решать в такой модели различные задачи. Расширяя систему моделей {Od&t}, можно учитывать предпочтения пользователя, а изменяя модель машины вывода, вводить специализированные критерии релевантность получаемой в процессе поиска информации и формировать специальные репозитарии накопленных данных, а также пополнять при

необходимости используемые онтологии. В модели ∑о имеются три онтологические компоненты:

- метаонтология;
- предметная онтология;
- онтология задач.

Метаонтология оперирует общими концептами и отношениями, которые не зависят от конкретной предметной области. Концептами метауровня являются общие понятия. Тогда на уровне метаонтологии мы получаем интенсиональное описание свойств предметной онтологии и онтологии задач. Онтология метауровня является статической, что дает возможность обеспечить здесь эффективный вывод.

Предметная онтология содержит понятия, описывающие конкретную предметную область, отношения, семантически значимые для данной предметной области, и множество интерпретаций этих понятий и отношений (декларативных и процедурных). Понятия предметной области специфичны в каждой прикладной онтологии, но отношения – более универсальны. Поэтому в качестве базиса обычно выделяют такие отношения модели предметной онтологии, как part_of, kind_of, contained_in, member_of, see_also и некоторые другие.

Онтология задач в качестве понятий содержит типы решаемых задач, а отношения этой онтологии, как правило, специфицируют декомпозицию задач на подзадачи.

Машина вывода онтологической системы в общем случае может опираться на сетевое представление онтологий всех уровней.

При этом ее функционирование будет связано: с активацией понятий и/или отношений, фиксирующих решаемую задачу (описание исходной ситуации); определением целевого состояния(ситуации); выводом на сети,

заключающемся в том, что от узлов исходной ситуации распространяются волны активации, использующие свойства отношений, с ними связанных. Критерием остановки процесса является достижение целевой ситуации или превышение длительности исполнения.

7.2. Языки описания онтологий

Язык описания онтологий — формальный язык, используемый для кодирования онтологии. Существует несколько подобных языков. Вот список некоторых из них [13-16]:

- OWL — Web Ontology Language, стандарт W3C, язык для семантических утверждений, разработанный как расширение RDF и RDFS;
- KIF (англ.)русск. (англ. Knowledge Interchange Format — формат обмена знаниями) — основанный на S-выражениях синтаксис для логики;
- Common Logic (CL) (англ.) русск. — преемник KIF (стандартизован — ISO/IEC 24707:2007);
- CycL (англ.) русск. — онтологический язык, использующийся в проекте Cyc. Основан на исчислении предикатов с некоторыми расширениями более высокого порядка;
- DAML+OIL (англ.) русск. (FIPA).

Главный элемент языка RDF – это тройка, или триплет. Тройка представляет собой совокупность трех сущностей: субъект; объект; предикат.

Предикаты еще часто называют отношениями. Тройка имеет также представление в виде графа вида субъект–предикат–объект, где субъект и объект представлены как узлы, а предикат выступает в роли ребра, которое эти узлы соединяет.

Базовым элементом языка OWL является класс всех классов, определяемый как owl:Class. Любой OWL-класс должен быть задан как экземпляр класса owl:Class.

В языке OWL также присутствуют два предопределенных класса:

- Класс owl:Thing (сущность), обозначающий множество всех индивидов.
- Класс owl:Nothing (ничто), обозначающий пустое множество.

Каждый класс OWL является дочерним классом класса owl:Thing и родительским классом класса owl:Nothing. Наследование классов в языке OWL задается с помощью конструкции rdfs:subClassOf.

В OWL существует разделение свойств на два класса:

- Объектные свойства используются для связывания индивидов друг с другом. Объектные свойства – это экземпляры класса owl:ObjectProperty.
- Свойства типов данных связывают индивидов с так называемыми значениями типов данных (data values). Под значениями здесь подразумеваются RDF-литералы, или типы данных, определенные в XML Schema. Свойства типов данных – это экземпляры класса owl:DatatypeProperty.

Классы owl:ObjectProperty и owl:DatatypeProperty являются дочерними классами класса rdf:Property.

Язык OWL позволяет описывать различные характеристики классов и свойств, которые обычно задаются как разного рода ограничения на структуру связей между своими экземплярами. Эти ограничения выражаются в виде предопределенных соотношений, называемых в языке OWL аксиомами. В этом состоит основное отличие языка OWL от RDFS. Эти ограничения позволяют выражать в онтологии более тонкие вещи, чем с помощью RDFS.

Таких ограничений в языке OWL множество. Но все они подобраны таким образом, чтобы не снизить производительность алгоритма логического вывода по фактам, которые описаны в онтологии.

Для работы с языками онтологий существует несколько видов технологий: редакторы онтологий (для создания онтологий), СУБД онтологий (для хранения и обращения к онтологии) и хранилища онтологий (для работы с несколькими онтологиями).

Наиболее широкое распространение онтологии нашли во Всемирной паутине. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты, до категоризаций продаваемых товаров и их характеристик.

Для создания и редактирования онтологий используются специальные программы – редакторы онтологий. В данный момент в сети Интернет представлено множество онтологических редакторов, большинство из которых кроссплатформенны, то есть могут быть установлены на компьютер с любой операционной системой.

Самые известные из них – Ontolingua, Protégé, OntoEdit, OilEd, WebOnto, OntoSaurus, HOZO и др. Для работы этих онторедакторов необходимо установленное Java ПО, которое бесплатно доступно в Интернете. Функциональность онторедакторов можно определить по следующим параметрам:

- осуществление поиска по онтологии;
- редактирование (ввод, корректировка, удаление);
- логический контроль при вводе;
- тестирование функциональности;
- взаимодействие с другими онтологиями.

Был выбран редактор Protégé, разработанный в Стэнфордском университете (Stanford University) под руководством Марка Мьюсенадля построения онтологий и баз знаний. Редактор Protégé доступен для

бесплатного скачивания на официальном сайте. Этот редактор поддерживает язык OWL в последней редакции.

7.3. Основные правила разработки онтологий

В литературе предлагаются следующие основные правила разработки онтологий.

1. Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.

2. Разработка онтологии – это обязательно итеративный процесс. В процессе создания важно возвращаться к уже созданным классам и отношениям и уточнять, и добавлять информацию в случае необходимости. Иногда полезно вносить и кардинальные изменения для улучшения общей структуры онтологии.

3. Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают предметную область. Это поможет лучше понимать онтологию людям, не являющимся разработчиками, но заинтересованными в применении ее к своим приложениям.

