Министерство образования и науки
Российской Федерации
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ
(ТУСУР)

Н. В. ЗАМЯТИН

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

Томск -2018

Рецензенты:
Доктор технических наук, профессор
Смирнов Г.В. (ТУСУР)
Доктор технических наук, профессор
Тарасенко Ф. П. (ТГУ)

Замятин Н. В.

В пособие включены необходимые материалы для самостоятельного изучения дисциплины «Системы искусственного интеллекта».

Рассматривается понятие искусственного интеллекта и системы искусственного интеллекта. Приводятся основные определения и понятия инженерии знаний, описываются модели и операции над знаниями. Классификация систем искусственного интеллекта. Описаны языки, инструментальные средства, методики разработки систем искусственного интеллекта, в том числе экспертных систем и других прикладных систем искусственного интеллекта.

Для студентов факультета дистанционного образования и факультета систем управления

ОГЛАВЛЕНИЕ

Введение

Уровень современных средств вычислительной техники способствует развитию интеллектуальной обработки информации и проектированию систем, реализующих технологии искусственного интеллекта, которые внедряются практически во все области человеческого общества. Поэтому будущим специалистам в области автоматизации обработки информации необходимо знать методы и средства извлечения и представления знаний, и на этой основе уметь проектировать системы искусственного интеллекта.

В настоящем пособии внимание сконцентрировано на концепциях искусственного интеллекта, положенных в основу систем искусственного интеллекта.

Недостаточное количество методической и специальной литературы по разработке различных классов прикладных систем искусственного интеллекта, недостаток квалифицированных специалистов в области инженерии знаний, незаполненный отечественный рынок программных продуктов для разработчиков экспертных систем затрудняют обучение и вхождение в сферу профессиональной деятельности будущих специалистов по автоматизации обработки информации.

Процесс разработки систем искусственного интеллекта имеет существенные отличия от разработки традиционных программных продуктов и требует участия специалистов (когнитологов) в инженерии знаний.

Поэтому основной целью издания настоящего учебного пособия является некоторое восполнение недостатка литературы и учебно-методических разработок по вопросам создания систем искусственного интеллекта, называемых также системами, основанными на знаниях.

Данное учебное пособие, подготовлено в соответствии с требованиями государственного образовательного стандарта по направлению 230100.62 «Информатика и вычислительная техника», ориентировано на подготовку специалистов

по программированию и информационным технологиям, в рамках которых читается дисциплина «Системы искусственного интеллекта»

Задачей дисциплины является разработка студентами под руководством преподавателя демонстрационного прототипа системы, содержащей базы знаний по соответствующей проблемной области. Реализация системы осуществляется на основе изученных методов, технологий искусственного интеллекта и инструментальных средств за счет времени, отведенного на лабораторные занятия (частично) и самостоятельную работу.

В пособии рассматриваются исторические этапы, методологические основы теории интеллектуальных информационных систем, их классификация, основные свойства и области применения. Основное внимание уделяется общим вопросам построения экспертных систем, их архитектуре, режимам работы, этапам разработки. Обсуждаются стратегии извлечения знаний, модели выявления знаний от экспертов и вопросы обработки экспертных оценок. Излагаются основы теории байесовских сетей вывода. Теоретические положения иллюстрируются примерами.

Пособие состоит из 12 глав. В первой главе дается общие понятия об искусственном интеллекте, классификации интеллектуальных систем. Во второй главе рассматриваются вопросы инженерии знаний. В третьей главе описаны модели представления знаний. В четвертой и пятой главах рассмотрены нейронные сети и нечеткая логика. В шестой главе даны методы вывода знаний и поиска решений. В седьмой и восьмой главах описаны языки и инструментальные методы систем искусственного интеллекта. В девятой и десятой главах представлены материалы по архитектуре и разработке систем искусственного интеллекта. В одиннадцатой главе рассмотрены общие принципы построения экспертных систем. Двенадцатая глава посвящена вопросам прикладных систем искусственного интеллекта.

В конце каждой главы учебника приведены контрольные вопросы. В библиографический список включена литература, рекомендуемая для детального изучения тематики систем искусственного интеллекта.

Глава 1. Понятие искусственного интеллекта

Термин интеллект (intelligence) происходит от латинского intellectus — ум, рассудок, разум, мыслительные способности человека. Интеллект определяет способность человека решать (интеллектуальные) задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.

А. Тьюринг предложил тест для определения искусственного интеллекта. В разных комнатах находятся люди и компьютерная система. Они не видят друг друга, но обмениваются информацией (например, с помощью электронной почты). Если в процессе диалога не удается установить, что один из участников – компьютерная система, то такую систему можно считать обладающей интеллектом. Также А. Тьюринг предложил разрабатывать сначала систему, имитирующую интеллект ребенка, а затем уже ее усовершенствовать до имитации разума взрослого человека.

Существует различные определения искусственного интеллекта (ИИ).

- автоматизация видов деятельности, ассоциируемых с человеческим мышлением, таких как принятие решений, решение проблем, обучение …" (Belman, 1978).
- "Прикладывание новых усилий для того, чтобы сделать думающие компьютеры, … машины с мозгами в полном и дословном смысле" (Hougeland, 1985).
- "Изучение ментальных способностей через использование вычислительных моделей" (Charniak, McDermott, 1985).
- "Искусство создания машин, которые осуществляют функции, требующие интеллекта при реализации их человеком" (Kurzweil, 1990).
- "Область науки, которая имеет дело с объяснением и воспроизведением

интеллектуального поведения в терминах вычислительных процессов" (Schalkoff, 1990).

- "Изучение вычислений, которые делают возможным распознавать, размышлять и действовать (Winston, 1992).
- "Область информатики, имеющая дело с автоматизацией интеллектуального поведения" (Luger, Stubblefield, 1993).

Такое разнообразие определений объясняется тем, что понятие "искусственный интеллект" может рассматриваться в различных контекстах: как наука, набор технологий, реализованная модель разума, раздел информатики, занимающийся изучением того, как работает мозг [4].

В области моделирования процесса мышления человека сформировалось два направления: логическое (информационное) и нейрокибернетическое (нейробионическое).

Первое основано на выявлении и применении в интеллектуальных системах различных логических и эмпирических приемов (эвристик), применяемых человеком для решения практических задач. Его достоинствами являются:

• возможность относительно легкого понимания работы системы;

• легкость отображения процесса рассуждений системы на ее интерфейс с пользователем на естественном языке или каком-либо формальном языке;

• достижимость однозначности поведения системы в одинаковых ситуациях.

Недостатками этого подхода являются:

• трудность и неестественность реализации нечетких знаков (образов);

• трудность (или даже невозможность) реализации адекватного поведения в условиях неопределенности (недостаточности знаний, зашумленности данных, не точно поставленной цели и т.п.);

• трудность и неэффективность распараллеливания процесса решения задач.

Второе направление основано на построении самоорганизующихся систем, состоящих из множества элементов, функционально подобных нейронам головного мозга. Это направление началось с концепции формального нейрона Мак-Каллока-Питтса и исследований Розенблатта с различными моделями перцептрона (системы, обучающейся распознаванию образов).

В нейрокибернетическом направлении, задав простые алгоритмы адаптации и структуру искусственной нейронной сети, можно получить систему, настраивающуюся на поведение сколь угодно сложное и адекватное решаемой задаче. В дальнейшем эти направления вылились в системы, основанные на знаниях, или интеллектуальные системы (ИнС).

Начало исследований в области ИИ (конец 50-х годов) связывают с работами Ньюэлла, Саймана и Шоу по решению интеллектуальных задач через формирование гипотез о путях их решения с последующей проверкой. Способы решения задач развивались на основе расширения математической и символической логики. Моделированию же человеческого мышления придавалось второстепенное значение. На дальнейшие исследования в области ИИ большое влияние оказало появление метода резолюций Робинсона, основанного на доказательстве теорем в логике предикатов и являющегося исчерпывающим методом доказательства. При этом определение термина ИИ претерпело существенное изменение. Целью исследований, проводимых в направлении ИИ, стала разработка программ, способных решать "человеческие задачи".

Первоначально развитие ИИ включало всевозможные игры, головоломки, математические задачи. В конце 60-х годов искусственный интеллект стал применяться в реальных проблемных средах, что привело к постановке задачи создания интегральных роботов. Это привело к необходимости исследований проблем представления знаний и средств зрительного восприятия, построения сложных планов поведения в динамических средах, общение с роботами на

естественном языке, Сочетание дополняющих друг друга возможностей человека и компьютера позволяет обойти трудности, путем перекладывания на человека тех функций, пока еще не доступны для ЭВМ.

Исторически сложились три основных направления в области ИИ. В рамках первого объектом исследований являются структура и механизмы работы мозга человека, а конечная цель заключается в раскрытии тайн мышления (сильный интеллект). Необходимыми этапами исследований в этом направлении являются: построение моделей на основе психофизиологических данных, проведение экспериментов с ними, выдвижение новых гипотез относительно механизмов интеллектуальной деятельности, совершенствование моделей и т. д. Второе направление связано с моделированием интеллектуальной деятельности и на компьютерах, создание алгоритмического и программного обеспечения вычислительных машин, позволяющего решать интеллектуальные задачи не хуже человека (слабый интеллект).Третий подход ориентирован на создание человеко-машинных, (интерактивных интеллектуальных систем), объединяющих возможности естественного и искусственного интеллекта. Оптимальное распределение функций между естественным и искусственным интеллектом и организация диалога между человеком и машиной. Сюда же входит моделирование на основе так называемого ”черного ящика” [1].

На данный момент отсутствует однозначное определение ***интеллектуальной системы (ИнС)***. Каждый разработчик и пользователь вкладывает свой смысл в это понятие с учетом своего понимания и предметной области. Задачи, алгоритмы, решения которых уже получены, не являются интеллектуальными, как отмечает специалист в области ИИ М. Минский. Решения таких задач могут выполнить человек, вычислительная машина (должным образом запрограммированная), без представления о сущности самой задачи. Примеры таких задач: вычислительные задачи: решение системы линейных алгебраических уравне-

ний, численное интегрирование дифференциальных уравнений и т. д. По своему функциональному назначению ИнС отличаются от традиционных вычислительных систем принципом работы и наличием свойств интеллектуальности. Основные различия между ИнС и традиционными системами представлены в табл.1.1.

Таблица 1.1 - Сравнение традиционных и интеллектуальных систем

Характеристики	Традиционные системы	Интеллектуальные системы
Тип информации	Данные	Знания
Тип обработки информации	Числовая	Символьная
Модель представления информации	Математическая	Эвристическая
Способ обработки информации	Алгоритм	Вывод на знаниях
Получаемое решение задачи	Оптимальное	Правдоподобное
Модификации системы	Редкие	Частые

Решение интеллектуальных задач в ИнС осуществляется путем использования символьного подхода к построению моделей человеческого мышления и общения. Символы или знаки в звуковой или зрительной форме, упорядоченные синтаксическими правилами, образуют языки естественных и точных наук, отображающих семантику (смысл) человеческих отношений.

Основными особенностями применения символьной обработки знаний в ИнС являются:

а) использование знаний специалистов (экспертов) для решения поставленных задач;

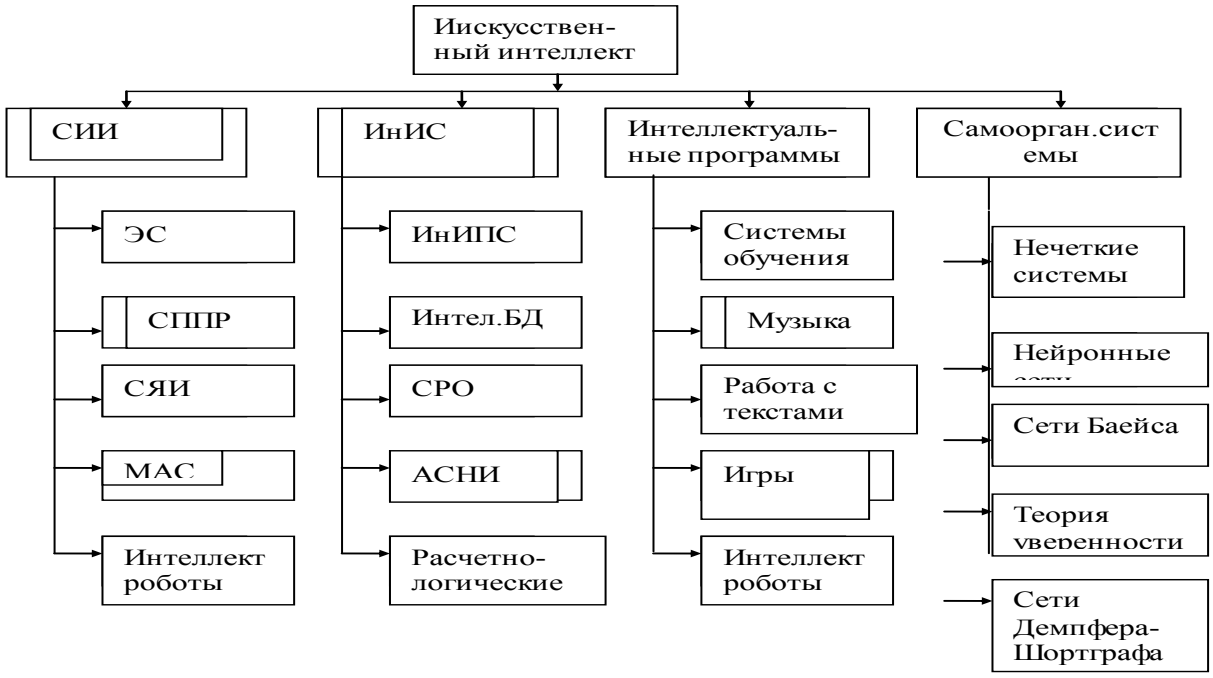
б) применение эвристических методов вывода на знаниях вместо традиционных математических алгоритмов решения задач;

в) использование процедур поиска, выбора, классификации и распознавания знаний, которые существенно отличаются от стандартных процедур, используемых при решении традиционных вычислительных задач;

г) усложнение структур знаний за счет перехода от традиционных линейных представлений числовой информации к представлениям в виде деревьев.

Интеллектуальные системы можно разделить на четыре класса (Рис.1.1)

- интеллектуальные информационные системы (**ИнИС**)
- Системы искусственного интеллекта (**СИИ**)
- Интеллектуальные программы (**ИнП**)
- Интеллектуальные самоорганизующиеся и эвристические системы



СП

Рис.1.1- Классификация Интеллектуальных систем.

Интеллектуальные информационные системы ИнИС.

Интеллектуальные информационные системы это естественный результат развития обычных информационных систем, сосредоточившие наукоемкие технологии подготовки информации и формирования решений, исходя из полученной информационной системой данные. ИнИС– это взаимосвязанная совокупность средств, методик и алгоритмов, имеющая возможность хранения, обработки и выдачи информации, а также самостоятельной настройки своих параметров в зависимости от состояния внешней среды (исходных данных) и специфики решаемой задачи. Признаком такой системы является наличие развитой базы данных с мощным СУБД и интеллектуальным интерфейсом.

К ИнИС можно отнести следующие системы:

- *интеллектуальные информационно-поисковые системы*, интеллектуальный интерфейс которых обеспечивает пользователю возможность входа в систему с запросами на обычном профессиональном языке, а наличие базы знаний и решателя позволяет в информационных системах такого уровня обеспечивать поиск ответа и тогда, когда прямого ответа на его в базе данных нет.

Интеллектуальные базы и хранилища данных в отличие от традиционных БД позволяют обеспечивать выборку необходимой информации, не присутствующей в явном виде, а выводимой из совокупности хранимых данных. Информационные хранилища данных представляют собой хранилища объемов значимой информации, регулярно извлекаемой из оперативных баз данных. Хранилище данных - это предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных, применяемых для поддержки процессов принятия управленческих решений.

- *Распознавание образов и обработка визуальной информации (СРО).* Это одно из самых ранних направлений ИИ, в котором распознание объектов осуществляется на основании применения специального математического аппарата,

обеспечивающего отнесение объектов к классам, а классы описываются совокупностями определенных значений признаков.

В задачах СРО исходные изображения преобразуются в данные другого типа, например в текстовые описания. При синтезе изображений на вход системы поступает алгоритм построения изображения, а выходными данными являются графические объекты (системы машинной графики).

- *Обучающие системы***.** Обучающие системы (тьюторы) имеют в своей базе знаний все необходимые знания для организации процесса обучения. Тьюторы работают в интерактивном режиме обратной связи с пользователем. Когнитивная графика тьюторов делает возможным не только текстовое обучение, но и обучение с помощью зрительных образов и мультфильмов, в создании которых участвует тьюторы и сам обучаемый.

- *Системы обработки текстов (гипертекстовые системы)*. Используются для реализации поиска по ключевым словам в базах данных с текстовой информацией. Для более полного отражения различных смысловых отношений терминов требуется сложная семантическая организация ключевых слов, в которых механизм поиска сначала работает с базой знаний ключевых слов, а затем - с самим текстом. Аналогичным образом проводится поиск мультимедийной информации, включающей кроме текста графическую информацию, аудио- и видео-образы.

- *Системы контекстной помощи.* Относятся к классу систем распространения знаний. Такие системы являются, как правило, приложениями к документации. Системы контекстной помощи - частный случай гипертекстовых и ЕЯ-систем. В них пользователь описывает проблему, а система на основе дополнительного диалога конкретизирует ее и выполняет поиск относящихся к ситуации рекомендаций. В обычных гипертекстовых системах, наобо-

рот, компьютерные приложения навязывают пользователю схему поиска требуемой информации.

- *Системы когнитивной графики***.** Ориентированы на общение с пользователем ИИС посредством графических образов, которые генерируются в соответствии с изменениями параметров моделируемых или наблюдаемых процессов. Когнитивная графика позволяет в наглядном и выразительном виде представить множество параметров, характеризующих изучаемое явление, освобождает пользователя от анализа тривиальных ситуаций, способствует быстрому освоению программных средств и повышению конкурентоспособности разрабатываемых ИИС. Применение когнитивной графики особенно актуально в системах мониторинга и оперативного управления, в обучающих и тренажерных системах, в оперативных системах принятия решений, работающих в режиме реального времени.

- *Системы автоматического доказательства теорем и решения задач.* Одной из первых областей применения систем искусственного интеллекта были задачи автоматического построения вычислительной процедуры (такие системы называются автоматическими решателями задач) и задачи автоматического доказательства теорем. В процессе создания таких программ были более глубоко изучены и получили дальнейшее развитие теории доказательств и эффективных методов их построения.

Интеллектуальные программы**.** Сюда входят:

- *программное обеспечение систем ИИ.* Инструментальные средства для разработки интеллектуальных систем включают в себя специальные языки программирования, ориентированные на обработку символьной информации (LISP, SMALLTALK, РЕФАЛ), языки логического программирования (PROLOG), языки представления знаний (OPS 5, KRL, FRL), интегрированные программные

среды, содержащие арсенал инструментальных средств для создания систем ИИ (KE, ARTS, GURU, G2), а также оболочки экспертных систем (BUILD, EMYCIN, EXSYS Professional, ЭКСПЕРТ), которые позволяют создавать прикладные экспертные системы, не прибегая к программированию.