То есть, знание того, для чего будет использована онтология и насколько детальной или общей она будет, повлияет на многие решения, касающиеся моделирования. Среди нескольких жизнеспособных альтернатив нужно определить, какая поможет лучше решить поставленную задачу и будет более наглядной, более расширяемой и более простой в обслуживании. Также нужно помнить, что онтология – это

модель реального мира и понятия в онтологии должны отражать эту реальность. После того, как будет определена начальная версия онтологии, можно оценить и отладить ее, используя ее в приложениях или в методах решения задач и/или обсудив ее с экспертами предметной области. В результате почти наверняка нужно будет пересмотреть начальную онтологию. Этот процесс итеративного проектирования, вероятно, будет продолжаться в течение всего жизненного цикла онтологии.

7.4. Обзор существующих онтологий

Онтология Pizza описывается в документации по созданию онтологий в Protégé и является распространенным примером для описания возможностей данной программы.

В ней описывается онтология пицц. На рис.7.1 представлена схема онтологии, созданная при помощи плагина программы Protégé OWLViz. Любая пицца (Pizza) состоит из основы (PizzaBase) и начинки (PizzaTopping). В качестве основы рассматриваются два вида: DeepPanBase и ThinAndCrispyBase. Разнообразных начинок гораздо больше, поэтому они объединены в подклассы. Например, подкласс CheeseTopping содержит все виды сырных начинок, а подкласс FishTopping – все виды рыбных начинок и т.д. Также есть подклассы MeatTopping (мясные начинки), FruitTopping (фруктовые начинки), HerbSpiceTopping (начинки из трав и специй), NutTopping (ореховые начинки), SauceTopping (соусы) и VegetableTopping (овощные начинки). Все эти подклассы не могут пересекаться, то есть ни один индивид не может входить в два или несколько этих подклассов одновременно, поэтому между ними существует отношение DisjointWith (не пересекается).

В классе Pizza существует подкласс NamedPizza, в котором подклассы – названия различных пицц. У каждой пиццы есть основа и есть начинки,

поэтому существует два основных отношения или свойства: hasBase (имеет основу) и hasTopping (имеет начинку). Два этих свойства находятся в отношении SubPropertyOf к свойству hasIngredient (иметь ингредиент). Ко всем этим свойствам определены противоположные свойства (то есть находящиеся в отношении InverseOf): isIngredientOf, isBaseOf и isToppingOf. У каждого отношения (свойства) определены Domains – множество классов, на котором определено это отношение и Ranges – множество классов, являющееся областью значений. Например, для отношения hasTopping в качестве Domains выступает класс Pizza, а в качестве Range – класс PizzaTopping.

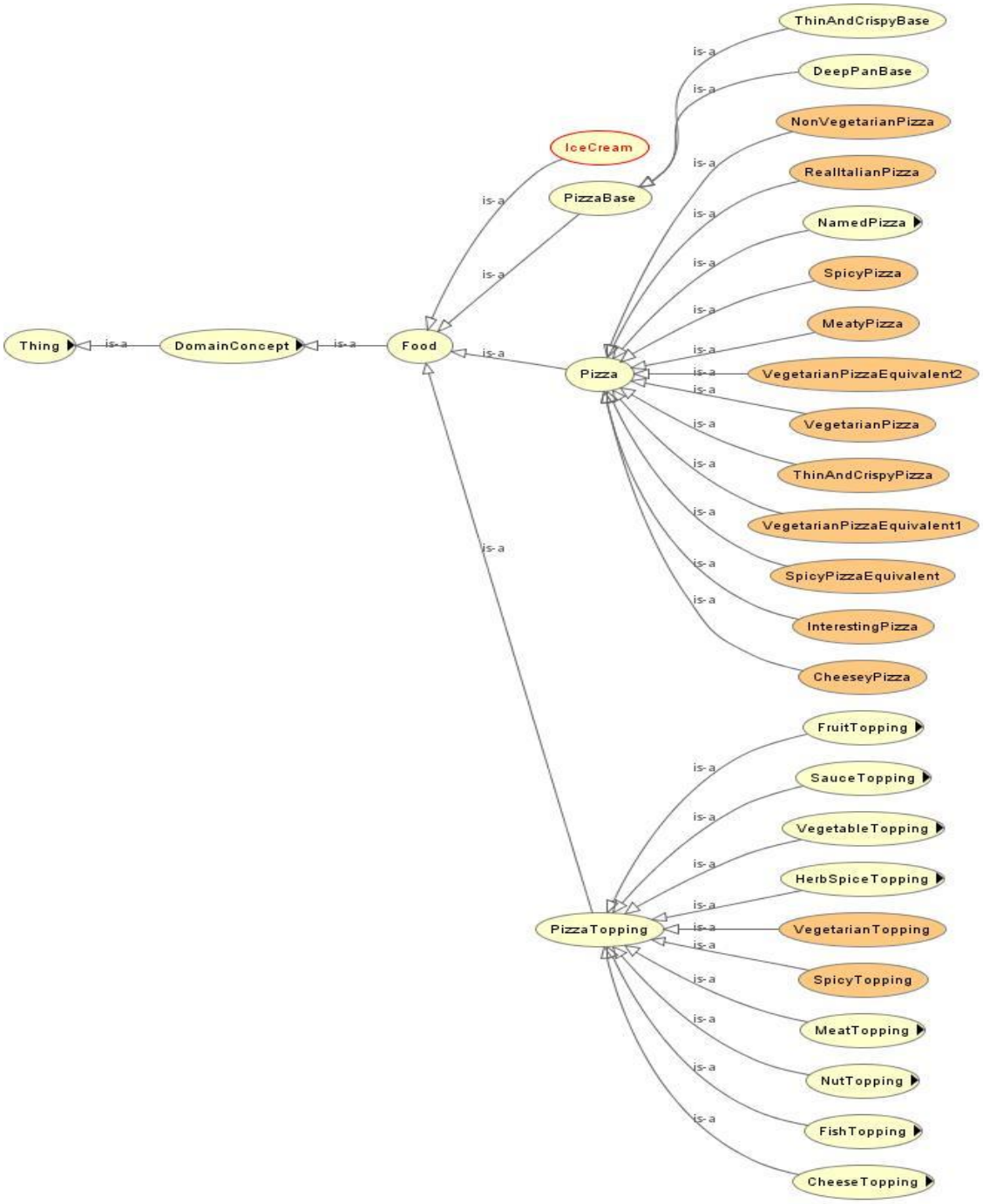


Рис.7.1. Схема онтологии Pizza

Также в классе Pizza есть подклассы, определенные с помощью отношения EquivalentTo (эквивалентности) некоторому логически заданному классу. Например, класс CheeseyPizza EquivalentTo Pizza and (hasTopping some CheeseTopping), это означает, что класс CheeseyPizza является пересечением двух классов: Pizza и (hasTopping some CheeseTopping). Второй класс представляет собой логический класс,

объединяющий всех индивидов, которые имеют хотя бы какую-то начинку из класса CheeseTopping. Аналогичным образом, например, определяется класс MeatyPizza EquivalentTo Pizza and (hasTopping some MeatTopping).

Для определения вегетарианской пиццы используется следующее выражение:VegetarianPizza EquivalentTo Pizza and (not (hasTopping some FishTopping)) and (not (hasTopping some MeatTopping)). То есть вегетарианская пицца – это пицца, которая не имеет рыбных и мясных начинок в своем составе. После этого легко определяется противоположный класс невегетарианской пиццы: NonVegetarianPizza EquivalentTo Pizza and (not (VegetarianPizza)).

Как уже говорилось, в классе Pizza есть подкласс NamedPizza с названиями наиболее распространенных пицц. Для каждой пиццы описывается ее состав. Например, пицца American hasTopping only (MozzarellaTopping or PeperoniSausageTopping or TomatoTopping), hasTopping some MozzarellaTopping, hasTopping some PeperoniSausageTopping, hasTopping some TomatoTopping. Это означает, что пицца American имеет в качестве начинки сыр Mozzarella, колбаса Paperoni, томаты и ничего другого. Исходя из этих сведений устройство логического вывода, в программе Protégé это OWL Reasoner, делает вывод, что пицца American является подклассом для следующих классов: CheeseyPizza, так как имеет в своем составе сыр, MeatyPizza, так как имеет в своем составе колбасу, InterestingPizza, так как имеет в своем составе не менее трех ингредиентов.

Эта онтология может быть использована в ресторанах, кафе и других заведениях, где готовят пиццу. Ее повсеместное использование принесет некоторую систематичность и будет гарантировать, что при заказе пиццы того или иного наименования, заказчик будет знать, что входит в ее состав, и будет уверен, что он получит то, что хочет, вне зависимости от заведения, города или даже страны.

На рис.7.2. представлена онтология нефтегазодобывающего предприятия и повышения экологической безопасности его работы, разрабатываемая для обеспечения оперативного мониторинга его технологической инфраструктуры.

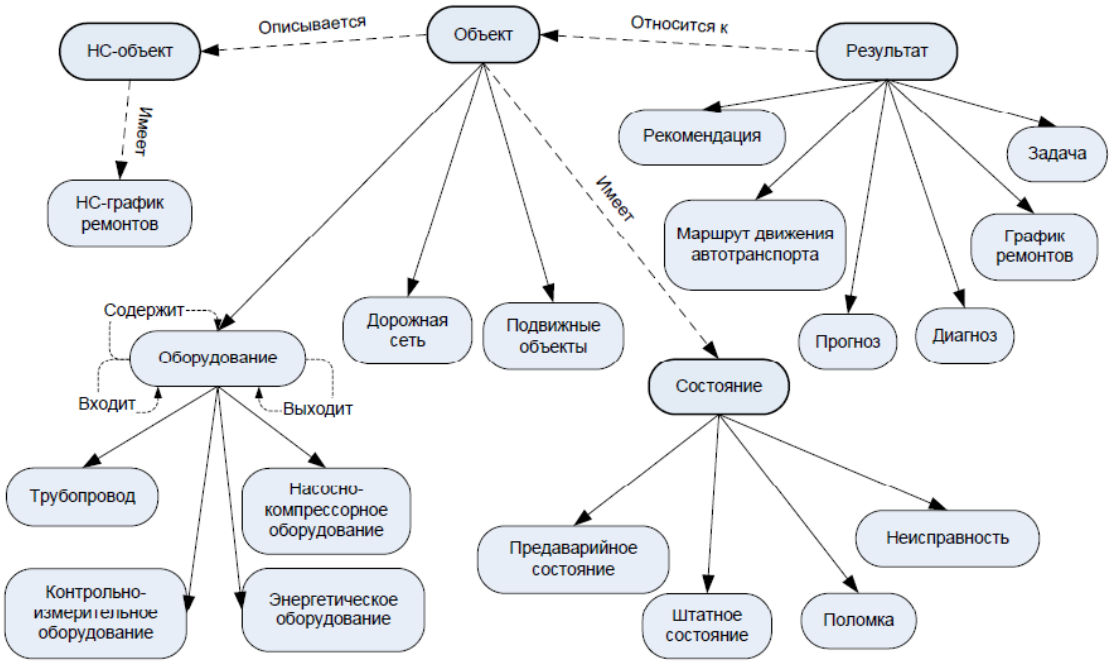


Рис. 7.2. Пример онтологии программного обеспечения нефтегазодобывающего предприятия

Существует множество разработанных онтологий в различных отраслях и сферах. Также существую онтологии, которые являются объединением ранее построенных онтологий [34-36].

Например, Biochemical ontologies (биохимические онтологии), содержащие более 30 онтологий, которые описывают атомы, молекулы, связи и процессы, связанные с ними.

Онтология *camera* описывает различные части фотокамеры и их взаимосвязи друг с другом.

Онтология Finance представляет собой онтологию по финансовым инструментам, а также процессам и процедурам, связанным с ценными бумагами.

Infrastructure Product Ontology – онтология, описывающая коммунальную инфраструктуру и охватывает пять секторов коммунальных услуг: водоснабжение, канализация, газ, электричество и связь.

Protein Ontology – онтология белков, описывающая все декларативные знания о них, в том числе их классификацию, что позволяет строить логические рассуждения.

Tourism – онтология для создания семантики веб-сайтов, связанных с туризмом.

Таких примеров большое количество, они доступны в интернете и могут применяться в различных информационных системах. Их тематика обширна и безгранична. Но остается еще много областей, в которых онтологии еще не получили широкого распространения или вовсе не применялись. Как уже ранее говорилось, онтологии могут существенно упростить работу и взаимодействие различных, распространенных в наше время информационных систем друг с другом. Поэтому проектирование онтологий на данный момент является достаточно актуальным направлением и приобретает все большую популярность.

8. Экспертные системы на базе прецедентов

8.1. Рассуждения по прецедентам

Создание экспертных систем на базе прецедентов тесно связана с актуальной проблемой в области искусственного интеллекта и конструирования перспективных интеллектуальных (экспертных) систем – проблемой моделирования человеческих рассуждений (рассуждений «здравого смысла»).