- *Игры и машинное творчество*. Машинное творчество охватывает сочинение компьютерной музыки, стихов, интеллектуальные системы для изобретения новых объектов. Создание интеллектуальных компьютерных игр является одним из самых развитых коммерческих направлений в сфере разработки программного обеспечения. Кроме того, компьютерные игры предоставляют мощный арсенал разнообразных средств, используемых для обучения.

Интеллектуальные самообучающиеся и эвристические системы. Самообучающиеся системы основаны на методах автоматической классификации ситуаций из реальной практики, или на методах обучения на примерах (прецедентах). Примеры реальных ситуаций составляют так называемую обучающую выборку, формируемую в течение определенного периода. Элементы обучающей выборки описываются множеством классификационных признаков.

Стратегия "обучения с учителем" предполагает задание специалистом для каждого примера значений признаков, показывающих его принадлежность к определенному классу ситуаций. При обучении "без учителя" система должна самостоятельно выделять классы ситуаций по степени близости значений классификационных признаков.

Нейронные сети представляют собой классический пример технологии, основанной на примерах.

Системы искусственного интеллекта. При построении систем, основанных на знаниях (СИИ), используются знания в виде конкретных правил решения различных задач. Работа таких систем заключается в имитации человеческого

искусства анализа неструктурированных и слабоструктурированных проблем. В данной области рассматриваются вопросы инженерии знаний, т.е. разработки методов Создание практических СИИ, построенных на основе достижений теории искусственного интеллекта, обладающих рядом особенностей, отличающих их от систем, создававшихся до развития работ по искусственному интеллекту.

Под системой искусственного интеллекта (СИИ) понимают систему практического применения способную принимать решение в условиях:

— ограниченной информации;

— неопределенности;

— многомерного пространства;

— необходимости распознавать ситуацию (образы, сцены и т.д.);

— различных стадий жизненного цикла объектов (процессов);

— проектирования, производства, эксплуатации;

— динамических фактов, влияющих на решение задачи;

— формализации и представления знаний;

— адаптации, самообучения, самоорганизации и т.д.

Таким образом, если СИИ имеет необходимую математическую, алгоритмическую, программную и инструментальную поддержку в принятии решения, то она имеет интеллектуальную поддержку при решении широкого класса разнообразных задач.

Существующие СИИ можно разбить на два класса: СИИ общего назначения и специализированные.

СИИ общего назначения не только исполняют заданные процедуры, но на основе метапроцедур поиска генерируют и исполняют процедуры решения новых задач. Эксперт формирует знания (данные и правила), описывающие выбранное приложение (прикладные задачи, предметную область). Затем на осно-

вании этих знаний, заданной цели и исходных данных метапроцедуры системы генерируют и исполняют процедуру решения конкретной задачи (рис. 1.2)[10].

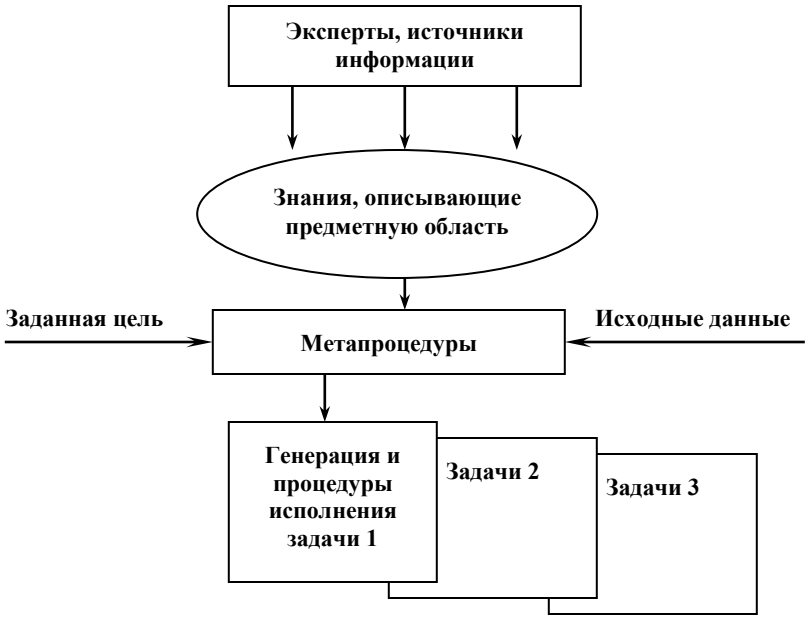


Рис.1.2- Технология использования СИИ общего назначения.

Данную технологию называют технологией систем, основанных на знании, или технологией инженерии знаний [1]

К специализированным отнесены СИИ, выполняющие решение фиксированного набора задач. При использовании таких систем требуется наполнить их данными, соответствующими выбранному приложению (прикладным задачам, предметной области). До недавнего времени при разработке специализированных СИИ использовалась технология процедурного программирования для обеспечения высокой эффективности. При этом ограничивалась адаптивность СИИ при изменении окружении, что необходимо при решении многих интеллектуальных задач. Для устранения этого недостатка разработчики используют технологию инженерии знаний.

В области развития СИИ выделено шесть основных направлений развития.

1. *Представление знаний.* В рамках этой проблемы решаются задачи, связанные с формализацией и представлением знаний в памяти СИИ. Для этого разрабатываются специальные модели представления знаний и языки для описания знаний, выделяются различные типы знаний. Изучаются источники, из которых СИИ могут черпать знания, и создаются процедуры и приемы, с помощью которых возможно приобретение знаний для СИИ. Проблема представления знаний для СИИ чрезвычайно актуальна, так как СИИ — это система, функционирование которой опирается на знания о проблемной области, которые хранятся в ее памяти.

2.*Манипулирование знаниями***.** Для того чтобы знаниями можно было пользоваться при решении задач, СИИ должна уметь:

- оперировать знаниями;
- пополнять знания (с помощью разрабатываемых способов на основе неполного описания знаний);
- классифицировать хранящиеся в системе знания;
- обобщать по тем или иным разработанным процедурам знания;
- формировать на основе знаний абстрактные понятия;
- осуществлять достоверный и правдоподобный вывод на основе имеющихся знаний с помощью создаваемых методов;
- пользоваться моделями рассуждений, имитирующими особенности человеческих рассуждений.

Манипулирование знаниями и представление знаний — эти два направления тесно связаны друг с другом. Создающаяся в настоящее время теория баз знаний включает исследования, относящиеся как к первому, так и ко второму направлению.

3. *Общение***.** В круг задач этого направления входят:

- проблема понимания связных текстов;

- понимание речи и синтез речи;
- теория моделей коммуникации между человеком и СИИ.

*4. Восприятие***.** Это направление включает:

- проблемы анализа трехмерных сцен;
- разработку методов представления информации о зрительных образах в базе знаний;
- создание методов перехода от зрительных сцен к их текстовому описанию и методов обработки перехода;
- разработку процедур когнитивной графики (КГ).

*5. Обучение***.** Основная черта СИИ — это способность к обучению, то есть решение задач, с которыми они ранее не встречались.

6. *Поведение***.** Так как СИИ должны действовать в некоторой окружающей среде, то необходимо разработать специальные поведенческие процедуры, которые позволили бы им адекватно взаимодействовать с окружающей средой, другими СИИ и людьми.

Эти вопросы, возникающие перед конечным пользователем и инженером по знаниям, необходимо решить на этапе предварительного системного анализа конкретной предметной области (ПрО).

Примерами специализированных СИИ являются интеллектуальные диалоговые системы и прикладные экспертные системы.

Широкую известность получили **экспертные системы (ЭС)**. Свое название они получили в связи с тем, что основу знаний, хранящихся в их памяти, составляют сведения, приобретенные от экспертов – профессионалов в тех или иных предметных областях. Знания эксперта используются для создания базы

знаний ЭС, моделирующих мышление эксперта при решении интеллектуальной задачи.

Интеллектуальные системы принятия решений (ИСПР) осуществляют планирование и принятие решений по управлению объектами и процессами в сложных (экстремальных) условиях с применением моделей человеческого мышления и поведения. ИСПР эффективно сочетают в себе человеческий интеллект, информационные технологии, интеллектуальное программное обеспечение, а также опыт и знания человечества, накопленные к настоящему времени. ИСПР представляют пользователю возможные варианты решений поставленной задачи с оценкой последствий реализации каждого варианта решения, а в отдельных случаях могут принимать и самостоятельные решения.

Интеллектуальные естественно-языковые системы (ИЕЯС) Проблемы компьютерной лингвистики и машинного перевода разрабатываются в ИИ с 1950-х гг. Системы машинного перевода с одного естественного языка на другой обеспечивают быстроту и систематичность доступа к информации, оперативность и единообразие перевода больших потоков, как правило, научно-технических текстов. Системы машинного перевода строятся как интеллектуальные системы, поскольку в их основе лежат базы знаний в определенной предметной области и сложные модели, обеспечивающие дополнительную трансляцию «исходный язык оригинала - язык смысла - язык перевода».

Для динамических **мультиагентных систем** характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе, например, через "доску объявлений"

Для многоагентных систем характерны следующие особенности:

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

Контрольные вопросы

1. Что представляет тест Тьюринга для определения интеллекта
2. Какие существуют направления искусственного интеллекта
3. В чем заключаются отличия традиционных и интеллектуальных систем
4. Для каких целей используются специализированные СИИ
5. Что понимается под системой искусственного интеллекта.

Глава 2. Иженерия знаний

Представление данных и знаний. Информация, с которой имеет дело компьютер, разделяется на процедурную и декларативную. Процедурная ин-

формация овеществлена в программах, которые выполняются в процессе решения задач, декларативная в данных, с которыми эти программы работают [7].

Для удобства сравнения данных и знаний можно выделить основные формы (уровни) существования знаний и данных. Знания имеют сложную структуру, и переход от данных к знаниям является следствием развития и усложнения информационных структур, поэтому правильно представлять знания как правила перехода от одних данных к другим [10].

Схема компьютерных программ с данными, представляемая как **ДАННЫЕ + АЛГОРИТМЫ = ПРОГРАММА**
заменяется на новую архитектуру, основу которой составляет БЗ и интерпретатор БЗ (машина логического вывода):

ЗНАНИЯ + ВЫВОДЫ = СИСТЕМА.

Данные. Параллельно с развитием структуры компьютеров происходило развитие информационных структур для представления данных. Появились способы описания данных в виде: векторов, матриц, списочных структур, иерархических структур, структур, создаваемых программистом (абстрактных типов данных).

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

При обработке на ЭВМ данные преобразуются, 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.

База данных (БД) как естественнонаучное понятие характеризуется двумя основными аспектами: информационным и манипуляционным. Первый аспект отражает структуризацию данных, являющуюся наиболее подходящей для обеспечения информационных потребностей предметной области.

Способы структуризации данных получили название моделей данных. Каждая такая модель характеризуется определенными средствами и методами структурирования данных. Наиболее известны иерархическая, сетевая и реляционная модели данных [7].

Табличные (реляционные) структуры данных. Во многих областях человеческой деятельности используется термин «таблица». При этом в каждом конкретном случае в него вкладывается свой смысл. Наряду со смыслом, или сущностью, таблицы обладают теми или иными формами их представления. Понятие таблицы включает несколько аспектов: прагматику, семантику и синтаксис. Прагматика задает цели рассмотрения таблиц. Исходя из прагматики определяется их сущность (семантика). Синтаксический аспект таблиц связан с построением их форм, наиболее подходящих для заданного восприятия (здесь может учитываться ориентация на человека, устройство-автомат и др.). Связь между этими аспектами таблиц выражается как принцип подчиненности: синтаксический аспект подчинен семантическому, а последний зависит от прагматического.

Реляционная модель (разработанная Э. Ф. Коддом в 1970 г.) выгодно отличается от всех других моделей данных за счет простоты, математической строгости и практической полезности. Реляционная алгебра, служит аппаратом построения одних структур данных из других, уже ранее построенных или первоначально заданных, нахождением различных типов зависимостей между компонентами структур данных. Такие зависимости, называемые еще ограничениями целостности БД, играют роль информационных инвариантов ПрО и в

связи с этим должны поддерживаться в БД на протяжении всего периода ее использования.

Сначала БД развивались по экстенсиональному пути, то есть включали широкий круг вопросов, не придавая особого значения их внутреннему содержанию. Поэтому вопросы построения и исследования основополагающих концепций БД, познания их глубинной природы затрагивались слабо. Только в реляционном подходе был сделан интенсиональный шаг в развитии БД.

В ходе дальнейшего развития концепций структур данных, средств манипулирования ими и базирующихся на них моделей БД больше внимания стало уделяться семантическому аспекту структур данных и средств их обработки.

Знания имеют более сложную структуру, чем данные (метаданные). При этом знания задаются как экстенсионально (т. е. через набор конкретных фактов, соответствующих данному понятию и касающихся предметной области), так и интенсионально (т. е. через свойства, соответствующие данному понятию, и схему связей между атрибутами).

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

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, подученного в результате практической деятельности. С информационной точки зрения можно дать следующее определение знаний.

При обработке на ЭВМ знания трансформируются аналогично данным [10].

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 — см. далее);

Z5 — база знаний на машинных носителях информации.

Свойства знаний.

1. *Интерпретируемость* обеспечивает интенсиональное определение знаний (через внутренние свойства, связи и структуру), противопоставляемое экстенсиональному (через набор фактов). Например, многие пользователи компьютеров имеют только экстенсиональное представление о процессорах. Знают несколько названий наиболее популярных моделей и факты, что они влияют на производительность компьютера, размещаются на материнской плате и т.д. При этом они не владеют интенсиональным знанием о функционировании процессоров.

2. *Рекурсивная структурированность и связность***.** Рекурсивная структурированность определяется через понятие концепта. Концепт есть семантическое целое понятий. Понятие P_1 и P_2 образуют семантическое целое, если имеет место какое-то из перечисленных условий:

(A) P_1 и P_2 связаны друг с другом как отношение и один из его аргументов;

(B) P_1 и P_2 связаны друг с другом как действие и его носитель или субъект;

(C) P_1 и P_2 связаны друг с другом как функция и ее аргумент (объект и его свойство).

По индукции следует, что множество $P = \{P_1, P_2, ..., P_n\}$ понятий образует семантическое целое, если каждый P_i в P образует семантическое целое как минимум с одним из P_j, где $P_j \in P/\{P_i\}$.

Концепт C есть упорядоченное множество C_i , где C_i - семантическое целое, содержащееся в семантическом целом, представляющем C.

3.Семантическое пространство с метрикой. Это свойство знаний связано

с возможностью их измерения в системе оценок (метрики) [истинно, ложь]. В системе с нечеткой логикой используется бесконечное множество оценок истинности знания. Например, заключение ЭС вида "завтра ожидается дождь с вероятностью 0,8" не является ни строго истинным, ни строго ложным, т.е. характеризуется определенной степенью правдоподобия.

4. *Активность знаний***.** Это свойство знаний "адаптироваться" под изменяющиеся факты (т.е. способность к обучению и самокоррекции).

*5. Функциональная целостность***.** Под функциональной целостностью знаний понимается их непротиворечивость и разрешимость. Непротиворечивость знаний означает невозможность появления в базе знаний двух взаимоисключающих фактов (типа "пациент жив" и "пациент мертв").

Требование разрешимости заключается в том, что любое истинное знание, формализуемое в базе знаний системы, может быть выведено в ней с помощью машины вывода.

6.Независимость означает невозможность вывода единого знания из другого формальным способом.

7. *Ситуативность.* Наличие ситуативных связей определяет совместимость тех или иных знаний, хранимых в памяти. В качестве таких связей могут выступать отношения времени, места, действия, причины и пр.

Инженерия знания появилась в связи с проблемами выявления и формализации знаний. Поэтому появление этого направления в искусственном интеллекте обусловлено задачами извлечения знаний, приобретения знаний, представления знаний, манипулирование знаниями и служит основой для создания экспертных систем и других практических систем искусственного интеллекта (СИИ). Инженерии знаний получение и структурирование знаний о некоторой предметной области, формирование для нее поля знаний и разработка баз знаний

Центральным понятием на стадиях получения и структурирования знаний является поле знаний.

Поле знаний – это условное неформальное описание основных понятий и взаимосвязей между сущностями предметной области, выявленных из различных источников, в том числе, полученных от экспертов, в виде графов, диаграмм, таблиц, текстов и т.п.

Специалист, накапливающий знания в определенной предметной области и поэтому компетентный, называется экспертом, что позволяет распознавать и оценивать ситуации. Средний специалист в конкретной предметной области помнит от 50 до 100 тыс. чанков (символьный образ, объединенный в мозге в блоки) и использует их для решения задач и проблем. Этим объясняется представление знаний в ИнС в виде БЗ, как сложных иерархических структур с соответствующими связями между этими структурами. Структура инженерии знаний представлена на рис.2.1.

Рис.2.1- Структура инженерии знаний

Представление знаний – процесс формализованного описания для ввода знаний в БЗ, структуризация знаний необходима для облегчения поиска решений. Описание проводится с помощью языка представления знаний (ЯПЗ). ЯПЗ – знаковая система, в которой описываются объекты и явления (или обобщения) согласно принятому множеству соглашений по знакам, синтаксису (построение, порядок, способ соединения слов и предложений) и семантике (смысловое значение). ЯПЗ обеспечивает возможность формальной записи знаний + оперирование знаниями. Процедура преобразования хаоса знаний в модель знаний представляется следующим образом (Рис.2.2):

1. Восприятие и интерпретация действительности предметной области некоторым экспертом, в результате образуется некоторая модель как семантическое представление действительности и его личного опыта.

2. Вербализация опыта эксперта, при объяснении им своих рассуждений и передачи своих знаний инженеру по знаниям. В результате образуется некоторое текстовое или речевое сообщение. Именно в процессе объяснения эксперт на размытые ассоциативные образы в лабиринтах своей памяти «надевает» четкие словесные ярлыки, т.е. вербализирует знания.

3. Восприятие и интерпретация некоторого сообщения инженером по знаниям. В результате в памяти инженера образуется некоторая модель предметной области.

4. Кодирование и вербализация модели в форме некоторого поля знаний, спроектированного инженером по знаниям для реализации в базе знаний.

Основная задача заключается в обеспечении максимального соответствия между действительным состоянием предметной области и полем знаний. Поле зна-

ний может быть представлено как пирамида, где следующий уровень служит для восхождения на новую ступень обобщения и углубления знаний.