В современных и дорогостоящих зарубежных средствах конструирования интеллектуальных (экспертных) систем (G2, RTworks и др.) практически отсутствуют развитые средства, реализующие механизмы правдоподобных рассуждений, которые способны обеспечить работу системы в условиях неопределенности как в исходной информации, получаемой от объекта управления и среды, так и в экспертных знаниях. Наличие подобных механизмов рассуждений (индуктивных, абдуктивных, нечетких, аргументации, на основе аналогий и прецедентов в системах экспертной диагностики и поддержки принятия решений позволяет своевременно осуществлять диагностирование проблемной ситуации на объекте и дает возможность лицам, принимающим решения (ЛПР), принимать более адекватные и экономически выгодные управляющие воздействия на объект управления с целью нормализации проблемной ситуации.

В настоящее время основное внимание уделяется правдоподобным рассуждениям на основе прецедентов (накопленного опыта), активно применяемым в диагностических системах (медицинской диагностике, диагностике спутникового оборудования и т.д.), в юриспруденции, экспертных системах и системах машинного обучения [20-21]. Рассматривается возможность использования различных метрических

алгоритмов для извлечения прецедентов из базы прецедентов системы и учета коэффициентов важности параметров объекта, задаваемых экспертом.

Рассуждения по прецедентам — это метод формирования умозаключений, опирающийся не на логический вывод от исходных посылок, а на поиск и анализ случаев формирования подобных умозаключений в прошлом.

С точки зрения решения задач, рассуждения по прецедентам — это метод получения решения путем поиска подобных проблемных ситуаций в памяти, хранящей прошлый опыт решения задач, и адаптации найденных решений к новым условиям. Применение данного метода для решения задач оправдано в случае выполнения следующих условий, касающихся природы прикладной области.

Во-первых, подобные задачи должны иметь подобные решения (принцип регулярности). В этом случае накопленный опыт решения задач может служить отправной точкой процесса поиска решения для новых подобных задач.

Во-вторых, виды задач, с которыми сталкивается решатель, должны иметь тенденцию к повторению. Это условие гарантирует, что для многих проблем в будущем будет существовать аналог в прошлом опыте.

Рассмотрим в общих чертах, что собой представляет процесс рассуждений по прецедентам.

Прецедент можно определить, как единичную запись предыдущего опыта. Какую именно информацию содержит такая запись, зависит от предметной области и целей использования прецедента.

В большинстве энциклопедических источников прецедент определяется как случай, имевший место ранее и служащий примером или оправданием для последующих случаев подобного рода. Вывод на основе прецедентов (CBR – Case-Based Reasoning) является подходом, позволяющим решить новую задачу, используя или адаптируя решение уже известной задачи. Как правило, такие методы рассуждений включают

в себя четыре основных этапа, образующие так называемый цикл рассуждения на основе прецедентов или CBR-цикл.

В случае применения CBR-метода для решения задач прецедент содержит, по меньшей мере, постановку задачи и способ ее решения. Множество всех прецедентов, накопленных в процессе работы CBR-метода, формируют информационное хранилище, называемое *библиотекой прецедентов*.

CBR в общем случае представляет собой циклический процесс: решение проблемы, запоминание этого решения в качестве прецедента, решение новой проблемы и так далее. CBR-цикл может быть описан следующими тремя процессами:

1. Поиск похожего прецедента(ов) — поиск прецедента(ов), у которых постановка задачи наиболее похожа на описание новой задачи.
2. Адаптация — получение на базе найденного прецедента решения для новой задачи. Этот этап может также включать проверку полученного нового решения на корректность и толерантность к ошибкам, и, возможно, дополнительную коррекцию решения.
3. Сохранение прецедента — сохранение той части полученного опыта, которая может оказаться полезной для решения новых задач (пополнение или корректировка библиотеки прецедентов).

Процесс адаптации разделяют на повторное использование и проверку. Так как эти процессы часто оказываются сильно взаимосвязанными, разделение — это весьма условно.

Таким образом, решение каждой задачи в CBR сводится к последовательному решению подзадач поиска схожих прецедентов, получения из них (путем адаптации) решения для новой задачи и сохранения нового случая решения задачи в библиотеке прецедентов. Графически, CBR-цикл можно представить в виде следующей диаграммы (рис.8.1).

Для процессов, входящих в CBR-цикл, а также для самого прецедента (как хранилища информации) существует множество вариантов реализации. На уровне процессов уже нет каких-то универсальных решений; реализации процессов опираются на знания о предметной и проблемной области, то есть они специфичны в каждой конкретной прикладной системе.

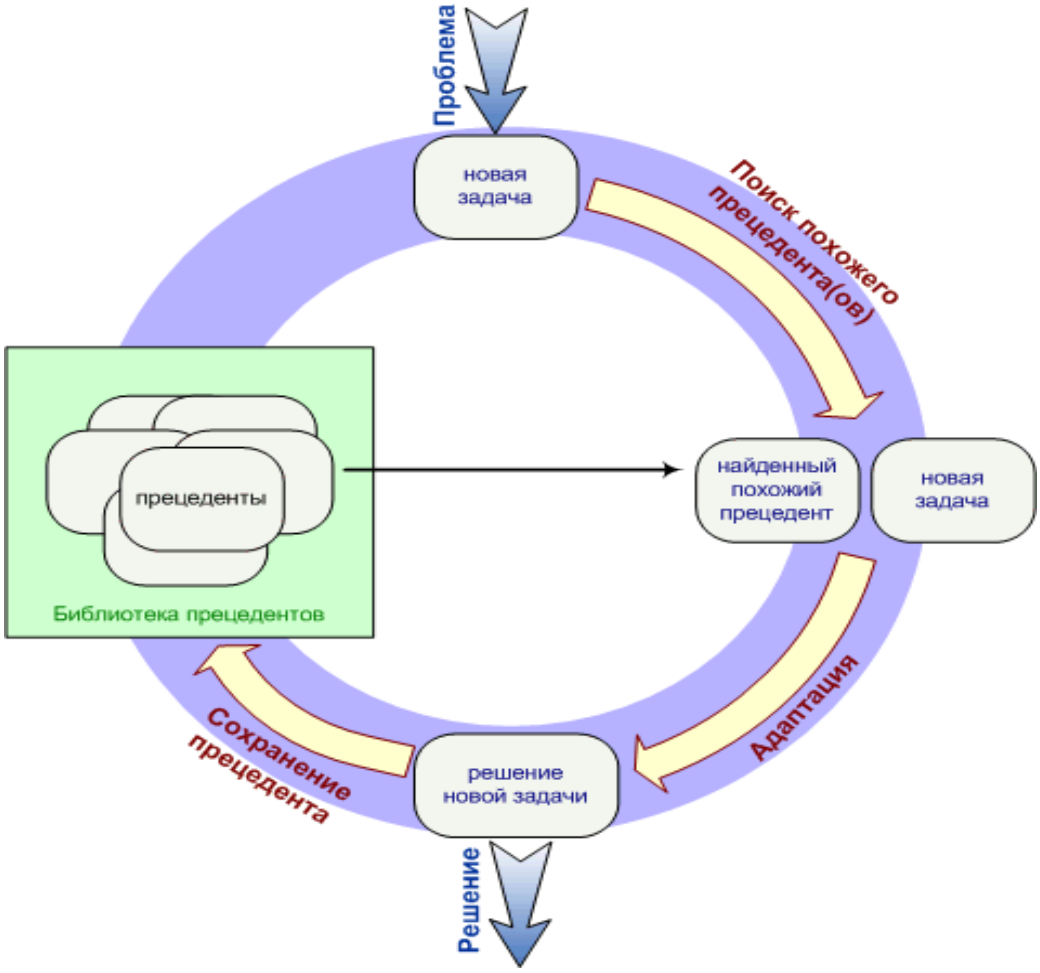


Рис. 8.1. CBR-цикл

8.2 Достоинства и недостатки использования прецедентов

Метод рассуждений по прецедентам имеет свои преимущества и недостатки по сравнению с другими методами получения решений. Среди *преимуществ* можно выделить следующее.

1. Легкость приобретения знаний (в противоположность системам, основанным на правилах). Создание системы, основанной на правилах, требует таких трудоемких этапов как получение, формализация и обобщение экспертных знаний, верификация системы на корректность и полноту. В системах, основанных на прецедентах, приобретение знаний происходит путем формального описания случаев из практики (нет необходимости обобщения, и вытекающей отсюда угрозы пере обобщения).

2. Возможность объяснения полученного решения (в противоположность системам, основанным на нейронных сетях). CBR-система может объяснить полученное решение путем демонстрации успешного прецедента с отражением показателей сходства и рассуждений, использовавшихся при адаптации прецедента к новой ситуации. Такое объяснение может быть даже лучше, чем объяснения, выдаваемые системами, основанными на правилах. Последние иногда выдают очень длинные последовательности рассуждений, а сами правила конечному пользователю (в отличие от эксперта) могут казаться неочевидными или слишком сложными.

3. Возможность работы в предметных областях, которые невозможно полностью понять, объяснить или смоделировать.

4. Возможность обучения в процессе работы. Причем обучение будет происходить только в определенных направлениях, которые реально встречаются на практике и востребованы (нет избыточности).

5. Возможность избежать повторения ошибки (обучение сбоям и их причинам для избегания их появления в дальнейшем).

6. Возможность получения решений путем модификации прецедентов позволяет уменьшить объем вычислений в предметных областях, где генерация решения «с нуля» требует больших усилий.

Основными *недостатками* использования прецедентов являются:

1. Метод применим только в областях, где выполняется принцип регулярности и имеет место повторяемость видов задач. Если все время

решаются принципиально новые задачи или если решения сходных задач различны, то метод на базе прецедентов неприемлем.

2. Некомпактное (без обобщения) хранение знаний (опыта).

3. Сложность и специфичность процессов поиска подобных случаев и адаптации решения.

8.3. Примеры реализации

В работе [20,21] рассмотрены методы правдоподобных рассуждений на основе аналогий и прецедентов для систем искусственного интеллекта. Исследованы различные способы представления и извлечения прецедентов из БЗ системы. Описаны возможности разработанных программных средств, базирующихся на CBR-технологии и методах структурной аналогии.

С помощью нейросетевой ЭС формируется, предварительное заключение, предъявляемое врачу-специалисту. Но во всяких правилах возможно исключения. Для этого использовалась методика обработки прецедентов для критического анализа результатов применения правил. Это выполняется путем поиска прецедентов, аналогичных рассматриваемому случаю, если этот случай можно считать исключением из правил.

В автоматизированных системах обучения возникает проблема получения строгих функциональных зависимостей между входными и выходными параметрами, связанная со слабой формализуемостью объекта управления. В работе предполагается использовать «обходной» вариант, основанный на методе вывода по прецедентам.

Рассматриваются вопросы, связанные с реализацией механизмов правдоподобных рассуждений на основе прецедентов для систем экспертной диагностики. Основное внимание уделено проблеме извлечения прецедентов из БП системы. Предложен модифицированный метод

ближайшего соседа для извлечения прецедентов. Разработаны соответствующие метрические алгоритмы извлечения прецедентов, обеспечивающие учет коэффициентов важности параметров объекта и работу с неполной информацией.

Применению прецедентов посвящена исследованию средств программирования для решения задач планирования целенаправленной деятельности. Описана реализация ассоциативного планирования в решателе геометрических задач. Предложен алгоритм поиска близких ситуаций, основанный на компиляции описаний стереотипных ситуаций в сеть специального вида. Рассмотрены критерии отбора стереотипов для получения эффективного плана. Приведены решения интеллектуальных многошаговых задач, демонстрирующие метод ассоциативного планирования на основе стереотипов и близких ситуаций.

Рассматриваются методы поиска решения на основе структурной аналогии и прецедентов в плане применения их в современных информационных систем поддержки принятия решений. Описаны базовые алгоритмы поиска решения на основе аналогии свойств и аналогии отношений. Предложены более эффективные в плане качества получаемого решения алгоритмы. Алгоритмы используют модифицированную структурную аналогию, что позволяет учитывать множество свойств, определяющих первоначальный контекст аналогии, и обеспечивает поиск решения в случае, когда такое множество свойств является пустым, а также переносить от источника аналогии на приемник только те факты, которые уместны в контексте построенной аналогии. Предложены методы определения сравнительных оценок полученных аналогий с учетом контекста и важности параметров объекта.

Метод принятия решений, основанный на совместном применении ранее не комбинировавшихся методов извлечения знаний и вывода по

прецедентам, где методы добычи данных используются для автоматического отбора из большой базы прецедентов [20]. В работе предложена локальная контекстно-зависимая метрика, имеющая интерпретацию расстояния и позволяющая ранжировать объекты, по отношению к исследуемому, целыми числами. При построении метрики используется предложенный авторами модифицированный метод кластерного анализа, ориентированный на распознавание объектов в ситуациях, когда объекты и кластеры имеют не полностью совпадающие наборы признаков. Эта метрика применима к широкому кругу приложений и не накладывает ограничений на типы используемых атрибутов. Что касается адаптации решения – предлагаемый метод позволяет сделать эту проблему более формализуемой. Хотя в общем случае проблема адаптации остается зависимой от предметной области, предложенный подход значительно упрощает эту задачу, так как учитывает фоновое знание.