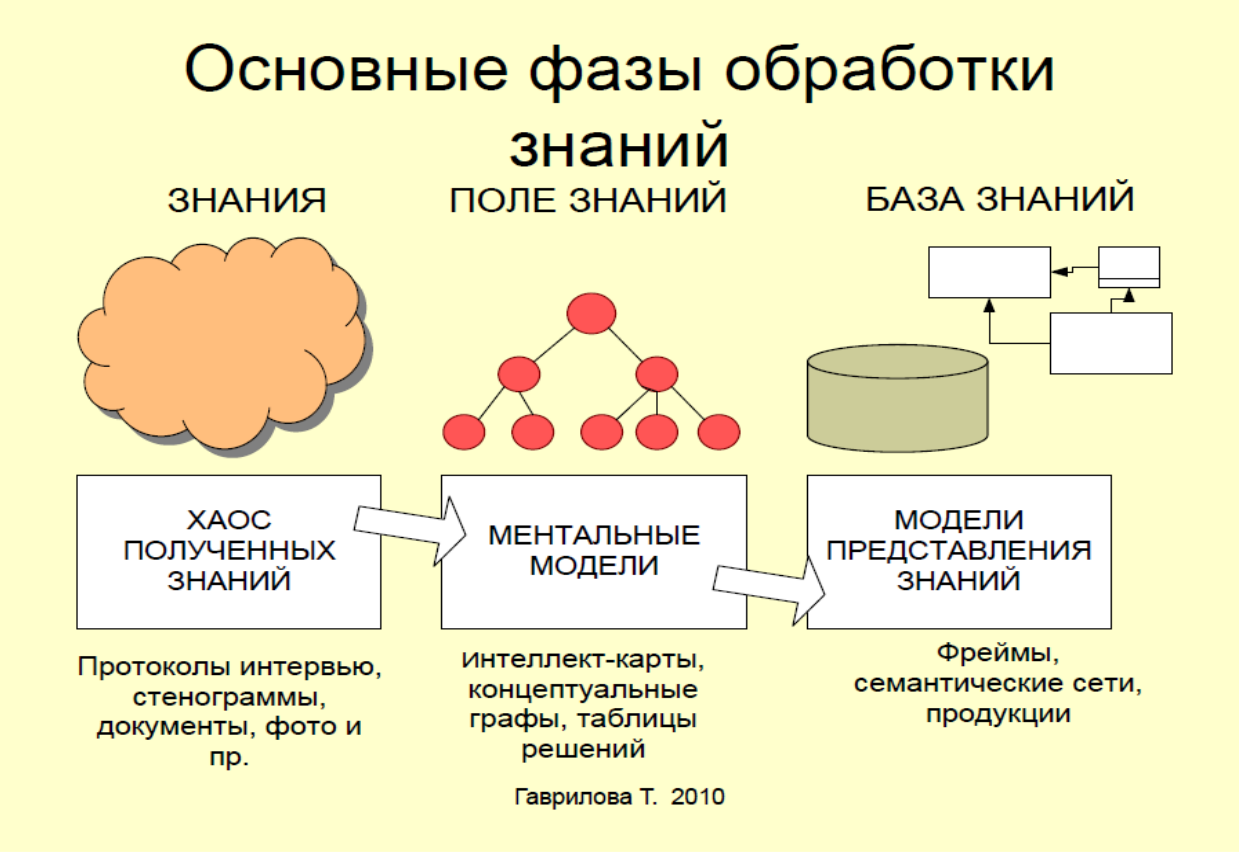


Рис.2.2. Процесс формирования поля знаний

В инженерии знаний используется термин “формирование знаний”, обозначающий процесс анализа данных и выявления скрытых закономерностей с использованием специального математического аппарата и программных средств ЭВМ. Основные методы извлечения знаний представлены на рис.2.3

Классификация методов извлечения знаний (рис.2.3) позволяет инженерам по знаниям, в зависимости от конкретной задачи и ситуации, выбрать конкретный метод. Из предложенной схемы видно, что основной принцип деления связан с источником знаний. Коммуникативные методы охватывают все виды контактов с живым источником знаний - экспертом, а текстологические касаются методов

извлечения знаний из документов (методик, пособий, руководств) и специальной литературы (статей, монографий, учебников). Разделение этих групп методов на верхнем уровне классификации не означает их антагонистичности, обычно инженер по знаниям комбинирует различные методы, например, сначала изучает литературу, затем беседует с экспертами, или наоборот.

В свою очередь коммуникативные методы можно также разделить на две группы: активные и пассивные. Пассивные методы подразумевают, что ведущая роль в процедуре извлечения знаний как бы передается эксперту, а инженер по знаниям только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным самостоятельно рассказать в форме лекции. В активных методах, напротив, инициатива полностью в руках инженера по знаниям, который активно контактирует с экспертом различными способами - в играх, диалогах, беседах за "круглым столом" и т.д.

Пассивные методы на первый взгляд достаточно просты, но на самом деле требуют от инженера по знаниям умения четко анализировать "поток мыслей" эксперта и выявлять в нем значимые фрагменты знаний. Отсутствие обратной связи (пассивность инженера по знаниям) значительно ослабляет эффективность этих методов, чем и объясняется их обычно вспомогательная роль при активных методах.

Активные методы можно разделить на две группы, в зависимости от числа экспертов, отдающих свои знания. Если их число больше одного, то целесообразно помимо серии индивидуальных контактов с каждым применять и методы групповых обсуждений предметной области. Такие групповые методы обычно активизируют мышление участников дискуссий и позволяют выявлять весьма нетривиальные аспекты их знаний. В свою очередь, индивидуальные методы на

сегодняшний день остаются ведущими, поскольку столь деликатная процедура, как "отъем знаний", не терпит лишних свидетелей.

Рис.2.3- Методы извлечения знаний

Все эти методы позволяют сформировать поле знаний на основании следующих последовательностей действий:

1.Определение входных и выходных данных, структура которых существенно влияет на форму и содержание поля знаний.

2.Составление словаря терминов и наборов ключевых фраз, при этом особенно важен словарь терминов.

3.Выявление объектов и понятий, выбор значимых понятий и их признаков.

4.Выявление связей между понятиями, построение сети ситуаций, где связи только намечены, но пока не поименованы.

5.Структуризация понятий с выявлением понятий более высокого уровня обобщения и детализацией на более низком уровне.

6.Построение пирамиды знаний с иерархической лестницей понятий по уровню общности.

7.Определение временных, причинно-следственных и других отношений с их обозначением путем присвоения имен своем связям.

8.Определение стратегий принятия решений. Выявление цепочек рассуждений связывает все сформированные ранее понятия и отношения в динамическую систему поля знаний.

Стратегии получения знаний.

Существуют следующие процедуры формирования знаний (рис.2.4):

- извлечение знаний
- приобретение знаний
- формирование знаний
- вывод новых знаний
- манипулирование знаниями.

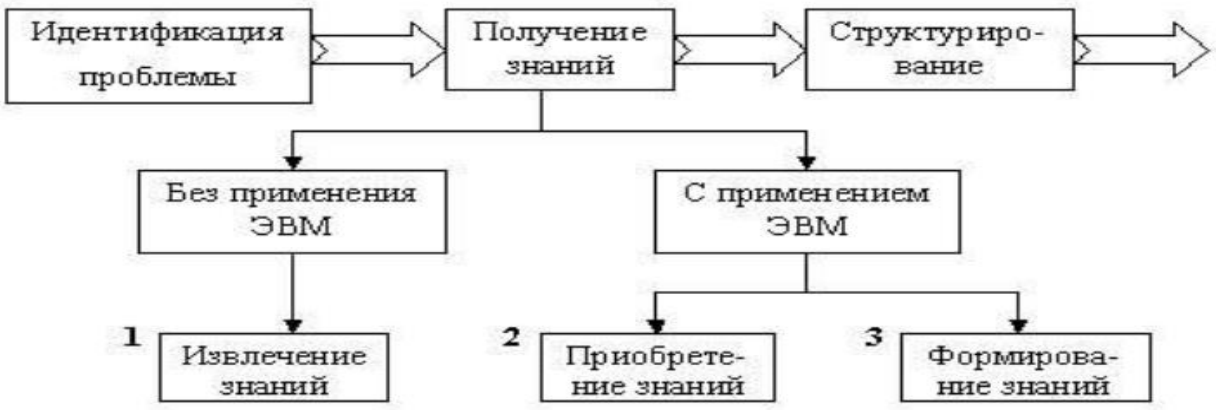


Рис.2.4- Процедуры формирования знаний

Извлечение знаний - это процедура взаимодействия инженера по знаниям с источником знаний (это может быть эксперт и различные источники). С точки зрения разработки СИИ извлечение знаний – это получение информации о предметной области и выражение ее на языке представления знаний. Этот термин выражает смысл процедуры переноса знаний в базу знаний СИИ.

Процесс извлечения знаний – это длительная и трудоемкая процедура, в которой инженеру по знаниям, вооруженному специальными знаниями по когнитивной психологии, системному анализу, математической логике и пр., необходимо воссоздать модель предметной области. Нежелательно извлечение знаний экспертом, потому что ему намного труднее создать модель предметной области вследствие глубины и необозримости информации, которой он обладает. В процессе объяснения инженеру по знаниям эксперт на размытые имеющиеся у него ассоциативные образы надевает четкие словесные ярлыки, т.е. вербализует знания.

Приобретение знаний - способ и процесс автоматизированного построения базы знаний посредством специальной программы (при этом структура знаний заранее закладывается в программу). Эта стратегия требует существенной предварительной проработки предметной области. Системы приобретения знаний начинают обладать готовыми фрагментами знаний в соответствии с заложенными структурами.

Формирование знаний – процесс анализа данных и выявление скрытых закономерностей с использованием специального математического аппарата и программных средств.

Три основных аспекта составляют процедуру извлечения знаний (рис. 2.5): психологический, лингвистический, гносеологический. A = {Al, A2, A3} = {психологический, лингвистический, гносеологический}.

Рис.2.5- Теоретические аспекты инженерии знаний

Психологический аспект является ведущим, поскольку он определяет успешность и эффективность взаимодействия инженера по знаниям (аналитика) с основным источником знаний — экспертом-профессионалом.

Извлечение знаний — это особый вид общения, который можно отнести к духовно-информационному типу. В соответствии с этим выделяют три «слоя» проблем, возникающих при извлечении знаний (рис.2.6): A1 ={S11, S12, S13 } = {контактный, процедурный, когнитивный}.

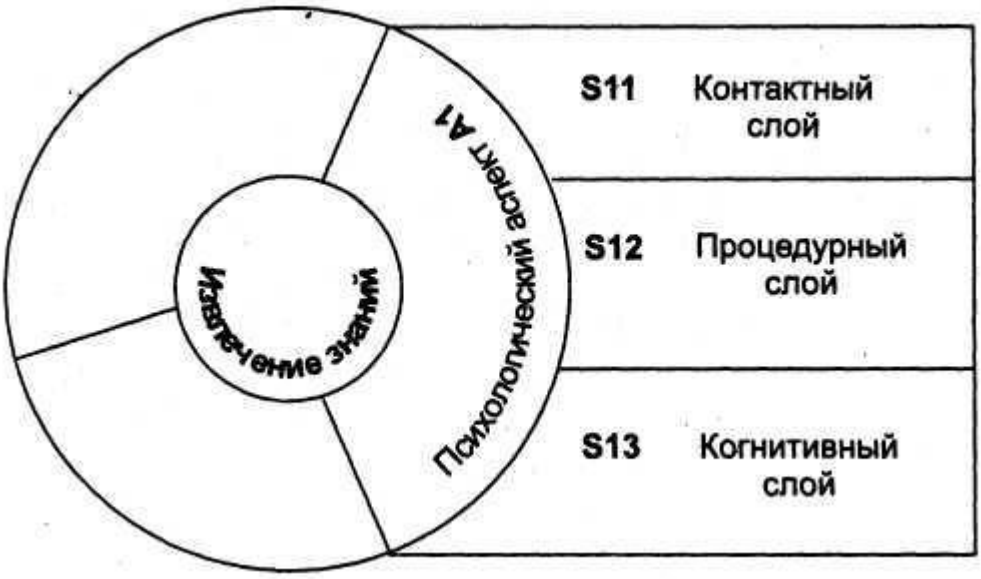


Рис.2.6 – Психологический аспект извлечения знаний

Контактный слой связан с атмосферой и уровнем общения в коллективе разработчиков ЭС. На коллективный процесс влияет атмосфера, возникающая в группе участников.

Процедурный слой касается проведения самой процедуры извлечения знаний. Инженер по знаниям, успешно овладевший наукой доверия и взаимопонимания с экспертом (контактный слой), должен уметь воспользоваться благоприятностью атмосферы. Но для решения проблемы контакта, необходимы профессиональные знания:

Когнитивный слой (англ. cognition - познание)связан со знанием механизмов, при помощи которых человек познает окружающий мир. С позиций когнитивной психологии при извлечении знаний желательно: не навязывать эксперту ту модель представления, которая ему (аналитику) более понятна и естественна.

Лингвистический аспект. Лингвистический (А2) аспект касается исследований языковых проблем, так как язык — это основное средство общения в процессе извлечения знаний. В инженерии знаний выделяют три слоя лингвистических проблем (рис.2.7). А2 = {S21, $22, S23} = {«общий код», понятийная структура, словарь пользователя}.

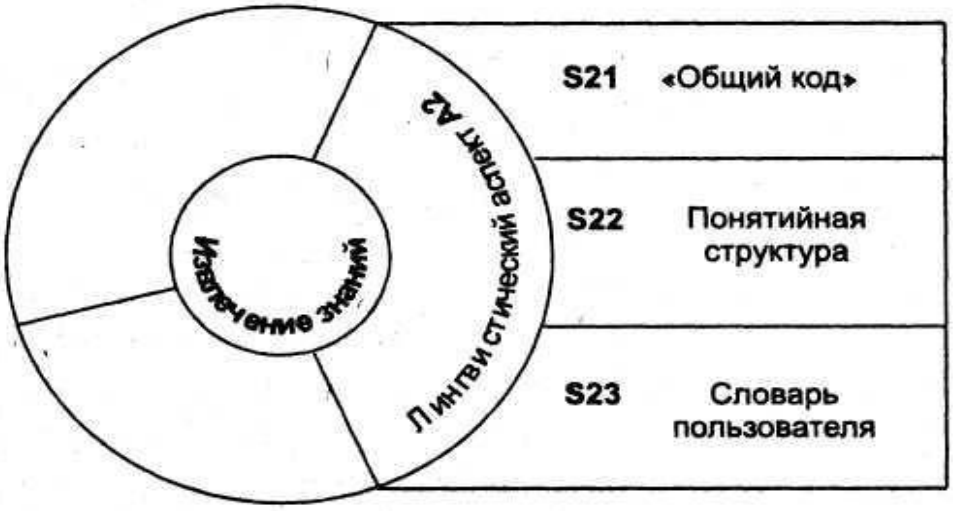


Рис.2.7- Лингвистический аспект извлечения знаний

Проблема «Общего кода. Языки, на которых говорят и размышляют аналитик и эксперт, могут существенно отличаться. Различие языков и обусловливает «языковый барьер» или «языковые ножницы» в общении инженера по знаниям и эксперта.

Язык инженера по знаниям (аналитика) состоит из трех компонентов:

- общенаучной терминологии из его «теоретического багажа»;
- терминов предметной области, которые он почерпнул из специальной литературы в период подготовки;
- бытового разговорного языка, которым пользуется аналитик.

Язык эксперта включает:

- общенаучную терминологию;
- специальную терминологию, принятую в предметной области;
- бытовой язык;
- неологизмы, созданные экспертом за время работы, то есть его профессиональный жаргон.

Бытовой и общенаучный языки у двух участников общения примерно совпадают, но некоторый общий язык, или код, который необходимо выработать партнерам для успешного взаимодействия, должен включать еще специальные терминологии инженера по знаниям и эксперта. В дальнейшем этот общий код преобразуется в некоторую понятийную (семантическую) сеть, которая является прообразом поля знаний предметной области.

Выработка общего кода начинается с выписывания аналитиком всех терминов, употребляемых экспертом, и уточнения их смысла. Фактически это составление словаря предметной области. Затем следуют группировка терминов и выбор синонимов (слов, означающих одно и то же). Разработка общего кода заканчивается составлением словаря терминов предметной области с предварительной

группировкой их по смыслу, т.е. по понятийной близости (это уже первый шаг структурирования знаний).

Понятийная структура. Большинство специалистов по искусственному интеллекту и когнитивной психологии считают, что основная особенность естественного интеллекта и памяти заключается в связанности всех понятий в некоторую сеть. Лингвистическая работа инженера по знаниям по данному слою проблем заключается в построении таких связанных фрагментов с помощью «сшивания» терминов.

Словарь пользователя. Для создания дружественного интерфейса с пользователем возникает необходимость в создании отдельного словаря для пользователей без профессиональных терминов. При внедрении систем искуссвенного интеллекта в основном используется этот словарь.

Гносеологический аспект связан с теорией познания, или теорией отражения действительности в сознании человека. Инженерия знаний дважды гносеологична: действительность сначала отражается в сознании эксперта, а затем деятельность и опыт эксперта интерпретируются сознанием инженера по знаниям, что служит уже основой для построения третьей интерпретации - поля знаний экспертной системы. Процесс познания в сущности направлен на создание внутреннего представления окружающего мира в сознании человека.

В процессе извлечения знаний аналитика в основном интересует компонент знания, связанный с эмпирическими знаниями экспертов, поскольку предметные области именно с таким типом знаний считаются наиболее восприимчивыми к внедрению экспертных систем. Эксперт порождает новые знания, в процессе беседы с аналитиком. Эта заставляет его за частным увидеть общее и построить гносеологические цепочки:

ФАКТ=> ОБОБЩЕННЫЙ ФАКТ=> ЭМПИРИЧЕСКИЙ ЗАКОН=> ТЕОРЕТИЧЕСКИЙ ЗАКОН.

Такой подход согласуется со структурой самого знания, которое имеет два уровня:

- эмпирический (наблюдения, явления);

- теоретический (законы, абстракции, обобщения).

Методологическая структура познания представляется, как последовательность этапов (рис.2.8)

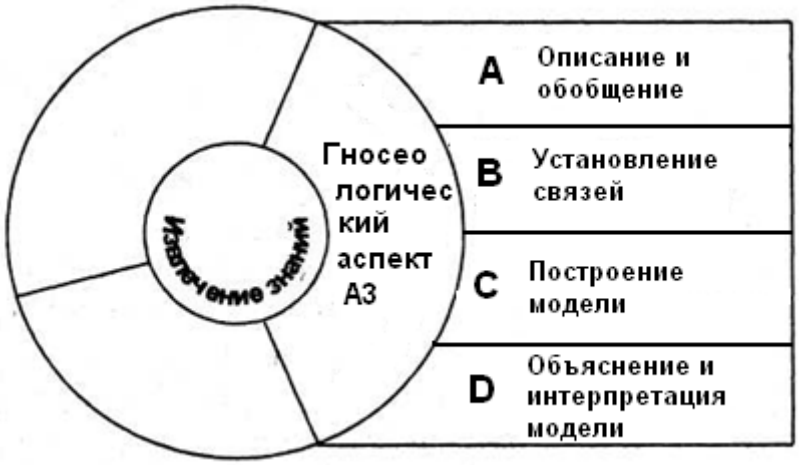


Рис.2.8- Структура познания.

*Описание и обобщение фактов***.** Это результат бесед аналитика с экспертом.

На этом этапе факты лучше просто собирать и как бы бросать в "общий котел". Опытный инженер по знаниям часто сразу пытается найти "полочку" или "ящичек" для каждого факта, тем самым косвенно готовясь к этапу концептуализации.