Рассматривается модуль правдоподобного вывода по прецедентам как один из компонентов системы автоматизации исследований. Компонент представляет собой динамическую библиотеку, реализующую функции: создания модели прецедента, формирования базы прецедентов и рассуждений по прецедентам. Приведено описание архитектуры и функциональных блоков модуля.

В статье [8] рассматривается реализация баз знаний прецедентов активных экспертных систем на основе ансамблевых моделей нейросетей. Решается задача распознавания прецедентов ансамблевой моделью нейросетей в процессе комплексных отказов, с учетом частичной или полной неопределенности параметров системы “временной автомат приемистости и авиационный газотурбинный двигатель”.

В статье [8] рассматривается технология сборки систем из компонентов. В данной работе под компонентом системы автоматизации

исследований понимается компонент, который, кроме основной функциональности, обеспечивает реализацию механизма внутренней памяти и предложенного авторами унифицированного интерфейса компонента. Среди множества методов, которые могут быть алгоритмизированы и реализованы в виде программного кода компонентов для системы автоматизации исследований, выделен метод, основанный на изучении существующего в определенной предметной области опыта – это правдоподобный вывод по прецедентам. Выбор данного метода обусловлен тем, что зачастую в областях, где требуется поддержка исследователя (например, обеспечение надежности и безопасности сложных технических систем в нефтехимии), к моменту возникновения новой проблемы уже накоплен значительный опыт решения похожих проблем, возникавших ранее на подобном оборудовании. Однако невозможность или сложность аналитической обработки этого опыта по ряду причин (например, отсутствие специалистов необходимой квалификации, экспертов и др.) приводит к невозможности эффективно использовать эти знания и реализовать их потенциал. Представление же этого опыта в виде прецедентов, и его автоматизированная обработка при помощи специализированных программных систем позволяют значительно повысить эффективность его повторного использования. В работе указано подробное решение проблемы при помощи прецедентов.

Нейросетевая экспертная система на основе прецедентов приведена на рис.8.2.

В общей структуре НЭСП главенствует основной модуль-интегратор, который управляет взаимодействием интеллектуальных и программных модулей. Интеллектуальная подсистема[2] включает следующие компоненты:

–модуль приобретения знаний – включает анализ и извлечение;

[2] http://ej.kubagro.ru/2013/02/pdf/24.pdf

–входной информации из базы данных абонентов (БДА); входные данные преобразуются в форму прецедента или в форму продукционного нечеткого правила;

– база знаний прецедентов (БЗП);

– продукционная нечеткая база знаний (ПНБЗ) содержит правила в форме нечетких продукций;

– механизм поиска по прецедентам (МПП);

– блок обучения нейронной сети – преобразует правила из ПНБЗ в обучающие выборки для нейронной сети;

– нейро-нечеткий механизм (ННМ) – программный блок, реализующий структуру нечеткого контроллера на основе нейронной сети;

– блок объяснений решения;

– блок адаптации данных (АД) – преобразует результат нейросетевого поиска решения в форму нового прецедента/

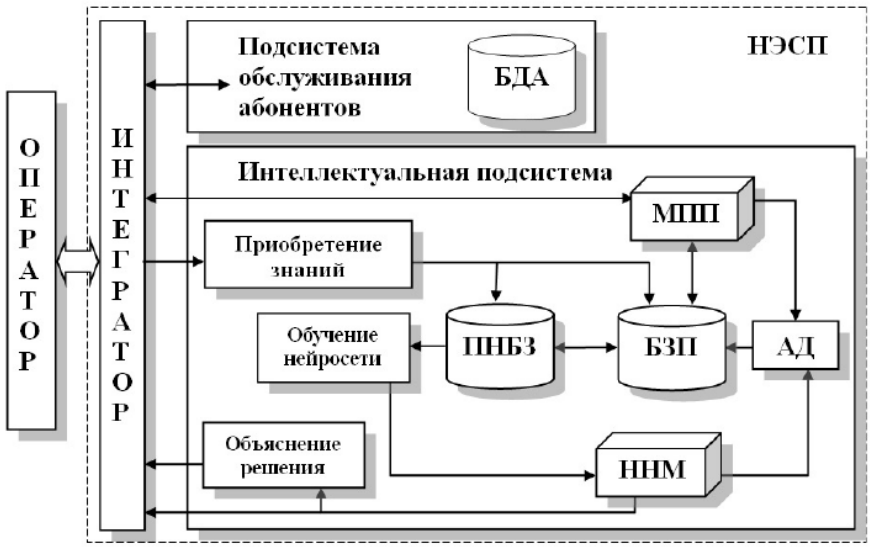


Рис.8.2. Структура нейросетевой экспертной системы на основе прецедентов

Входные переменные представляют собой характеристики проблемы, которая возникла у абонента. В качестве выходных переменных выступают причины, повлекшие возникновение проблемы.

8.4. Принципы создания экспертной системы на базе прецедентов

Экспертная система на базе прецедентов должна удовлетворять следующим требованиям:

– наличие базы знаний, хранящей прецеденты;

– реализация модуля вывода на основе прецедентов;

– наличие модуля, формирующего протокол решения;

– разработка не менее 16 прецедентов для демонстрации работы программы.

Для демонстрации возможностей экспертной системы, базирующейся на прецедентах, была выбрана предметная область – кулинария, а конкретно, приготовление мясных блюд. Разрабатываемая ЭС предназначается для формирования кулинарных рецептов из мяса. Эта программа принимает информацию о целевых характеристиках блюда (тип, вкусовые качества, основные ингредиенты) и формирует подходящий рецепт. Результатом работы программы должен быть рецепт последовательности операций, позволяющий приготовить такое блюдо.

Получив заказ, программа просматривает свою базу прецедентов, отыскивает в ней рецепт приготовления аналогичного блюда и адаптирует его в соответствии с особенностями текущего запроса.

Исходя из поставленных задач, программа должна содержать в себе модули извлечения прецедентов, модификации и сохранения нового прецедента. Прецеденты должны храниться в специальной базе, которая позволит описать все взаимосвязи. Структурная схема программы, имеющая отношение к манипуляции с базой прецедентов представлена на рис. 8.3.