*Установление связей и закономерностей***.** В голове эксперта связи установлены, хотя часто и неявно; задача инженера - выявить каркас умозаключений эксперта. Реконструируя рассуждения эксперта, инженер по знаниям может опираться на две наиболее популярные теории мышления - логическую и ассоциативную. Основными операциями такого мышления являются ассоциации, приобретенные на основе различных связей; припоминание прошлого опыта; пробы и ошибки со случайными успехами; привычные ("автоматические") реакции и пр.

*Построение модели***.** Для построения модели, отражающей представление субъекта о предметной области, необходим специализированный язык, с помощью которого можно описывать и конструировать те идеализированные модели мира, возникающие в процессе мышления.

Объяснение и интерпретация модели. Этот завершающий этап структуры познания является одновременно и частичным критерием истинности полученного знания. Если выявленная система знаний эксперта полна и объективна, то на ее основании можно делать прогнозы и объяснять любые явления из данной предметной области, получать новые закономерности и объяснять случаи, не указанные в явном виде в базе знаний.

Контрольные вопросы

1. Чем данные отличаются от знаний?
2. Что такое поле знаний?
3. Какие свойства знаний существуют?
4. В чем особенность инженерии знаний?
5. В чем заключаются теоретические аспекты инженерии знаний?

Глава 3. Модели представления знаний в системах искусственного интеллекта

Эффективность функционирования СИИ во многом зависит от модели представления знаний для формального описания реального мира, и области их использования. Выбор модели представления знаний представляет баланс между декларативным (ДП) и процедурным представлением (ПП). Различие между ДП и ПП можно выразить как различие между «знать что» и «знать как».

ПП основано на предпосылке, что интеллектуальная деятельность есть знание проблемной среды, вложенное в компьютерные программы, то есть знание о том, как можно использовать те или иные сущности.

ДП основано на предпосылке, что знание неких сущностей («знать что») не имеет глубоких связей с процедурами, используемыми для обработки этих сущностей. При использовании ДП считается, что интеллектуальность базируется на некотором универсальном множестве процедур, обрабатывающих факты любого типа, и на множестве специфических фактов, описывающих частную область знаний.

Основное достоинство ДП по сравнению с ПП заключается в том, что в ДП нет необходимости указывать способ использования конкретных фрагментов знания. Простые утверждения могут использоваться несколькими способами, и может оказаться неудобным фиксировать эти способы заранее. Указанное свойство обеспечивает гибкость и экономичность ДП, так как позволяет по-разному использовать одни и те же факты.

В ДП знание рассматривается как множество независимых или слабозависимых фактов, позволяющих осуществлять модификацию знаний и обучение простым добавлением или устранением утверждений. Для ПП проблема модификации значительно сложнее, так как здесь необходимо учитывать, каким образом используется данное утверждение. Необходимость использования достоинств ДП и ПП привело к разработке формализмов, использующих смешанное представление, то есть декларативное представление с присоединенными процедурами или процедурное представление в виде модулей с декларативными образцами. В наиболее совершенном виде эта проблема реализована в объектно-ориентированном подходе.

Модели представления знаний обычно делят на ***логические*** (формальные) и ***эвристические*** (формализованные). В основе ***логических моделей*** представле-

ния знаний лежит понятие формальной системы (теории). Примерами формальных теорий могут служить исчисление предикатов и любая конкретная система каких-либо правил. В логических моделях, как правило, используется исчисление предикатов первого порядка, дополненное рядом эвристических стратегий. Эти методы являются системами ***дедуктивного типа,*** так как в них используется модель получения вывода из заданной системы посылок с помощью фиксированной системы правил вывода. Дальнейшим развитием предикатных систем являются системы ***индуктивного типа,*** в которых правила вывода порождаются системой на основе обработки конечного числа обучающих примеров.

Эвристические модели имеют разнообразный набор средств, передающих специфические особенности проблемных областей. Поэтому эвристические модели превосходят логические по возможности адекватно представить проблемную среду, и по эффективности используемых правил вывода. К эвристическим моделям, используемым в экспертных системах, можно отнести ***сетевые, фреймовые, продукционные, онтологические*** и ***объектно-ориентированные*** модели.

В настоящее время для баз знаний СИИ применяются следующие модели представления знаний:

1. Логические модели
2. Продукционные модели;
3. Фреймовые модели;
4. Семантические сети;
5. Онтологические модели.

Логические модели. Одним из способов представления знаний является язык математической логики, формально описывающий сущности предметной области и связи между ними. Языки высказываний и логики предикатов используют

легко формализуемые конструкции, в отличии естественного языка. В общем виде логическая модель знаний - это множество отношений логики предикатов первого порядка, называемых логическими формулами, имеющих два значения - истина или ложь.

Язык логики предикатов использует выражения, описывающие:

- понятия и объекты изучаемой предметной области;
- свойства объектов и понятий,
- поведение и отношения между ними.

В терминах логики предикатов первые два типа слов называется термами, а третий - предикатами.

Термы представляют собой средства для обозначения интересующих нас сущностей, а предикаты выражают отношения между сущностями, обозначаемых с помощью термов.

При логическом программировании пользователь описывает предметную область совокупностью предложений в виде логических формул, затем манипулируя этими предложениями, формируется необходимый для решения задач вывод.

Терм — это знак (символ) или комбинация знаков (символов), являющаяся наименьшим значимым элементом языка (рис. 3.1).

Рис.3.1 -Конструкции языка логики предикатов

К термам относятся константы, переменные и функции (структуры):

- Константа применяется для обозначения конкретных объектов реального мира.
- Переменные используются для обозначения одного из возможных объектов реального мира или совокупности этих объектов.
- Функции (структуры) - последовательность из нескольких констант или переменных, заключенных в круглые скобки, которые следуют за функциональным символом (функтором). Функторы обозначают операторы, которые после воздействия на объект возвращают некоторое значение.

Предикат — это логическая функция, выражающая отношение между своими аргументами и принимающая значение "истина", если это отношение имеется, или "ложь", если оно отсутствует.

Заключенная в скобки последовательность из n термов, перед которой стоит предикатный символ, называется n-местным (или n-арным) предикатом, который принимает значения "истина" или "ложь" в соответствии со значением термов, являющимися его аргументами.

В логике предикатов сложным предложениям естественного языка соответствуют предикатные формулы, которые образуются из атомарных предикатов и

логических связок. Наиболее часто в логическом программировании используются связки "И", "НЕ" "ЕСЛИ". Комбинируя логические связки и кванторы, можно рекурсивно определить составную формулу логики предикатов, называемую правильно построенной формулой (ППФ).

Применительно к естественному языку (ППФ) описывает предложение общего вида. Множество всех предложений, построенных согласно данным правилам, образуют язык логики предикатов первого порядка.

Язык логики предикатов задается следующими классами символов:

- переменные, обозначаемые через **x, y, z, v, u, ...,**
- константы, обозначаемые посредством **a, b, c, d, ...,**
- функциональные символы, представляемые как **f, g, h, ...,**
- символы отношений **p, q, r, s, ...,**
- символы пропозициональных констант: TRUE (истина) и FALSE (ложь)
- логические операторы (связки): - (отрицание, НЕ), $\vee$ (дизъюнкция, ИЛИ), & (конъюнкция, И), $\rightarrow$ (импликация, ЕСЛИ ...ТО), $\leftrightarrow$ (эквиваленция, ЕСЛИ И ТОЛЬКО ЕСЛИ)
- кванторы: $\exists$ (существование), $\forall$ (всеобщности)
- круглые скобки (,) и запятую ",".

Каждый символ функции и отношения характеризуется числом аргументов данной функции (отношения) называемы арностью. Например, функция **sin(x)** является одноместной, **f(x^2, x, c)** - трехместной и т.п.

Особенностью логических моделей является единственность теоретического обоснования и возможность реализации системы формально точных определений и выводов. Основные задачи, решаемые на логических моделях, следующие:

- установить или опровергнуть выводимость некоторой формулы (в об-

щем случае эта задача алгоритмически неразрешима);

- доказательство полноты/неполноты некоторой формально логической системы, представленной множеством логических формул;
- установление выполнимости системы логических формул (нахождение интерпретирующей функции) или отыскание контрпримера, опровергающего их;
- определение следствий из заданной системы формул;
- доказательство эквивалентности двух формально-логических систем;
- поиск решения задачи на основе доказательства теоремы существования решения и др.

Продукционные модели. Продукции соответствуют навыкам решения задач в долгосрочной памяти человека и подобно навыкам в долгосрочной памяти эти продукции не изменяются при работе системы. Они вызываются по "образцу" для решения данной специфической проблемы. Рабочая память (база данных) продукционной системы соответствует краткосрочной памяти, или текущей области внимания человека. Содержание рабочей области после решения задачи не сохраняется.

Работа продукционной системы инициируется начальным описанием (состоянием) задачи. Из продукционного множества правил выбираются правила, пригодные для применения на очередном шаге. Эти правила создают так называемое конфликтное множество. Для выбора правил из конфликтного множества существуют стратегии разрешения конфликтов, которые могут быть и достаточно простыми, например, выбор первого правила, а могут быть и сложными эвристическими правилами. Продукционная модель в чистом виде не имеет механизма выхода из тупиковых состояний в процессе поиска. Она продолжает работать, пока не будут исчерпаны все допустимые продукции. Практические

реализации продукционных систем содержат механизмы возврата в предыдущее состояние для управления алгоритмом поиска.

Самая простая стратегия разрешения конфликтов сводится к тому, чтобы выбирать первое соответствующее перемещение, которое ведет в еще не посещаемое состояние. Конфликтное множество это простейшая база целей.

Продукционные модели представляют распространенные модели знаний, где знания описываются с помощью правил «если-то» (условие→заключение) в виде:

ЕСЛИ условие (антецедент), То действие (консеквент)

Под условием понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под действием – набор действий, выполняемых при успешном исходе поиска. Внутри консеквента могут также генерироваться и добавляться в базу новые факты, которые были получены в результате вычислений или взаимодействия с пользователем.

Программные средства, оперирующие со знаниями, представленными правилами, получили название продукционных систем (или систем продукции) и впервые были предложены Постом в 1941 году.

Продукция в системе Поста имеет следующую схему:

$$\frac{t_1, t_2, \dots, t_n}{t},$$

где t_1, t_2, ..., t_n – это посылки, а t – представляет собой заключение.

Применение схемы Поста основывается на подстановке цепочек знаков (символов) вместо переменных. Причем вместо вхождения одной и той же переменной подставляется одна и та же цепочка знаков (символов).

При использовании таких моделей у систем, основанных на знаниях, имеется возможность:

- применение простого и точного механизма использования знаний;
- представления знаний с высокой однородностью, описываемых по единому синтаксису.

Эти две отличительные черты и определили широкое распространение методов представления знаний правилами [11]. Общим для систем продукций является то, что они состоят из трех основных элементов:

1. Набора правил, используемых как база знаний (БЗ), которую также называют базой правил.
2. Рабочей памяти, где хранятся факты, касающиеся отдельных задач, а также результаты выводов, получаемых на основе этих предпосылок (динамическая база данных).
3. Механизма логического вывода, использующего правила в соответствии с содержимым рабочей памяти.

Конфигурация систем продукции и взаимосвязь ее основных элементов упрощенно представлена на рис.3.2.

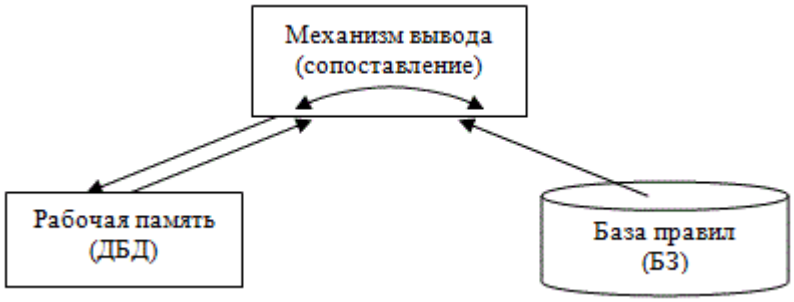


Рис.3.2. Конфигурация продукционной системы

Для знакомства с механизмом функционирования систем продукций рассмотрим пример, описывающий предметную область, связанную с выбором транспорта для отпуска в горах.

Данные, записанные в базу данных, представляют факты:

намерение – отдых

место отдыха - горы

Правила в системах продукций отражают содержимое рабочей памяти. В условной части любого правила “если…” находятся:

- либо одиночные образцы,
- либо несколько условий, соединенных предлогом «И».

После записи в рабочую память всех фактов, а в базу правил – всех правил, описывающих предметную область, система продукций использует механизма вывода (рис.3.3), реализующий применение этих правил.

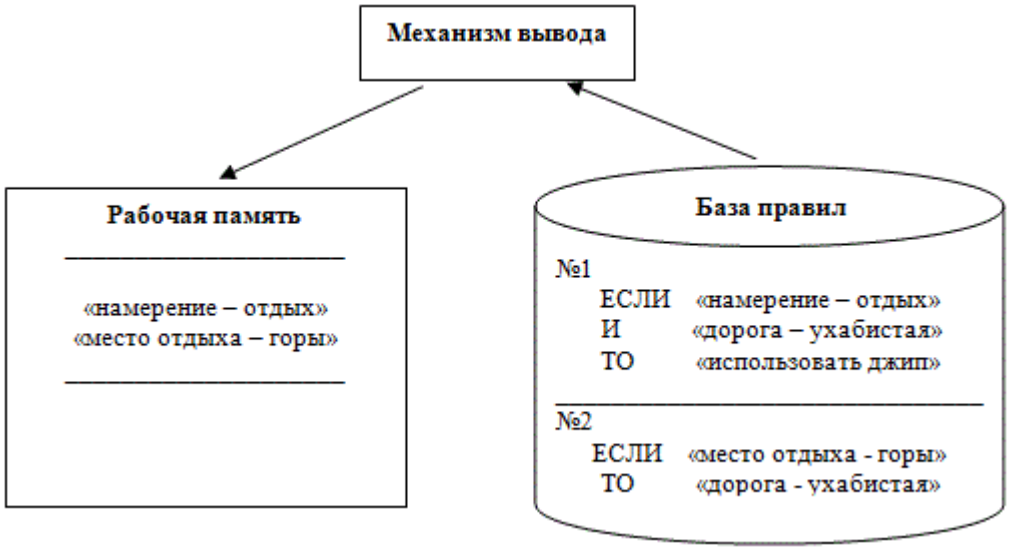


Рис.3.3. Исходное состояние системы продукций

Для этого механизм вывода сопоставляет образцы из условной части правила с образцами, хранимыми в рабочей памяти (БД). Если все образцы имеются в рабочей памяти, условная часть считается истинной, в противном случае – ложной. После запуска в работу последовательность логического вывода будет следующей:

1. Анализ правила, начиная с первого, и определение наличие образца «намерение – отдых» в рабочей памяти и отсутствие в ней образца «дорога – ухабистая».
2. Условная часть правила №1 считается ложной, и механизм вывода переходит к следующему правилу (к правилу №2).
3. Условная часть правила признается истинной, т.к. образец «место отдыха – горы» присутствует в рабочей памяти и механизм вывода переходит к выполнению его заключительной части.
4. Заключительная часть правила №2 «дорога – ухабистая» заносится в рабочую память.
5. После просмотра всех правил происходит вторичное их применение, начиная с первого правила, за исключением тех, которые уже были применены (в примере это правило №2).
6. При повторном сопоставлении правила №1 его условная часть становится истинной ввиду доопределения рабочей памяти, и механизм вывода выполняет его заключительную часть.
7. Заключительная часть «использовать - джип» переносится в рабочую память, а правило №1 исключается из дальнейшего согласования.
8. Правил для сопоставления не остается, и система останавливается.

Если теперь обратиться к рабочей памяти (рис. 3.4), то исходя из посылок что намерение – отдых, место отдыха – горы, результатом является рекомендация использовать джип с пояснением причин данного вывода, определяемых тем, что дорога ухабистая.

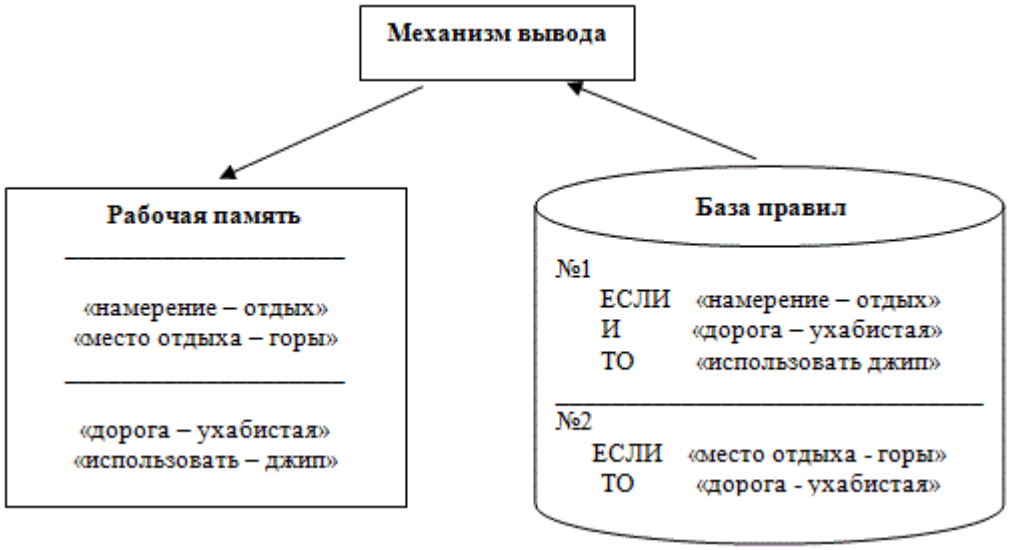


Рис. 3.4- Прямая цепочка рассуждений

В процессе реализации алгоритма механизма логического вывода для данного примера выполнялись действия:

- многократный просмотр содержимого базы правил;
- последовательное применение правил на основе предварительно записанного содержимого рабочей памяти;
- дополнение данных, помещаемых в рабочую память.

Такие выводы называются прямыми (прямая цепочка рассуждений). Другой способ, при котором на основании фактов исследуется возможность применения правила, пригодного для подтверждения, называется обратным выводом (обратная цепочка рассуждений).

Целью запроса к системе является факт установления целесообразности использования Джипа при отдыхе в горах. Считая, что рабочая память содержит образцы:

Намерение- отдых

Место отдыха - горы

а база правил содержит оба, отмеченных выше правила, то целью составления (логического вывода) является доказательство факта ‘использовать джип”

Таким образом, рабочая память и база правил будет иметь исходный вид, аналогичный тому, что приведен на рис.3.5.