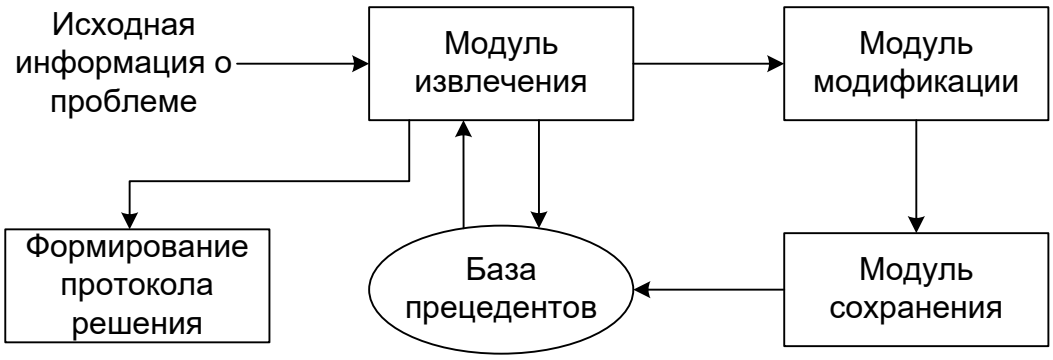


Рис. 8.3. Структурная схема программы

Для успешной реализации рассуждений на основе прецедентов необходимо обеспечить корректное извлечение прецедентов из БП системы экспертной диагностики. Выбор метода извлечения прецедентов напрямую связан со способом представления прецедентов и соответственно со способом организации БП.

Основные способы представления прецедентов можно разделить на следующие группы:

– параметрические;
– объектно-ориентированные;
– специальные (в виде деревьев, графов, логических формул и т.д.).

В большинстве случаев для представления прецедентов достаточно простого параметрического представления, т.е. представления прецедента в виде набора параметров с конкретными значениями и решения (диагноз по проблемной ситуации и рекомендации ЛПР): CASE(x_1, x_2, … , x_n, R), где $x_1 \ldots x_n$ – параметры ситуации, описывающей данный прецедент ($x_1 \in X_1$, $x_2 \in X_2, \ldots, x_n \in X_n$), R – диагноз и рекомендации ЛПР, n – количество параметров прецедента, а X_1, … , X_n – области допустимых значений соответствующих параметров прецедента.

Существует целый ряд методов извлечения прецедентов и их модификаций:

1. *Метод ближайшего соседа (NN – Nearest Neighbor).* Это самый распространенный метод сравнения и извлечения прецедентов. Он позволяет довольно легко вычислить степень сходства текущей проблемной ситуации и прецедентов из БП системы. С целью определения степени сходства на множестве параметров, используемых для описания прецедентов и текущей ситуации, вводится определенная метрика. Далее в соответствии с выбранной метрикой определяется расстояние от целевой точки, соответствующей текущей проблемной ситуации, до точек, представляющих прецеденты из БП, и выбирается ближайшая к целевой точка.

Преимуществами данного метода являются простота реализации и универсальность в смысле независимости от специфики конкретной проблемной области. К существенным недостаткам метода можно отнести сложность выбора метрики для определения степени сходства и прямую зависимость требуемых вычислительных ресурсов от размера БП, а также неэффективность при работе с неполными и зашумленными исходными данными.

На практике применяются различные модификации указанного метода. Обычно решение выбирается на основе нескольких ближайших точек (соседей), а не одной (метод k ближайших соседей). Возможно использование метода ближайшего соседа, основанного на знаниях о предметной области (определенных зависимостях между параметрами объекта).

2. Метод извлечения прецедентов на основе деревьев решений. Этот метод предполагает нахождение требуемых прецедентов путем разрешения вершин дерева решений. Каждая вершина дерева указывает, по какой ее ветви следует осуществлять дальнейший поиск решения. Выбор ветви осуществляется на основе информации о текущей проблемной ситуации.

Таким образом, необходимо добраться до концевой вершины, которая соответствует одному или нескольким прецедентам. Если концевая вершина связана с некоторым подмножеством прецедентов, то тогда для выбора, наиболее подходящего из них может использоваться метод ближайшего соседа. Такой подход рекомендуется применять для больших БП, т.к. основная часть работы по извлечению прецедентов выполняется заранее на этапе построения дерева решений, что значительно сокращает время поиска решения.

3. Метод извлечения прецедентов на основе знаний. В отличие от методов, описанных выше, данный метод позволяет учесть знания экспертов (ЛПР) по конкретной предметной области (коэффициенты важности параметров, выявленные зависимости и т.д.) при извлечении. Метод может успешно применяться совместно с другими методами извлечения прецедентов, особенно когда база прецедентов имеет большие размеры, и предметная область является открытой и динамической.

4. Метод извлечения с учетом применимости прецедентов. В большинстве систем, использующих механизмы рассуждений на основе прецедентов, предполагается, что наиболее схожие с текущей проблемной ситуацией прецеденты являются наиболее применимыми в этой ситуации. Однако это не всегда так. В основе понятия извлечения на основе применимости (адаптируемости) лежит то, что извлечение прецедентов базируется не только на их сходстве с текущей проблемной ситуацией, но и на том, насколько хорошую для желаемого результата модель они собой представляют. На выбор извлекаемых прецедентов влияет возможность их применения в конкретной ситуации. В некоторых системах эта проблема решается путем сохранения прецедентов вместе с комментариями по их применению. Использование указанного подхода позволяет сделать поиск решения более эффективным, заранее отбрасывая часть заведомо неперспективных прецедентов.

Помимо рассмотренных методов для извлечения прецедентов могут

успешно применяться и другие методы (например, аппарат искусственных нейронных сетей).

Сравнение систем, основанных на правилах и прецедентах представлено в [22]. Существует достаточно сильная аналогия между системами, основанными на правилах и прецедентах. И те, и другие необходимо каким-то образом индексировать, чтобы обеспечить эффективное извлечение. И те, и другие выбираются в результате сопоставления, причем выбор и ранжирование производятся на основании фоновых знаний, хранящихся в каких-либо дополнительных структурах, например, в виде фреймов (в MYCIN аналогичную роль выполняют таблицы знаний).

Различия между этими двумя классами систем. Они суммированы ниже.

Правила являются образцами — содержат переменные и не описывают непосредственно решение, а прецеденты являются константами.

Правило выбирается на основе точного сопоставления антецедента и данных в рабочей памяти. Прецедент выбирается на основе частичного сопоставления, причем учитываются еще и знания о сущности характеристик, по которым выполняется сопоставление.

Применение правил организовано в виде итерационного цикла — последовательности шагов, приводящих к решению. Прецедент можно рассматривать как приближенный вариант полного решения. Иногда, однако, появляется возможность итеративно проводить аналогию с разными прецедентами, которые "подходят" для различных частей проблемы.

Построение суждений на основе прецедентов поддерживает и другую стратегию решения проблем, которую называют "извлечение и адаптация". Эта стратегия существенно отличается и от эвристической классификации, и от стратегии "предложение и проверка".