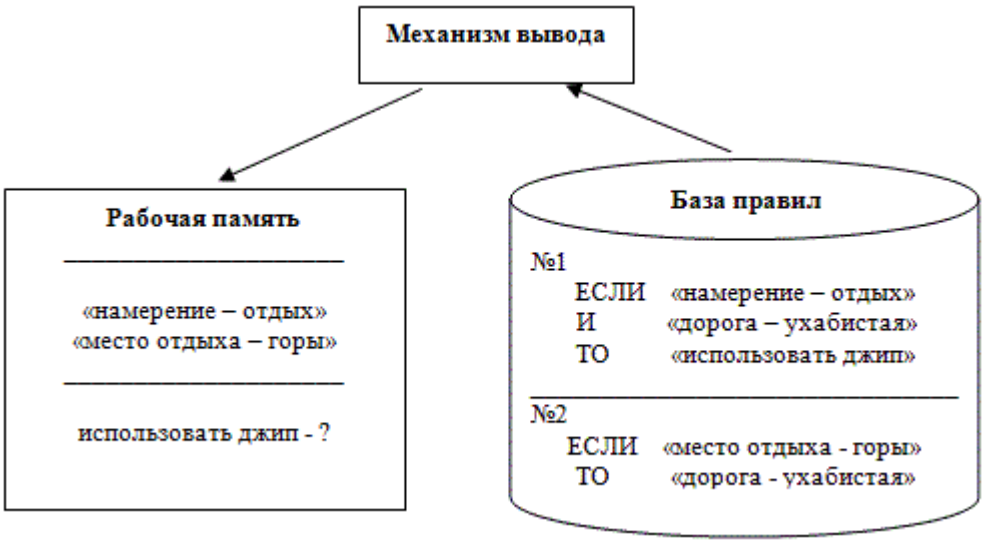


Рис.3.5 - Обратная цепочка рассуждений

Последовательность составления системой продукции следующая:

1. Определяется правило, в котором в заключительной части содержится целевой факт.
2. Исследуется возможность применения первого правила для подтверждения исходного факта.
3. Поскольку образец «намерение – отдых» из условной части правила №1 занесен в рабочую память, то для достижения цели достаточно подтвердить факт «дорога – ухабистая».
4. Образец «дорога – ухабистая» принимается за новую цель, и необходимо найти правило, подтверждающее этот факт.

5. Исследуется возможность применения правила №2. Условная часть этого правила является истинной, т.к. образец «место отдыха – горы» имеется в рабочей памяти;
6. В виду возможности применения правила №2, рабочая память пополнится образцом «дорога – ухабистая» и появляется возможность применения правила №1 для подтверждения цели «использовать – джип».

Таким образом, результатом вывода является подтверждение цели

Использовать джип при условии, что дорога ухабистая

На глобальном уровне управления последовательностью применения правил обычно выделяют две основные стратегии — применять правила в прямом или обратном порядке.

- Прямой порядок означает, что цепь рассуждений строится, отталкиваясь от данных (условий, о которых известно, что они удовлетворяются), к гипотезам (состоянию проблемы, вытекающему из этих условий).
- Обратная цепочка означает, что рассуждения строятся, отталкиваясь от заданной цели (гипотезы, представляющие целевое состояние системы) к условиям, при которых возможно достижение этой цели.

На практике наиболее часто встречаются механизмы логического вывода, опирающиеся на обратную цепочку рассуждений. Это обусловлено их более надежной работой (практически всегда имеется возможность найти цепочку рассуждений от конца до начала) и большей производительностью, что становится особенно заметно при большом количестве продукций.

Основные преимущества продукционных систем:

- простота и гибкость выделения знаний;

- отделение знаний от программы поиска;
- модульность продукционных правил (правила не могут "вызывать" другие правила);
- возможность эвристического управления поиском;
- возможность трассировки "цепочки рассуждений";
- независимость от выбора языка программирования;
- продукционные правила являются правдоподобной моделью решения задачи человеком.

Фреймовая модель представления знаний предложена М.Минским в 1979 г. как структура знаний для восприятия пространственных сцен. Эта модель, имеет психологическое обоснование в виде понятия абстрактного образа. Такой абстрактный образ называется фреймом. Frame (рамка) является единицей представления информации об объекте, который можно описать некоторой совокупностью понятий и сущностей. **Фреймом** называется также и формализованная модель для отображения образа.

Например, понятие «комната» вызывает образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6-20 м^2». Из этого описания ничего нельзя убрать (например, убрав окна, получим уже чулан, а не комнату), но в нем имеются слоты, представляющие незаполненные значения некоторых атрибутов — количество окон, цвет стен, высота потолка, покрытие пола.

В качестве идентификатора фрейму присваивается имя фрейма. Это имя должно быть единственным во всей фреймовой системе.

Фрейм имеет определенную внутреннюю структуру, состоящую из множества элементов, называемых слотами, которым также присваиваются имена. За слотами следуют шпации, в которые помещают данные, представляющие теку-

щие значения слотов. Каждый слот в свою очередь представляется определенной структурой данных. В значение слота подставляются атрибуты, относящиеся к объекту, описываемому этим фреймом.

Структуру фрейма можно представить так:

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).

Ту же запись представим в виде таблицы (таб.3.1), дополнив двумя столбцами.

Таблица3.1 – Структура фрейма

Имя слота	значение слота	способ получения значения	присоединённая процедура

В таблице дополнительные столбцы предназначены для описания способа получения слотом его значения и возможного присоединения к тому или иному слоту специальных процедур. В качестве значения слота может выступать имя другого фрейма. Таким образом, образуются сети фреймов.

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

Модель фрейма является достаточно универсальной, позволяющая отобрать все многообразие знаний о мире:

- через фреймы-структуры, для обозначения объектов и понятий (заем, залог, вексель);
- через фреймы-роли (менеджер, кассир, клиент);
- через фреймы-сценарии (банкротство, собрание акционеров, празднование именин);
- через фреймы-ситуации (тревога, авария, рабочий режим устройства) и др.

Важнейшим свойством теории фреймов является наследование свойств, которое происходит по АКО-связям (A-Kind-Of - это). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

Значением слота может быть практически что угодно: числа, формулы, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов. В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет реализовывать во фреймовых представлениях «принцип матрешки».

Имя фрейма служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Число слотов в каждом фрейме устанавливается проектировщиком системы, при этом часть слотов определяется самой системой для выполнения специфических функций (системные слоты), примерами которых являются: 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.д.

Указатели наследования показывают, какую информацию об атрибутах слотов из фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. Указатели наследования характерны для фреймовых систем иерархического типа, основанных на отношениях типа «абстрактное — конкретное». В конкретных системах указатели наследования могут быть организованы различными способами и иметь разные обозначения:

U (Unique) — значение слота не наследуется;

S (Same) — значение слота наследуется;

R (Range) — значения слота должны находиться в пределах интервала значений, указанных в одноименном слоте родительского фрейма;

O (Override) — при отсутствии значения 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 указателей U и S.

Указатель типа данных показывает тип значения слота. Наиболее употребляемые типы: 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.д.

Значение слота должно соответствовать указанному типу данных и условию наследования.

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 с условием IF-NEEDED запускается, если в момент обращения к сло-

ту его значение не было установлено. Демон типа IF-ADDED запускается при попытке изменения значения слота. Демон IF-REMOVED запускается при попытке удаления значения слота. Возможны также другие типы демонов. Демон является разновидностью связанной процедуры.

В качестве значения слота может использоваться процедура, называемая методом в языках объектно-ориентированного программирования. Присоединенная процедура запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными в единую систему. Эти процедурные знания являются средствами управления выводом во фреймовых системах, причем с их помощью можно реализовать любой механизм вывода.

Пример. Фрейм, описывающий человека.

Фрейм	Человек
Имя слота:	Значение слота
Класс:	Животное
Структурный элемент:	Голова, шея, руки, ...
Рост:	180 см
Масса:	80 кг
Хвост:	Нет
Язык:	Русский, английский, ...

Основным преимуществом **фреймов** как **модели представления знаний** является то, что они отражают концептуальную основу организации памяти че-

ловека, а также ее гибкость и наглядность. Достоинство фреймовых систем представления знаний проявляются в том случае, если родовидовые связи изменяются нечасто и предметная область насчитывает немного исключений. Во фреймовых системах данные о родовидовых связях хранятся явно, как и знания других типов. Значения слотов представляются в системе в единственном экземпляре, поскольку включаются только в один фрейм, описывающий наиболее понятия из всех тех, которые содержит слот с данным именем. Такое свойство систем фреймов обеспечивает экономное размещение базы знаний в памяти компьютера. Также достоинство фреймов состоит в том, что значение любого слота вычисляется с помощью соответствующих процедур или находится эвристическими методами. Поэтому фреймы позволяют манипулировать как декларативными, так и процедурными знаниями. В целях увеличения гибкости системы, декларативные и процедурные знания концептуальных объектов комбинируют;

К недостаткам фреймовых систем относят их относительно высокую сложность, что проявляется в снижении скорости работы механизма вывода и увеличения трудоемкости внесения изменений в родовую иерархию. Поэтому при разработке фреймовых систем уделяют наглядным способам отображения и эффективным средствам редактирования фреймовых структур.

Семантические сети представляют способ структурирования знаний с процедурами их использования и механизмом вывода. Семантической сетью является структура, имеющая определенный смысл как модель представления знаний. Семантическая сеть- это система знаний, имеющая определенный смысл в виде целостного образа сети, узлы которой соответствуют понятиям и объектам, а дуги отношениям между понятиями и объектами

В иерархической структуре понятий существуют отношения, по крайней мере, двух типов:

- отношение включения или совпадения (IS - A);
- отношение «целое – часть» (PART - OF).

Например, в предложении “ноутбук ” (IS - A) – “компьютер” имеет место отношение включения, иначе ноутбук представляет собой один из элементов множества устройств, составляющих класс компьютеров.

Для этих отношений характерным является то, что экземпляры понятий нижнего уровня содержат все атрибуты понятий верхнего уровня. Это свойство называется наследованием атрибутов между уровнями иерархии (IS - A). Это означает, что ноутбук, как понятие более низкого уровня, будет обладать всеми свойствами (атрибутами), определенными для понятия компьютер.

Отношение «целое – часть» можно иллюстрировать предложением, которое характеризует то, что экземпляры понятия «процессор» являются частью любого экземпляра понятия «компьютер», иначе “процессор” (PART - OF) “компьютер”.

Отношения типа (PART - OF) позволяют определить некоторые общие свойства (набор атрибутов) для конкретного класса понятий. Общим свойством всех компьютеров является наличие в их составе процессоров. И это свойство будет наследоваться всеми экземплярами понятия «ноутбук».

Наиболее часто используется графическое представление семантических сетей в виде диаграммы. Так, предложение «все ласточки – птицы» можно представить графом, содержащим две вершины, соответствующие понятиям, и дугу, указывающую отношение между этими понятиями (рис. 3.6.).

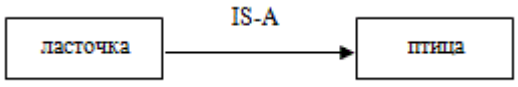


Рис.3.6- Простейшая семантическая сеть

Если ласточка имеет конкретное имя, например, "Ласта", то семантическая сеть может быть расширена (рис.3.7).

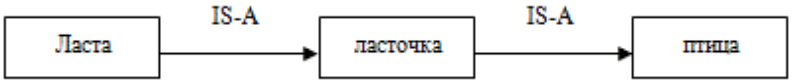


Рис.3.7- Семантическая сеть с иерархией двух понятий

Наряду с тем, что с помощью данной сети описаны два факта:

“Ласта – ласточка”

“ласточка – птица”

из нее можно вынести, используя отношение наследования, новый факт:

“Ласта – птица”

Этот пример показывает, что способ представления семантической сетью позволяет легко делать выводы благодаря иерархии наследования между уровнями.

Семантическими сетями можно также представлять знания, касающиеся атрибутов объекта. Например, факт “Птица имеет крылья” можно отобразить в виде рис. 3.8.

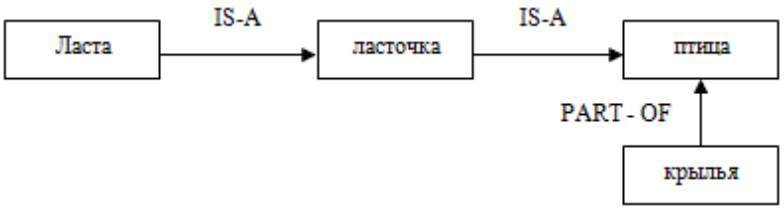


Рис. 3.8 – Семантическая сеть с понятиями и атрибутами

Это означает, что, используя отношения (IS – A) и (PART–OF) можно вывести новый факт, а именно

“Ласта имеет крылья”

Важнейшей концепцией формализма семантических сетей является иерархия понятий и связанное с ней наследование атрибутов между уровнями иерархии (IS - A).

Семантическая сеть представляет собой ориентированный граф с помеченными (поименованными) дугами и вершинами. Основными элементами сети являются вершины и дуги. При этом вершинам семантической сети соответствуют понятия, события и свойства.

- Понятия — представляют собой сведения об абстрактных или физических объектах предметной области или реального мира.
- События — представляют собой действия, происходящие в реальном мире, и определяются:
    - указанием типа действия;
    - указанием ролей, которые играют объекты в этом действии.
- Свойства — используются для уточнения понятий и событий. Применительно к понятиям они описывают их особенности и характеристики (цвет, размер, качество), а применительно к событиям — продолжительность, время, место.

Дуги графа семантической сети отображают многообразие семантических отношений, которые условно можно разделить на четыре класса.

- Лингвистические отношения отображают смысловую взаимосвязь между событиями, между событиями и понятиями или свойствами.

    - Логические отношения — это операции, используемые в исчислении высказываний (алгебре логики): дизъюнкция, конъюнкция, инверсия, импликация.

    - Теоретико-множественные отношения — это отношения подмножеств, отношение части и целого, отношение множества и элемента. Примерами таких отношений являются "IS-A" и "PART-OF"

- Квантизированные отношения выражают отношения, описываемые посредством квантизаторов.

Пример семантической сети, отображающей связи понятий при описании знаний о структуре понятия предприятие, рис.3.9.

Рис.3.9- Пример семантической сети

Так для понятия «предприятие» в этой сети:

- определен класс, к которому оно принадлежит, и все свойства которого оно наследует («Юридическое лицо»);
- выделено 3 свойства, которые выделяют это понятие из всех остальных понятий класса «Юридическое лицо»;
- определен экземпляр данного понятия (объекта), а именно «Завод «Салют».

Преимущества использования семантических сетей:

- описание понятий и событий производится на уровне, очень близком к естественному языку;
- обеспечивается возможность сцепления различных фрагментов сети;
- отношение между понятиями и событиями образуют достаточно небольшое и хорошо формализованное множество;

- для каждой операции над данными и знаниями можно выделить из полной сети, представляющей всю семантику (или все знания), некоторый ее участок, который охватывает необходимые в данном запросе смысловые характеристики.

Онтологическая модель знаний. Онтологии, как способ представления знаний, позволяет приложениям распознавать семантические отличия, которые понятны для людей, но не известны компьютеру, и поэтому онтологии позволяют эффективно обрабатывать сложную и разнообразную информацию.

Под онтологией понимается система понятий некоторой предметной области, представляемая как набор сущностей, соединенных различными отношениями. Определения онтологии рассматриваются с разных точек зрения. В современных информационных технологиях наиболее часто упоминается и используется определение онтологии, сформулированное Н. Грубером: «Онтология – это спецификация концептуализации». Под «концептуализацией» понимается строгое описание системы понятий, объектов и других сущностей и отношений, связывающих их друг с другом. Концептуализация – это абстрактное, упрощенное видение мира, который мы хотим представить для каких-то целей

В общем виде структура онтологии представляет собой набор элементов четырех категорий:

- понятия;
- отношения;
- аксиомы;
- отдельные экземпляры;

Понятия рассматриваются как концептуализации класса всех представителей некой сущности или явления (например, Животное, Чувство). Классы (или понятия) являются общими категориями, которые могут быть упорядочены иерар-

хически. Каждый класс описывает группу индивидуальных сущностей, которые объединены на основании наличия общих свойств.

Отношения связывают различного рода понятия (например, Длина, Местоположение), которые связывают воедино классы и описывают их. Самым распространенным типом отношений, использующимся во всех онтологиях, является отношение категоризации, то есть отнесение к определенной категории. Этот тип отношений имеет ряд других названий, встречающийся в различных исследованиях:

- таксономическое отношение;
- отношение IS-A;
- класс – подкласс;
- родовидовое отношение;
- отношение a-kind-of.

Аксиомы задают условия соотнесения категорий и отношений, они выражают очевидные утверждения, связывающие понятия и отношения. Под аксиомой можно понимать утверждение, вводимое в онтологию в готовом виде, из которого могут быть выведены другие утверждения. Они позволяют выразить ту информацию, которая не может быть отражена в онтологии посредством построения иерархии понятий и установки различных отношений между понятиями.

В качестве примера аксиомы можно привести следующее высказывание: «Если X смертен, то X когда-нибудь умрет». Аксиомы позволяют в дальнейшем осуществлять умозаключения в рамках онтологии и представлять ограничения, накладываемые на какие-либо отношения, делающие возможным выведение умозаключений.

Составляющие онтологии подчиняются своеобразной иерархии. На нижнем уровне этой иерархической лестницы находятся экземпляры, конкретные индивиды, выше идут понятия, то есть категории. На уровень выше располагаются отношения между этими понятиями, а обобщающей и связующей является ступень правил или аксиом.

Онтологии сильно различаются по ряду параметров, влияющих на их классификацию. Существует разбиение онтологий по количеству и качеству понятий, включаемых в них.

Онтологии верхней зоны обычно насчитывают примерно 100-500 концептов. В них включены наиболее абстрактные категории, обладающие свойством универсальности. В таких онтологиях типичными на верхнем уровне разбиения являются такие понятия, как:

- сущность;
- явление;
- объект;
- процесс;
- роль

Обычно они строятся теоретиками и философами. Преимуществом таких онтологий является возможность их использования во многих областях и даже во многих языках.

Другим типом являются онтологии средней зоны, здесь элементов обычно уже больше (500 – 100 000 концептов). Они представляют мир в целом и в общем случае это неаксиоматизированная область. Для данного вида онтологий требуется выводить большое количество аксиом. Обычно выходом является использование методов автоматизированного вывода аксиом из уже существующих

онтологий. Построением онтологий этого уровня чаще всего занимаются когнитологи и лингвисты.

Онтологии нижней зоны (онтологии предметной области) наиболее обширные. Эти онтологии насчитывают около 200 – 2 000 концептов и описывают конкретные предметные области с их спецификой. При этом круг решаемых задач и вопросов, на которые онтология отвечает, ограничен выбранной областью. Для данного типа онтологий характерно наличие отношений, специфичных для конкретной области.

Наряду с описанным делением все онтологии, как знания, разделены на глубинные и поверхностные. Поверхностные онтологии строятся на поверхностной семантике, они определяют понятия через значения слов. Однако здесь возникает проблема, какое количество смыслов выделять для каждого слова. Глубинные же онтологии используют глубинную семантику.

Онтология для представления знаний и работе с ними относится к онтологиям нижней зоны и описывает конкретную предметную подобласть, с характерными отношениями, являющиеся для нее специфичными.

Для создания онтологии надо сначала перечислить категории, обозначающие сущности или явления в моделируемой области. Затем следует связать эти категории определенными отношениями. И на последнем шаге надо соотнести категориям набор конкретных экземпляров.

Семантика и представление знаний основываются на композиционной гипотезе: можно определить ограниченный набор единичных сущностей («атомов»), а все остальные («молекулы») представлять как комбинацию (композицию) атомов. Существует два подхода: экономный и неэкономный.

При экономном подходе создают небольшое количество элементарных семантически простых концептов, с помощью которых можно объяснить значение более сложных понятий. При этом легко обнаружить связанность понятий, про-

сто определять и осуществлять умозаключения, однако достаточно сложно составлять сложные значения.

Неэкономный подход позволяет создавать любое количество индивидуальных сущностей – столько, сколько захочется создателю онтологии. Это количество может варьироваться от 10 до 100 000 и более. Здесь затруднительно определять связанность понятий, сложно работать с умозаключениями. Данный подход сопровождается, по существу, отказом от композиционной гипотезы, но это влечет за собой и преимущество: отсутствие необходимости составлять сложные значения.

Формальная процедура выявления понятий на базе совокупности слов представляет алгоритм перехода от слов к значениям, а затем к понятиям:

1. Инициализация: Для данного слова соберите несколько десятков предложений, содержащих его. Подберите определения из различных словарей.

2. Расположите значения слова в предварительные, грубо схожие группы

3. Процесс дифференциации: Начните строить дерево, расположив все группы в корне

4. Рассматривая все группы, определите группу, наиболее отличную от других:

- Если вы можете найти одну ясно выделяющуюся группу, выпишите ее наиболее яркое отличие в явной форме – оно послужит отличительным признаком и будет формализовано в виде аксиомы.
- Если отличия, по которым можно далее подразделить группу, не обнаруживаются, остановите работу с этой ветвью и перейдите на другую ветвь
- Если обнаруживается несколько отличий, позволяющих подразделить группу несколькими равнозначными способами, также прекратите работу с этой ветвью и перейдите на другую ветвь.

- Создайте в древесной структуре две новые ветви, расположите новую группу под одной ветвью, а остальные под другой.
- Повторите действия с шага 4, исследуя по отдельности группу/группы под каждой ветвью
- Формирование понятий. Когда ветвление прекращается, конечным результатом является дерево все более дробных отличительных признаков, которые в явном виде перечислены на каждом уровне дерева. Каждый лист становится отдельным понятием, далее не делимым в настоящей задаче (приложении, предметной области). Каждое отличие должно быть формализовано в виде аксиомы, которая срабатывает для ветви, с которой ассоциируется.
- Добавление понятия в онтологию. Начиная с вершины, пройдите каждый узел с ветвлением.

Затем нужно проверить условия. Имеют ли уже созданная и добавляемая ветвь примерно одно и то же значение?

- если да, то объедините их в онтологии в подходящем узле и остановите прохождение этой ветви

- если нет, разделите дерево и повторите шаг 8 для каждой ветви. Процесс повторяется, пока не дойдете до конца.

Существующие онтологии требуют постоянного пополнения и усовершенствования. Поэтому целесообразно использование автоматических и полуавтоматических методов для не только обновления онтологий, но даже для их создания.

Контрольные вопросы

1. К какому уровню онтологий относятся понятия предметной области
2. Как реализуется наследование во фреймах.

3. Что определяют вершины и дуги в семантической сети
4. Что является антецедентом и консеквентом
5. Какую роль играют действия IS-A и PART OF
6. Каким образом представляются знания в логических моделях

Глава 4. Нейронные сети

Существуют системы искусственного интеллекта, которые не требуется заранее знать обо всех закономерностях исследуемой области, но нужно иметь достаточное количество примеров для настройки. После обучения такая система способна получать требуемые результаты с определённой степенью достоверности. В качестве таких адаптивных систем рассматриваются искусственные нейронные сети, представляющие технологии, основанные на примерах. Нейронные сети – обобщённое название математических алгоритмов, обладающих способностью обучаться на примерах, распознавая впоследствии черты встреченных образцов и ситуаций. Нейронная сеть – это кибернетическая модель, представляющая множество простых элементов – нейронов, топология соединения которых зависит от типа сети. Чтобы создать нейронную сеть для решения какой-либо конкретной задачи, следует выбрать способ соединения нейронов друг с другом и подобрать значения параметров межнейронных соединений.

Начало эпохи нейронных сетей считают сороковые годы 20 столетия, когда В. Маккаллок и В. Питтс предложили концепцию информационных вычислений, основанную на двоичных решающих элементах, названных нейронами. Каждый из этих элементов принимает значений 0 или 1 в зависимости от состояния покоя или активности. Некоторое состояние определяется также влиянием соседних нейронов. Новое состояние нейрона определяется суммой взвешенных воздействием других нейронов и выражается линейной комбинацией их выходных значений. В модели принято, что нейрон становится активным при превышении

определенного порога. Веса могут быть положительными или отрицательными, что соответствует возбуждению или торможению нейрона. В. Маккаллок и В. Питтс показали, что такое функционирование сети нейронов позволяет выполнять вычисления, также как компьютеры. Особенность сети в том, что операции могут выполняться параллельно.

Реальные нейроны имеют отличие от нейронов, предложенных Маккаллоком и Питтсом:

- не являются пороговыми устройствами, а имеют нелинейное соотношение между входом и выходом,
- вырабатывают последовательность (фазовые соотношения), а не одиночный импульс,
- не все нейроны имеют одинаковую задержку импульсов,

На первом этапе (40-50 годы) развития нейронных сетей канадский ученый Д. Хебб сформулировал постулат о том, что если два нейрона являются взаимно активными, то их взаимодействие усиливается.

Второй этап развития (50-60 годы) называют золотым веком развития нейронных сетей. Ф. Розенблат рассматривал специальный тип нейронных сетей, названный ими персептрон, как элемент биологической сенсорной информации. Это простейший элемент, состоящий из трех слоев: входного и скрытого и выходного, и нейроны выходного слоя принимают сигналы от входного. Ф. Розенблат разработал итеративный алгоритм определения весов, с помощью которого входной образ преобразуется в требуемый выходной и доказал сходимость этого алгоритма.

Третий этап развития, получивший название «застойный», начался в конце 1960-х гг.), после опубликования работы М. Минского и С. Паперта “Perceptrons: an introduction to computational geometry)” (1969 г.), в которой было доказано, что существует ряд задач (например, “исключающее ИЛИ”), которые

не могут быть решены однослойным персептроном. После этой работы развитие нейронных сетей замедлилось на 10 лет. Ф. Розенблат разработал многослойные сети, но отсутствие алгоритма их обучения не позволило развиваться дальше. Только некоторые ученые, и среди них С. Гроссберг и Т. Кохонен, продолжали исследования, опубликовав около 150 работ.

Четвертый этап соответствует середине 80-тых годов. Появился алгоритм определения весов в многослойных сетях со скрытыми слоями (Back propogation), предложенный в 1974 г. П. Вербозом, но развитый только в 1985 г. Появление этого алгоритма оказало существенное влияние на дальнейшее развитие нейросетевых технологий.

Проблема стабильности-пластичности является одной из самых сложных и трудно решаемых задач при построении искусственных систем, моделирующих представление знаний. Некоторые нейронные системы не приспособлены к запоминанию и отделению новых образов от старых. Поэтому разработаны АРТ и RBF сети, положившие начало новым парадигмам, близким к функционированию человеческого мозга.

Биологическая модель нейрона. Структура искусственных нейронных сетей создана по результатам изучения человеческого мозга, хотя сходство между ними и незначительно. Искусственные нейронные сети имеют такие аналогичные мозгу свойства, как способность обучаться на опыте, основанном на знаниях, делать абстрактные умозаключения и совершать ошибки, что является более характерным для человеческой мысли, чем для созданных человеком компьютеров.

Большая часть мозга остается непознанной. Основные исследования направлены на идентификации функций мозга. Нейрон оказался сложнее, чем представлялось ранее, и до сих пор нет полного понимания процесса его функциониро-

вания. Но все-таки мозг может быть использован в качестве модели для развития искусственных нейронных сетей, на его основе созданы структуры нейронных сетей, решающих интеллектуальные задачи.

По оценкам, человеческий мозг содержит свыше тысячи миллиардов вычислительных элементов, называемых нейронами, связанные сотнями триллионов нервных нитей, называемых синапсами. Эта сеть нейронов отвечает за все явления, которые называются мыслями, эмоциями, познанием, а также за совершение множества сенсомоторных и автономных функций.

Мозг является основным потребителем энергии тела. Включая в себя лишь 2% массы тела, в состоянии покоя он потребляет приблизительно 20% кислорода тела. Даже во время сна расходование энергии продолжается. Потребляя примерно 20 Вт, мозг с энергетической точки зрения очень эффективен. Компьютеры с одной долей вычислительных возможностей мозга потребляют сотни и тысячи ватт и требуют сложных систем для охлаждения [2].

Нейрон является основным строительным блоком нервной системы и содержит клетки, подобные другим клеткам тела; однако определенные отличия выполняют вычислительные функции и функции связи внутри мозга. Нейрон состоит из: тела клетки, синапсов, дендритов и аксонов (рис. 4.1).

Функционально дендриты получают сигналы от других клеток через контакты, называемые синапсами. Затем сигналы проходят в тело клетки, где они суммируются с другими такими же сигналами. Если суммарный сигнал в течение короткого промежутка времени является достаточно большим, клетка возбуждается, вырабатывая в аксоне импульс, который передается на следующие клетки. Поэтому нейрон может находиться в двух состояниях: возбужденном или невозбужденном. Эта схема функционирования, хотя и упрощенная, объясняет большинство известных процессов мозга.

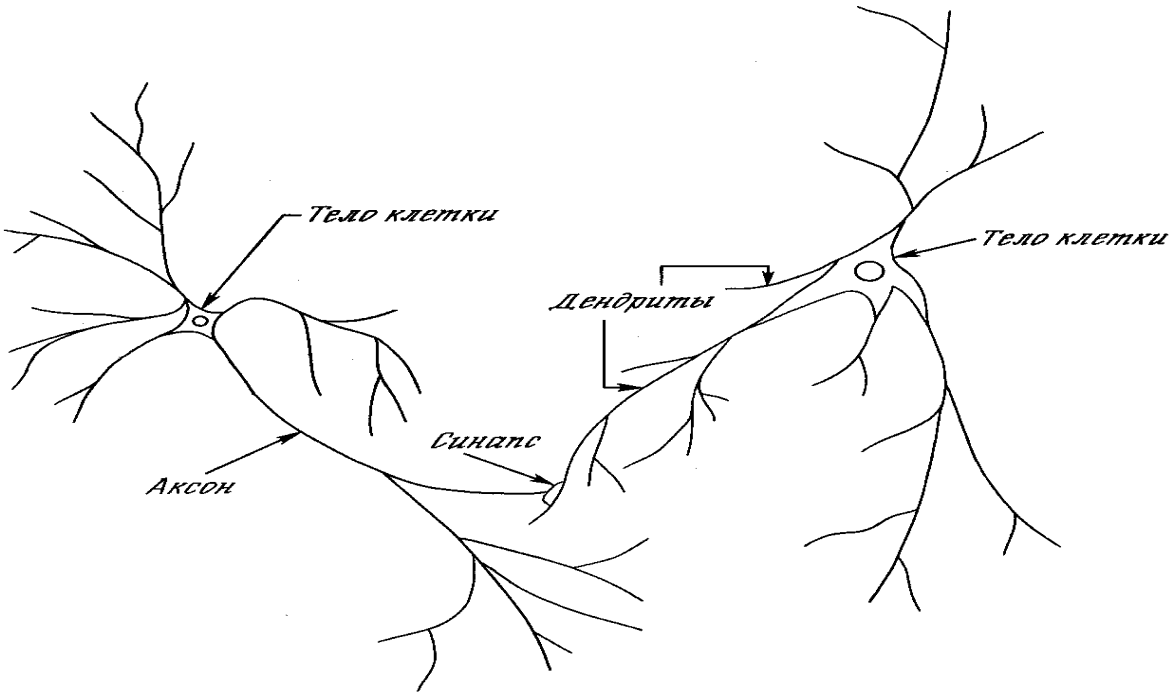


Рис. 4.1- Компоненты нейрона

Классификация нейронных сетей. Нейронные сети различают по структуре сети (связей между нейронами), особенностям модели нейрона, особенностям способа обучения сети [3].

По структуре нейронные сети можно разделить на неполносвязные (или слоистые) и полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями (рис. 4.2).

Неполносвязные нейронные сети (описываемые неполносвязным ориентированным графом и обычно называемые перцептронами) подразделяются на однослойные (простейшие перцептроны) и многослойные, с прямыми, перекрестными и обратными связями. В нейронных сетях с прямыми связями нейроны j-го слоя по входам могут соединяться только с нейронами i-х слоев, где j > i, т. е. с нейронами нижележащих слоев. В нейронных сетях с перекрестными связями допускаются связи внутри одного слоя, т. е. вышеприведенное неравенство заменяется на j > = i. В нейронных сетях с обратными связями ис-

пользуются и связи j-го слоя по входам с i-м при j < i. Кроме того, по виду связей различают перцептроны с регулярными и случайными связями.

По используемым на входах и выходах сигналам нейронные сети можно разделить на аналоговые и бинарные.

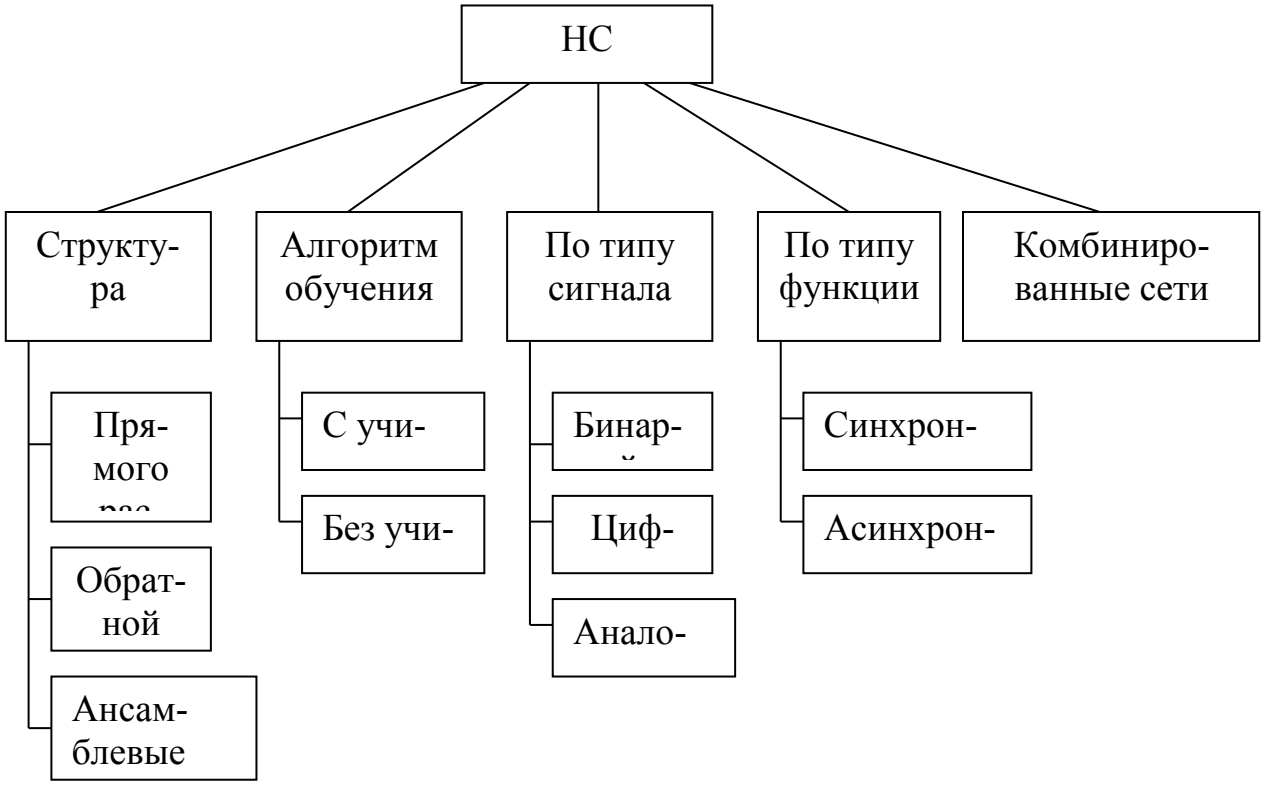


Рис. 4.2 - классификация нейросетевых парадигм

По моделированию времени нейронные сети подразделяются на сети с непрерывным и дискретным временем. Для программной реализации применяется, как правило, дискретное время.

По способу подачи информации на входы нейронной сети различают:

- подачу сигналов на синапсы входных нейронов;
- подачу сигналов на выходы входных нейронов;
- подачу сигналов в виде весов синапсов входных нейронов;
- аддитивную подачу на синапсы входных нейронов.

По способу съема информации с выходов нейронной сети различают:

- съем с выходов выходных нейронов;
- съем с синапсов выходных нейронов;
- съем в виде значений весов синапсов выходных нейронов;
- аддитивный съем с синапсов выходных нейронов.

По организации обучения разделяют обучение нейронных сетей с учителем и без учителя. При обучении с учителем предполагается, что есть внешняя среда, которая предоставляет обучающие примеры (значения входов и соответствующие им значения выходов) на этапе обучения или оценивает правильность функционирования нейронной сети и в соответствии со своими критериями меняет состояние нейронной сети или поощряет (наказывает) нейронную сеть, запуская тем самым механизм изменения ее состояния. Под состоянием нейронной сети, которое может изменяться, обычно понимается:

- веса синапсов нейронов (карта весов – map) (коннекционистский подход);
- веса синапсов и пороги нейронов (обычно в этом случае порог является более легко изменяемым параметром, чем веса синапсов);
- установление новых связей между нейронами (свойство биологических нейронов устанавливать новые связи и ликвидировать старые называется пластичностью).

По способу обучения разделяют обучение по входам и по выходам. При обучении по входам обучающий пример представляет собой только вектор входных сигналов, а при обучении по выходам в него входит и вектор выходных сигналов, соответствующий входному вектору.

По способу предъявления примеров различают предъявление одиночных примеров и «страницы» примеров. В первом случае изменение состояния нейронной сети (обучение) происходит после предъявления каждого примера.

Во втором – после предъявления «страницы» (множества) примеров на основе анализа сразу их всех.

Полносвязные сети представляют сети, в которых имеются все возможные связи: многослойные или слоистые. Сеть разбивается на слои, где каждый слой содержит совокупность нейронов с едиными входными сигналами. Слои нумеруются слева направо, начиная с 0.

Для реализации хотя бы некоторых возможностей мозга необходимо воссоздать его архитектурные особенности. Коннекционистская машина или нейронная сеть является высокосвязной сетью простых процессоров (искусственных нейронов), каждый из которых имеет много входов и много выходов.