В новом подходе есть нечто очень близкое нам интуитивно, поскольку весьма напоминает наш повседневный опыт. Даже на первый взгляд ясно, как привлекательно вспомнить аналогичный случай, принесший успех в прошлом, и поступить так же. Редко кто из нас затрудняет себя "нудными рассуждениями", когда можно быстро извлечь готовое решение.

Человеческая память подвержена сильному эмоциональному влиянию — нам свойственно помнить успехи и забывать о неудачах. Прошлые успехи всегда предстают в розовом свете, а потому во многих случаях нельзя рассматривать прецеденты как достаточно надежную основу для правильных выводов. Есть и еще одно существенное соображение, которое не позволяет нам безоговорочно довериться прецедентам, — масштабность. Можно говорить об анализе десятков прецедентов, но когда масштаб решаемой проблемы потребует сопоставления сотен и тысяч прецедентов, существующему механизму анализа задача окажется не по плечу.

Контрольные вопросы

1. Достоинства и недостатки создания ЭС на базе прецедентов.
2. Каким образом производится поиск наиболее близкого прецедента?
3. Возможна ли конфликтная ситуация при поиске прецедента?
4. Характеристика и состав базы прецедентов.
5. Понятие адаптации решения.
6. Методы извлечения прецедентов из базы прецедентов.
7. Метод ближайшего соседа для извлечения прецедентов из базы прецедентов.
8. Метод извлечения прецедентов на основе деревьев решений.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АД - блок адаптации данных

БДА - базы данных абонентов

БЗ - база знаний

ГЭС - гибридная экспертная система.

МЭС - мягкая экспертная система

НС - нейронная сеть

НЭС - нейросетевая экспертная система

ОВ - обучающая выборка

ППФ - правильно построенная функция

ЭС - экспертная система

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ростовцев, В.С. Принципы построения экспертных систем [Электронный ресурс]: учеб. пособие/ В.С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров: 2007. –155с.
2. Ростовцев, В.С. Искусственные нейронные сети [Электронный ресурс]: учеб. Для студентов направления 230101.68.05/ В.С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров: 2014. –208 с.
3. Ростовцев, В.С. Теория и применение нечеткой логики [Электронный ресурс]: учебное пособие: для студентов направления 09.04.01 «Информатика и вычислительная техника» всех профилей подготовки, всех форм обучения / В.С. Ростовцев; ВятГУ, ФАВТ, каф. ЭВМ. – Киров: 2016. –111 с.
4. Башлыков, А. А. Экспертная диагностическая система как компонент интеллектуальной системы поддержки принятия решений реального времени [Текст] / А.А. Башлыков, А.П. Еремеев // Новости искусственного интеллекта. - 2002. - №3. - С. 35-40.
5. Джексон, П. Введение в экспертные системы [Текст] / П. Джексон. - М.; СПб.; Киев: Вильямс, 2001. – 624 с.: ил.
6. Люгер, Д., Ф. Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] - 4-е изд. - М.: Вильямс, 2003. – 864 с.
7. Савушкин С.А. Нейросетевые экспертные системы [Текст]// Нейрокомпьютер. 1992. №2. С. 29–36.
8. Жернаков С.В. Нейросетевая база знаний прецедентов активной экспертной системы для комплексного контроля и диагностики параметров авиационного двигателя[Текст]// Информационные технологии. 2002. №5. С. 45–53.
9. Ярушкина, Н. Г. Основы теории нечетких и гибридных систем [Текст]: учеб. пособие /Н. Г. Ярушкина. - М.: Финансы и статистика, 2004. – 320 с.: ил.

10. Яхъяева, Г. Э. Нечеткие множества и нейронные сети [Текст] / Г.Э. Яхъяева; Лаб. знаний, Интернет-ун-т информ. технологий - ИНТУИТ.ру. – М.: БИНОМ, 2006.

Интернет-источники

11. Д.П. Ветров, Д.А. Кропотов Программный комплекс для проектирования экспертных систем «ExSys» [Электронный ресурс] /Д.П.Ветров. – Режим доступа:http://www.ccas.ru/mmro/received.html.

12. Иллюстрированный самоучитель по экспертным системам. [Электронный ресурс] Режим доступа: http://claw.ru/enciklopedija-programmirovanija/ekspertnie-sistemi-samouchitel.html

13. Knowledge Systems Laboratory Stanford University. Ontologies Come of Age. 2000. [Электронный ресурс]. – Режим доступа: http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-abstract.html.

14. Лапшин В. А. Онтологии в компьютерных системах. [Текст] — М.: Научный мир, 2010.

15. Добров Б. В., Иванов В.В., Лукашевич Н.В., Соловьев В.Д. Онтологии и тезаурусы: модели, инструменты, приложения. [Текст] — М.: Бином. Лаборатория знаний, 2009. — 173 с. — ISBN 978-5-9963-0007-5.

16. Соловьев В.Д. Онтологии и тезаурусы [Текст]/ В.Д. Соловьев, В.В. Иванов, Б.В. Добров.– Казань, Москва. 2006. – 157 с.

17. Аксенов К.А. Построение оболочки экспертных систем для предметной области процессов преобразования ресурсов [Текст] // Современные проблемы науки и образования. – 2012. – № 6; URL: www.science-education.ru/106-7849

18. Варшавский П.Р. Реализация метода правдоподобных рассуждений на основе прецедентов для интеллектуальных систем поддержки принятия решений // Труды 10 национальной конференции по ИИ с международным участием КИИ-2006. В 3-х т. М.: Физматлит, 2006. Т.1. С. 303 – 311.

19.Варшавский П.Р., Еремеев А.П. Методы правдоподобных рассуждений на основе аналогий и прецедентов для интеллектуальных систем поддержки принятия решений // Новости Искусственного Интеллекта, № 3, 2006, с. 39-62.

20.Рассуждения, основанные н прецедентах [Электронный ресурс]. Режим доступа: http://masters.donntu.org/2011/fknt/pominchuk/library/article5.htm.

21.Малыхина М.П., Бегман Ю.В. Оценка эффективности гибридизации интеллектуальных методов на примере нейросетевой экспертной системы на основе прецедентов // Научный журнал КубГАУ, №86(02), 2013 года. http://ej.kubagro.ru/2013/02/pdf/24.pdf

22. Сравнение систем, основанных на правилах и прецедентах [Электронный ресурс]. Режим доступа: http://www.hardline.ru/selfteachers/Info/Programming/Introduction%20to%20expert%20systems/Glava%2022/Index10.htm.