В искусственных нейронных сетях каждый нейрон непрерывно обновляет свое состояние порождением значения активации, которая является функцией входных сигналов и внутренних параметров нейрона. Активация используется для генерации выхода через некоторую функцию.

Характеристики искусственных нейронных сетей (ИНС) следующие:

1. Синаптические коэффициенты нейронов (веса) настраиваются посредством подачи на входы обучающих векторов и изменением весов таким образом, чтобы нейрон выдавал требуемые выходные сигналы. Таким образом, нейрон является адаптивной, а не заранее запрограммированной системой.
2. Работу НС целесообразно представлять как эволюцию динамической системы и описывать системой дифференциальных уравнений.
3. Нейронные сети устойчивы к шумам в сигналах и отказам компонентов (нейронов и синапсов). Отказ компонента не влечет отказа всей ИНС в целом, а лишь несколько ухудшает ее характеристики.

4. Характерной особенностью работы нейронных сетей является то, что они способны находить статистические закономерности или особенности в обучающей выборке. Это позволяет нейронной сети отнести новый входной объект к одному из уже усвоенных сетью образов либо к новому классу.
5. Представление основной идеи в нейронной сети осуществляется через вектор активностей нейронов, распределенный по сети, так что каждый нейрон участвует одновременно в представлении многих объектов .

Искусственные нейронные сети состоят из элементов, функциональные возможности которых идентичны большинству элементарных функций биологического нейрона. Несмотря на поверхностное сходство, искусственные нейронные сети демонстрируют свойства, присущие мозгу. Они обучаются на основе опыта, обобщают предыдущие прецеденты на новые случаи и извлекают существенные свойства из поступающей информации, содержащей излишние данные [9].

Схема математического нейрона представлена на рис. 4.3.

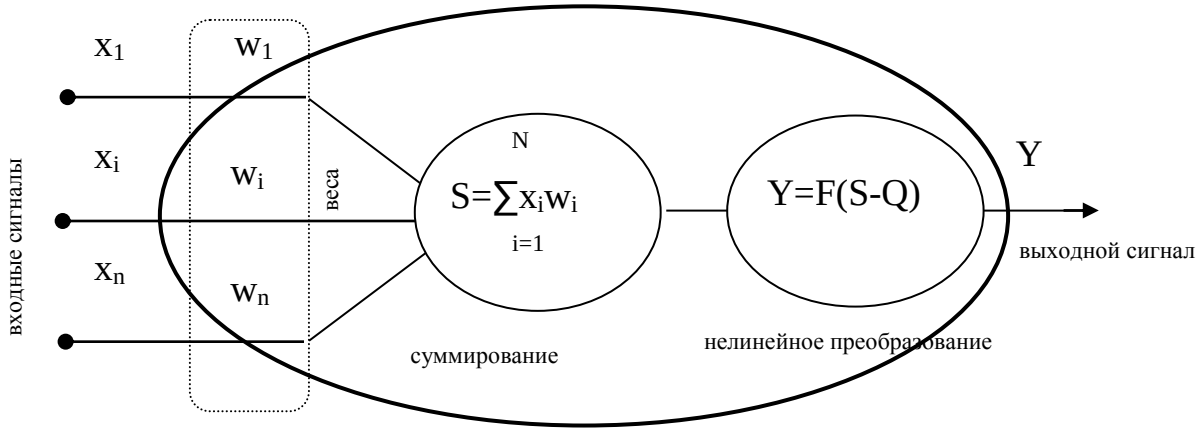


Рис. 4.3 - Математический нейрон

Дендриты получают входной сигнал, представленный вектором x_i. Каждому входу ставится в соответствие весовой (синаптический) коэффициент w_i нейрона, определяющий силу воздействия входного сигнала X на выходной сигнал Y. Затем нейрон обрабатывает поступивший сигнал, производя взвешен-

ное суммирование и нелинейное преобразование, используя для этого активационную функцию (функции активации могут быть разными, наиболее распространенные – линейная, пороговая и сигмойда), аргументом которой будет результат суммирования минус пороговое значение. Это значение определяет уровень сигнала, на который нейрон будет реагировать.

Искусственные нейронные сети могут менять свое поведение в зависимости от внешней среды, чем они и интересны. После предъявления входных сигналов, они самонастраиваются, чтобы обеспечивать требуемую реакцию. Отклик сети после обучения может быть до некоторой степени нечувствителен к небольшим изменениям входных сигналов. Эта внутренне способность видеть образ сквозь шум и искажения важна для распознавания образов, позволяющая преодолеть требование строгой точности, предъявляемое к компьютером, в нечеткой среде,. Искусственная нейронная сеть обобщает автоматически благодаря своей структуре, а не использованием «человеческого интеллекта» в форме специально написанных компьютерных программ.

Искусственные нейронные сети обладают способностью извлекать сущность из входных сигналов.

Пример. Сеть может быть обучена на последовательность искаженных версий буквы «А». После соответствующего обучения предъявление такого искаженного примера приведет к тому, что сеть породит букву совершенной формы, т.е. сеть может порождать то, что никогда не видела - новые знания.

Предложенный в 1986 г. Д. Румельхардом и другими учеными алгоритм обратного распространения ошибки (BP) является эффективным способом обучения нейронной сети произвольной структуры.

Нейронные сети могут реализовываться как программно, так и аппаратно. Постепенно направление нейрокибернетики преобразовалось в нейрокомпьютинг (шестое поколение компьютеров), базирующийся на нейронных сетях

[9]. Сравнительный анализ последовательных компьютеров фон Неймана и биологической нейронной системы, являющейся прообразом нейрокомпьютера, представлен в таблице 4.1.

Таблица 4.1- сравнение вычислительной структуры фон Неймана с нейронной сетью

	Структура фон Неймана	Нейронная сеть
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализована	Распределенная
	Адресация не по содержанию	Адресация по содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	Хранимые программы	Самообучение
Надежность	Высокая уязвимость	Живучесть
Специализация	Численные и символьные операции	Проблемы восприятия
Среда функционирования	Строго определенная	Плохо определенная
	Строго ограниченная	Без ограничений

Однослойные искусственные нейронные сети. Один нейрон способен выполнять простейшие процедуры распознавания, но только соединение нескольких нейронов способно решить практически полезную задачу. Простейшая сеть (рис. 4.4) состоит из группы нейронов, образующих слой [9].

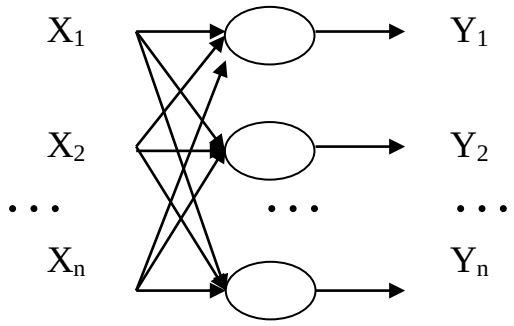


Рис. 4.4 - однослойная нейронная сеть

Процесс обучения нейронной сети можно представить как процесс минимизации функции ошибки. Направление изменения значения функции устанавливают, вычислив производную для функции одного переменного или градиент для функции многих переменных. При минимизации значения функции многих переменных меньшее значение необходимо искать в направлении антиградиента.

В данном алгоритме обучения начальные веса могут быть любыми.

Процесс обучения можно считать завершенным, если достигнута некая заранее установленная минимальная ошибка или алгоритм проработал условленное количество раз.

1-й шаг: инициализация матрицы весов (и порогов, в случае использования пороговой функции активации) случайным образом.

2-й шаг: предъявление нейронной сети образа (на вход подаются значения из обучающей выборки – вектор X, берется соответствующий выход – вектор D).

3-й шаг: вычисление выходных значений нейронной сети (вектор Y).

4-й шаг: вычисление для каждого нейрона величины расхождения реального результата с желаемым:

$$\varepsilon_i = (d_i - y_i),$$

где d_i – желаемое выходное значение на i-нейроне, y_i – реальное значение на i-нейроне.

5-й шаг: изменение весов (и порогов при использовании пороговой функции) по формулам:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \varepsilon_i \cdot x_j,$$
$$\theta_i(t+1) = \theta_i(t) - \eta \cdot \varepsilon_i,$$

где t – номер текущей итерации цикла обучения, w_{ij} – вес связи j-входа с i-нейроном, η – коэффициент обучения (задается от 0 до 1), x_j – входное значение, θ_i – пороговое значение i-нейрона.

6-й шаг: проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то 2 шаг, иначе заканчиваем обучение.

Многослойные нейронные сети. Многослойные сети обладают большими вычислительными возможностями за счет образования каскадов слоев. Выход одного слоя является входом для последующего слоя, такая организация сети образует многослойную нейронную сеть с прямым распространением (рис. 4.5).

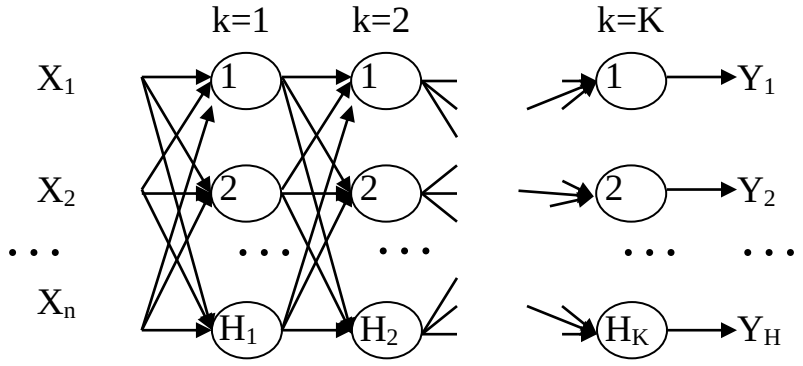


Рис. 4.5 - многослойная нейронная сеть прямого распространения

Многослойная сеть может содержать произвольное количество слоев, и каждый слой состоит из нескольких нейронов, число которых также может быть произвольно (H_k – количество нейронов в слое), количество входов n, количество выходов $H=H_K$– числу нейронов в выходном (последнем) слое [9].

Слои между первым и последним называются промежуточными или скрытыми. Веса в такой сети имеют три индекса: i – номер нейрона текущего слоя, для которого связь входная; j – номер входа или нейрона слоя, для которого связь выходная; k – номер текущего слоя в нейронной сети (для входов, вектора X, k=0).

Обучение алгоритмом обратного распространения ошибки предполагает два прохода по всем слоям сети: прямой и обратный.

При прямом проходе входной вектор подается на входной слой нейронной сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы.

Во время обратного прохода все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого, в результате чего формируется сигнал ошибки. Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название – алгоритм обратного распространения ошибки. Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому.

Алгоритм обучения с обратным распространением ошибки:

1-й шаг: инициализация матриц весов случайным образом (в циклах).

2-й шаг: предъявление нейронной сети образа (на вход подаются значения из обучающей выборки – вектор X и берется соответствующий выход – вектор D).

3-й шаг (прямой проход): вычисление в циклах выходов всех слоев и получение выходных значений нейронной сети (вектор Y):

$$y_i^k = f(\sum_{j=0}^{H_{k-1}} w_{ij}^k \cdot y_j^{k-1}),$$
$$y_j^0 = x_j,$$
$$y_0^{k-1} = 1,$$
$$x_0 = 1,$$

где y_i^k – выход i-нейрона k-слоя, f – функция активации, w_{ij}^k – синаптическая связь между j-нейроном слоя k-1 и i-нейроном слоя k, x_j – входное значение.

4-й шаг (обратный проход): изменение весов в циклах по формулам:

$$w_{ij}^k(t+1) = w_{ij}^k(t) + \eta \cdot \delta_i^k \cdot y_j^{k-1},$$
$$\delta_i^k = (d_i - y_i) \cdot y_i \cdot (1 - y_i)$$

– для последнего (выходного) слоя,

$$\delta_i^k = y_i \cdot (1 - y_i) \cdot \sum_{l=1}^{H_{k+1}} \delta_l^{k+1} \cdot w_l^{k+1}$$

– для промежуточных слоев, где t – номер текущей итерации цикла обучения (номер эпохи), η – коэффициент обучения (задается от 0 до 1), y_i^k – выход i-нейрона k-слоя, w_{ij}^k – синаптическая связь между j-нейроном слоя k-1 и i-нейроном слоя k, d_i – желаемое выходное значение на i-нейроне, y_i – реальное значение на i-нейроне выходного слоя.

5-й шаг: проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не за-

вершено, то 2-й шаг, иначе заканчиваем обучение. Среднеквадратичная ошибка вычисляется следующим образом:

$$\varepsilon = \frac{1}{Q} \cdot \sum_{q=1}^{Q} \sum_{i=1}^{H} (d_i - y_i)^2,$$

где Q – общее число примеров, H – количество нейронов в выходном слое, d_i– желаемое выходное значение на i-нейроне, y_i– реальное значение на i-нейроне выходного слоя.

Для создания и обучения нейронных сетей разработано около 200 пакетов. Наиболее используемые это **STATISTICA** Neural Networks, Neural networks toolbox пакета MATLAB, нейропакет Neuro Pro, Neural Planner.

Контрольные вопросы

1. Что такое дендрит?
2. Что такое аксон?
3. Какую роль играют синапсы?
4. Каким образом формируются первоначальные значения синапсов?
5. Почему возник десятилетний застой в развитии нейронных сетей?

Глава 5. Нечеткие системы искусственного интеллекта

Человеческим суждениям свойственны эмоции, нелогичность, непоследовательность, что приводит к неопределенности. Выражать неопределенность позволяет нечеткая логика, в которой, кроме понятий «истина» или «ложь», как в классической логике, используются другие значения истинности с интервалом от 0 до 1.

Мышление человека основано на нечеткой логике и вместо чисел используется классы объектов в виде нечетких множеств с нечеткой истинностью, не-

четкими связями и нечеткими правилами вывода. Нечеткая логика имеют следующие отличительные черты:

- использование нечетких и лингвистических переменных, нечетких множеств;
- отношения между переменными описываются с помощью нечетких высказываний и нечетких алгоритмов;
- сложные отношения между переменными описываются нечеткими алгоритмами.

Аппарат нечетких множеств формально описывает лингвистические понятия, используемые человеком в процессе принятия решений, а также имитацию рассуждения на основе категорий и правил, на которые он опирается. Большинство понятий и правила нечеткой логики являются обобщением или развитием логики предикатов [2].

Понятие нечетких множеств (fuzzy sets - “пушистое” множество) введено в 1965 г. Л. Заде как обобщение понятия классических множеств. В нечетком множестве каждый его элемент может принадлежать множеству частично, тогда как в классических множествах элемент или целиком принадлежит множеству, да или нет. Степень принадлежности элемента а нечеткому множеству А характеризуется коэффициентом принадлежности, обозначаемому $\mu_A(x)$, где $\mu_A(x)$ — действительное число , принимающее значение в диапазоне (0,1), при этом 1 означает 100%-ю (безусловную) принадлежность а к множеству $\mathbf{A}$, а 0 — безусловное отсутствие а в $\mathbf{A}$.

Отображение множества элементов x во множество значений $\mu_{\mathbf{A}}(x)$ образует функцию принадлежности $\mu_{\mathbf{A}}(x)$, которая определена явно в виде, например, алгебраического выражения или таблично (дискретно) в виде массива пар

$$\{x_1/\mu_{\mathbf{A}}(x_1), x_1/\mu_{\mathbf{A}}(x_2), \ldots, x_N/\mu_{\mathbf{A}}(x_N)\}.$$

Чаще всего используются следующие функции принадлежности:

- функция Гаусса для переменной x с центром с и параметром ширины σ имеет следующий вид

$$\mu_A(x) = \exp\left(-\left(\frac{x-c}{\sigma}\right)^2\right)$$

На рис.30 даны графики этой функции для c=1 и σ=1; 0,5; 0,25; 0,05.

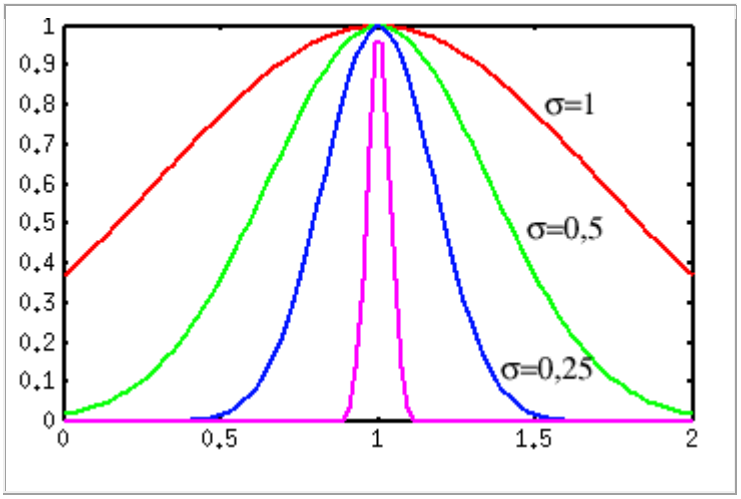


Рис. 5.1 - графики функции Гаусса

Подбором параметра формы функции Гаусса можно придать треугольную и трапецеидальную формы.

- симметричная треугольная функция принадлежности описывается, как

$$\mu_A(x) = 1 - \frac{|x-c|}{d} \text{ для } c - d < x < c + d,$$

$$\mu_A(x) = 0 \text{ для остальных } x.$$

График треугольной функции принадлежности представлен на рис. 31.

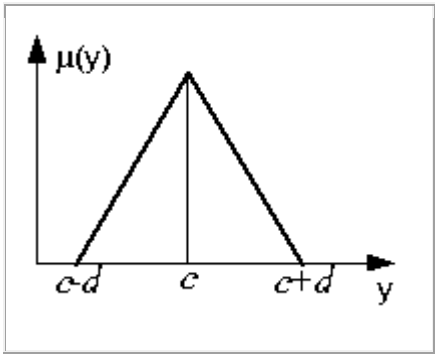


Рис. 5.2- график треугольной функции принадлежности

В теории нечетких множеств помимо числовых переменных существуют лингвистические переменные, представляющие нечеткие знания. Например, лингвистическая переменная "температура тела человека" может принимать значения: "пониженная", "нормальная", "повышенная", "высокая".

Нечеткое множество "нормальная температура тела" может быть дискретно задано следующим образом:

{36,2/0,15; 36,3/0,33; 36,4/0,6; 36,5/0,85; 36,6/1,0; 36,7/0,85; 36,8/0,6; 36,9/0,33; 37,0/0,15}.

То же множество может быть представлено следующим выражением:

$$\mu_{\text{норм}} = \exp\left(-\left(\frac{t-36,6}{0,3}\right)^2\right).$$

На рис. 5.3 представлены графики функций принадлежности для множеств, связанных с лингвистической переменной "температура тела человека".

Рис. 5.3-функции принадлежности

Носителем нечеткого множества $\mathbf{A}$ $Supp(\mathbf{A})$ являются все элементы, для которых коэффициент принадлежности больше нуля, т.е. $Supp(\mathbf{A}) = \{x \mid \mu_{\mathbf{A}}(x) > 0\}$. Для табличного задания функции принадлежности носителем нечеткого множества "нормальная температура" является следующее множество:

{36,2; 36,3; 36,4; 36,5; 36,6; 36,7; 36,8; 36,9; 37,0}.

Два нечетких множества $\mathbf{A}$ и $\mathbf{B}$ **равны** между собой тогда и только тогда, когда $\mu_{\mathbf{A}}(x) = \mu_{\mathbf{B}}(x)$ для всех элементов этих множеств.

Кардинальное число нечеткого множества $\mathbf{A}$ $M(\mathbf{A})$ представляется суммой коэффициентов принадлежности всех элементов этого множества, т.е. $M(\mathbf{A}) = \sum_i \mu_{\mathbf{A}}(x_i)$.

Нечеткое множество называется **нормальным**, если хотя бы один его элемент имеет коэффициент принадлежности равный 1.

Сечением $\mathbf{A}_\alpha$ нечеткого множества $\mathbf{A}$ называется подмножество элементов $\mathbf{A}$, для которых $\mu_{\mathbf{A}}(x) > \alpha$ (слабое сечение) или $\mu_{\mathbf{A}}(x) \geq \alpha$ (сильное сечение), при этом α принадлежит [0,1].

Операции на нечетких множествах.

Основные операции на нечетких множествах следующие:

- Логическая сумма (объединение) множеств $\mathbf{A} \cup \mathbf{B}$

$$\mu_{\mathbf{A}\cup\mathbf{B}}(x) = \max(\mu_{\mathbf{A}}(x), \mu_{\mathbf{B}}(x)).$$

- Логическое произведение (пересечение) множеств $\mathbf{A} \cap \mathbf{B}$

$$\mu_{\mathbf{A}\cap\mathbf{B}}(x) = \min(\mu_{\mathbf{A}}(x), \mu_{\mathbf{B}}(x)).$$

- Отрицание множества $\bar{\mathbf{A}}$

$$\mu_{\bar{\mathbf{A}}}(x) = 1 - \mu_{\mathbf{A}}(x).$$

- Концентрация множества $CON(\mathbf{A})$

$$\mu_{CON(\mathbf{A})}(x) = (\mu_{\mathbf{A}}(x))^2.$$

Эта операция при действиях с лингвистической переменной обычно отождествляется с модификатором "очень".

- Растяжение множества $DIL(\mathbf{A})$

$$\mu_{DIL(\mathbf{A})}(x) = \sqrt{\mu_{\mathbf{A}}(x)}.$$

Эта операция при действиях с лингвистической переменной обычно отождествляется с модификаторами "примерно", "приблизительно".

- Алгебраическое произведение множеств $\mathbf{A}\cdot\mathbf{B}$

$$\mu_{\mathbf{A}\cdot\mathbf{B}}(x) = \mu_{\mathbf{A}}(x)\cdot\mu_{\mathbf{B}}(x).$$

- Нормализация множества $NORM(\mathbf{A})$

$$\mu_{NORM(\mathbf{A})}(x) = \frac{\mu_{\mathbf{A}}(x)}{\max\{\mu_{\mathbf{A}}(x)\}}.$$

Нечеткое множество $\mathbf{A}$ считается подмножеством нечеткого множества $\mathbf{B}$, если для всех элементов $\mathbf{A}$ выполняется неравенство $\mu_{\mathbf{A}}(x) \leqslant \mu_{\mathbf{B}}(x)$.

Описанные выше операции на нечетких множествах обладают следующими свойствами:

- ассоциативность — $(\mathbf{A} * \mathbf{B}) * \mathbf{C} = \mathbf{A} * (\mathbf{B} * \mathbf{C})$;
- дистрибутивность — $\mathbf{A} * (\mathbf{B} \circ \mathbf{C}) = (\mathbf{A} * \mathbf{B}) \circ (\mathbf{A} * \mathbf{C})$;
- коммутативность — $\mathbf{A} * \mathbf{B} = \mathbf{B} * \mathbf{A}$ (за исключением ограниченной разности).

Здесь $*$ и $\circ$ — символы, обозначающие операции над нечеткими множествами.

Меры нечеткости нечетких множеств. Для определения степени нечеткости множества введено понятие меры нечеткости, сводящейся к измерению уровня различия между нечетким множеством $\mathbf{A}$ и его отрицанием $\bar{\mathbf{A}}$.

Наиболее популярна мера Е.Егера

$$FUZ_p(\mathbf{A}) = 1 - \frac{d_p(\mathbf{A}, \bar{\mathbf{A}})}{n^{\frac{1}{p}}},$$

где n — количество элементов в $\mathbf{A}$, $d_p(\mathbf{A}, \bar{\mathbf{A}})$ — расстояние между между множествами $\mathbf{A}$ и $\bar{\mathbf{A}}$ в метрике p (p равно 1 или 2). Значение p=1 соответствует метрике Хемминга

$$d_1(\mathbf{A}, \bar{\mathbf{A}}) = \sum_{i=1}^{n} |2 \cdot \mu_{\mathbf{A}}(x_i) - 1|,$$

а значение p =2 соответствует эвклидовой метрике

$$d_2(\mathbf{A}, \bar{\mathbf{A}}) = \sqrt{\sum_{i=1}^{n} (2 \cdot \mu_{\mathbf{A}}(x_i) - 1)^2}.$$

Приближенные рассуждения и нечеткий вывод

Существует обобщение на нечеткие понятия двузначной логики, которая оперирует двумя значениями истинности: «истина» и «ложь». В отличие от нее в нечеткой логике истинностному значению высказывания соответствуют произвольные величины из отрезка [0, 1].

Л. Заде ввел нечеткую логику с лингвистическими, а не числовыми значениями истинности, в которой логика истинности высказывания определяется значениями: истинно, ложно, очень истинно, абсолютно истинно, не очень истинно, очень ложно и т. п. Эта логика получила название нечеткой логики, на которой также основываются приближённые рассуждения. Под приближенными рассуждениями понимается процесс получения из нечетких посылок некото-

рых следствий. Приближенное рассуждение может рассматриваться как обобщение правил вывода Modus ponens и Modus tollens логики высказываний. Композиционное правило вывода включает, как частный случай, обобщение правила Modus ponens.

Приближенные рассуждения

В классической теории исчисления высказываний выражение «Если A, То B», где A и B – пропозициональные переменные (пропозициональная переменная – это переменная для предложений, рассматриваемых исходя из их истинности или ложности), записывается как A→B, где импликация (→), представляющая связку.

A→B≡¬A∨B

В том смысле, что A→B (A влечет B) и ¬A∨B (не A или B) имеют идентичные таблицы истинности. Представление знаний представляется неопределенным высказыванием «Если A, То B», коротко A→B, в котором A (антецедент) и B (консеквент) – нечеткие множества. Примеры высказываний:

Если «большой», То «Вылючить»

Если «скользкий», То «Опасный;»

Они являются сокращениями предложений:

Если x-«большой», То y-«выключить»;

Если дорога «скользкая», То езда «опасна».

Предложения такого вида описывают отношения между двумя неопределенными переменными. Это означает, что неопределенное высказывание следует скорее определить как нечеткое отношение, а не как связку.

Декартово произведение двух нечетких множеств определяется как A×B, что означает нечеткое множество упорядоченных пар (u,v), $u \in U, v \in V$, со степенью принадлежности (u,v) к (A×B), задаваемой формулой $\mu_A(u) \wedge \mu_B(v)$. В этом смысле A×B есть нечеткое отношение U и V.

Пример. Пусть

U=1+2

V=1+2+3

A=1/1+0,8/2

B=0,6/1+0,9/2+1/3

То

A×B=0,6/(1,1)+0,9/(1,2)+1/(1,3)+0,6/(2,1)+0,8/(2,2)+0,8/(2,3)

Отношение, определенное можно представить матрицей отношения

$$\begin{array}{c} \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0{,}6 & 0{,}9 & 1 \\ 0{,}6 & 0{,}8 & 0{,}8 \end{bmatrix} \end{array}$$

Композиционное правило вывода. Пусть U и V – два универсальных множества с базовыми переменными u и v соответственно. Пусть R(u), R(u,v) и R(v) обозначают ограничения на u, (u,v) и v соответственно и представляют собой нечеткие отношения в U, U×V и V.

Пусть A и F – нечеткие подмножества множеств U и U×V, то композиционное правило вывода утверждает, что решение уравнений назначения

R(u)=A (унарное нечеткое отношение)

R(u,v)=F (бинарное нечеткое отношение)

Имеет вид (v)=A°F

Где A°F – композиция A и F. В этом смысле будет вывод. R(v)= A°F из того, что R(u)=A и R(u,v)=F.

При этом функция принадлежности определяется как

$$\mu_R(v) = \max\left[\min(\mu_R(u), \mu_R(u,v))\right].$$

Для определения композиционного правила вывода применимы нечеткие отношения [3]. Для представления этого правила предположим, что

U=V=1+2+3+4

A=малый=1/1+0.6/2+0./3

F=примерно равны=1/(1,1)+1/(2,2)+1/(3,3)+1/(4,4)+0,5/((1,2)+(2,1)+(2,3)+(3,2)+(3,4)+(4,3)).

Другими словами, A-унарное нечеткое отношение в U, названное «малый»; F – бинарное нечеткое отношение в U×V, названное «примерно равны».

Уравнения назначения в этом случае имеют вид:

R(u)=малый

R(u,v)=примерно равны,

И следовательно

R(v)=малый°примерно равны= $[1 \quad 0{,}6 \quad 0{,}2 \quad 0] \circ \begin{bmatrix} 1 & 0{,}5 & 0 & 0 \\ 0{,}5 & 1 & 0{,}5 & 0 \\ 0 & 0{,}5 & 1 & 0{,}5 \\ 0 & 0 & 0{,}5 & 1 \end{bmatrix} = [1 \quad 0{,}6 \quad 0{,}5 \quad 0{,}2],$

Аппроксимация выполняется следующим образом:

R(v)=более или менее малый.

Используя композиционное правило вывода, из того, что R(u)=малый и R(u,v)= примерно равны, выводится, что

R(v)=[1 0.6 0.5 0.2] точно

и R(v)= «более или менее малый».

Словами это приближенный вывод можно записать в виде:

u-«малый» (предпосылка)

u и v – «примерно равны» (предпосылка)

v – «более или менее малый» (приближенный вывод)

Каждый факт или предпосылка записывается в виде уравнений назначения в отношениях, содержащих одно или больше число ограничений на базовые переменные. Эти уравнения решаются относительно желаемых ограничений при

помощи композиции нечетких отношений. Получаемые решения и представляют собой вывод из данного набора предпосылок.

Нечеткие знания на основе модели продукции

В простейшем случае модель знаний в виде нечеткой продукция имеет следующий вид:

ЕСЛИ x это $\mathbf{A}$, ТО y это $\mathbf{B}$,

где $\mathbf{A}$ и $\mathbf{B}$ — значения лингвистических переменных, задаваемые функциями принадлежности. Левая часть продукции также называется условием (или предпосылкой), а правая — следствием (или заключением, и продукцию записывают как $\mathbf{A}\rightarrow\mathbf{B}$.

В более общем случае нечеткая продукция принимает такую форму: ЕСЛИ x_1 это $\mathbf{A}_1$ И ... И x_N это $\mathbf{A}_N$, ТО y это $\mathbf{B}$.

Для вычисления значения коэффициента принадлежности сложного коньюнктивного условия продукции используются 2 способа:

- логическое произведение

$$\mu_{И}(x) = \min\{\mu_{\mathbf{A}_i}(x_i)\},\ i = 1,2,\ldots,N;$$

- алгебраическое произведение

$$\mu_{И}(x) = \prod_{i=1}^{N}\mu_{\mathbf{A}_i}(x_i).$$

Приписывание значения коэффициента принадлежности сложному условию продукции называют агрегированием условия.

Для вычисления коэффициента принадлежности продукции в целом также используется два способа:

- логическое произведение

$$\mu_{\mathbf{A}\to\mathbf{B}}(x) = \min\{\mu_{\mathbf{A}}(x), \mu_{\mathbf{B}}(y)\};$$

- алгебраическое произведение

$$\mu_{\mathbf{A}\to\mathbf{B}}(x) = \mu_{\mathbf{A}}(x)\cdot\mu_{\mathbf{B}}(y).$$

Такой расчет значения функции принадлежности называется агрегированием на уровне продукции.

Нечеткая продукционная система Мамдани – Заде. В технических и ряде других приложений в качестве входов $x_1, x_2, \ldots, x_N$ и выхода y часто выступают доступные к измерению числовые величины. В такой ситуации для согласования нечетких продукций, оперирующих лингвистическими переменными, с входами/выходами в виде числовых значений в состав систем продукций вводятся фаззификатор и дефаззификатор. Фаззификатор преобразует множество входных данных в нечеткое множество, определяемое с помощью значений функции принадлежности, а дефаззификатор преобразует нечеткое множество, определяемое с помощью значений функции принадлежности, в конкретное значение. На рис. 5.4 показано функционирование нечеткой продукционной системы Мамдани – Заде.

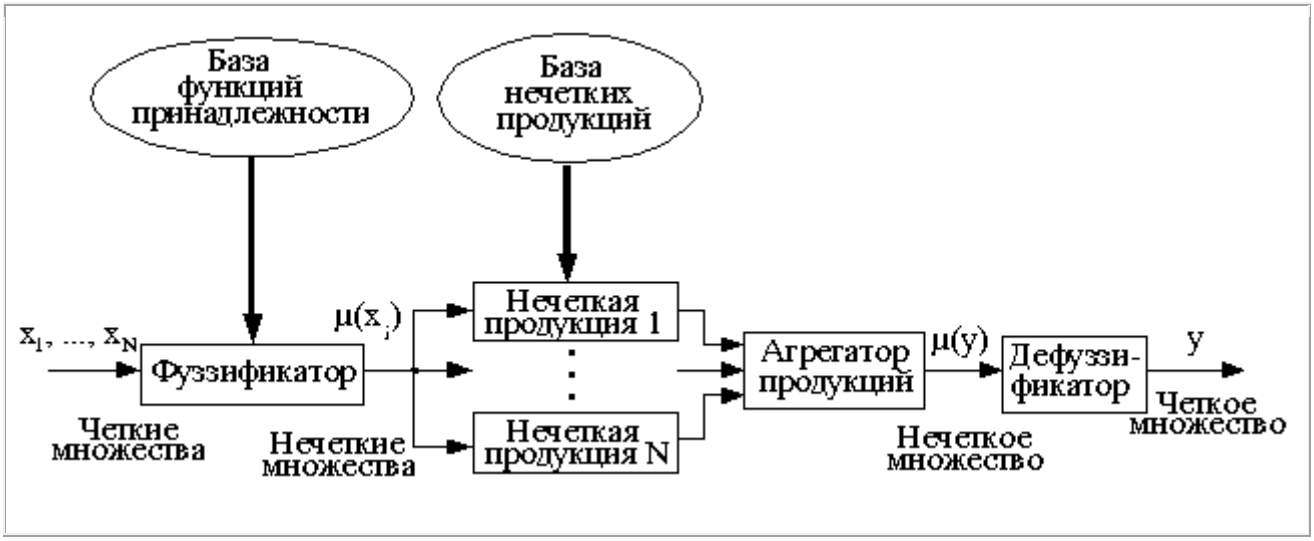


Рис. 5.4.- нечеткая продукционная система Мамдани – Заде

Поступившие на вход значения (например, результаты измерений на реальном физическом объекте) $x_i,\ i = 1, 2, \ldots, N$, преобразуются в значения $\mu(x_i)$ функций принадлежности нечетких множеств, с которыми оперирует ПС. Интепретатор ПС выбирает из базы нечетких продукций все применимые к входным данным продукции и определяет функции принадлежности переменных y из правой части продукций. Поскольку в общем случае применимыми оказываются несколько продукций, то возникает проблема агрегирования функций принадлежности из правых частей отдельных продукций. Это объединение функций принадлежности реализуется, как правило, оператором логического сложения. Нечеткий результат в виде функции принадлежности $\mu(y)$ трансформируется дефаззификатором в конкретное значение выхода y.

Пример 1. Пусть на вход нечеткой ПС поступают две величины x_1 и x_2. Фаззификатор, используя базу функций принадлежности всех известных ему нечетких множеств, определяет значения коэффициентов принадлежности. Отличными от 0 оказались три коэффициента принадлежности $\mu_{\mathbf{A}_1}(x_1) = 0,5;$ $\mu_{\mathbf{A}_2}(x_2) = 0,25;\ \mu_{\mathbf{A}_3}(x_2) = 0,75$ для трех нечетких множеств $\mathbf{A}_1$, $\mathbf{A}_2$ и $\mathbf{A}_3$, как это показано на рис.5.5.

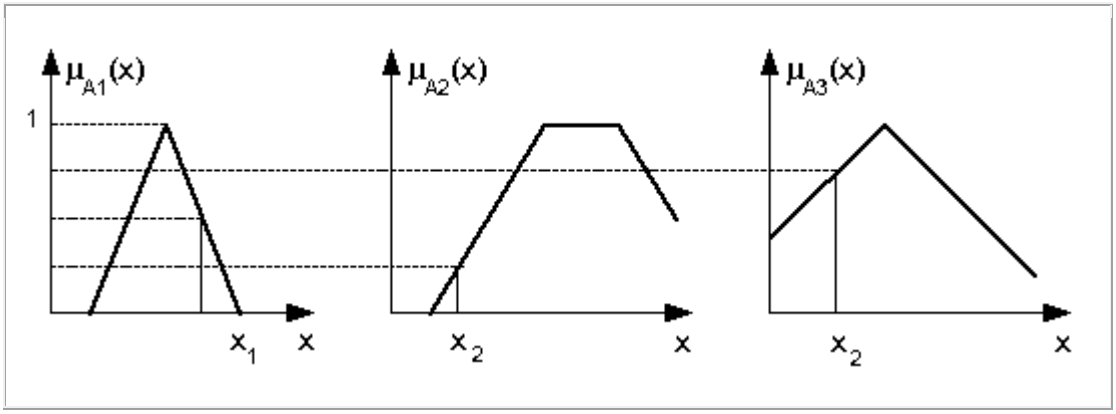


Рис. 5.5 - графики функций принадлежности для трех нечетких множеств

Далее интерпретатор выявил две применимые нечеткие продукции:

- $P1$: ЕСЛИ $x1$ это $\mathbf{A}_1$ И $x2$ это $\mathbf{A}_2$, ТО y это $\mathbf{B}_1$,
- $P2$: ЕСЛИ $x1$ это $\mathbf{A}_1$ И $x2$ это $\mathbf{A}_3$, ТО y это $\mathbf{B}_2$.

Для каждой из продукций в виде логического произведения выполняется агрегирование левой части (условия):

- $\mu_{И1} = \min\{\mu_{\mathbf{A}_1}(x_1), \mu_{\mathbf{A}_2}(x_2)\} = 0,25;$
- $\mu_{И2} = \min\{\mu_{\mathbf{A}_1}(x_1), \mu_{\mathbf{A}_3}(x_2)\} = 0,5.$

Затем выполняется агрегирование (опять же логическим произведением) на уровне продукций, как это показано на рис. 5.6.

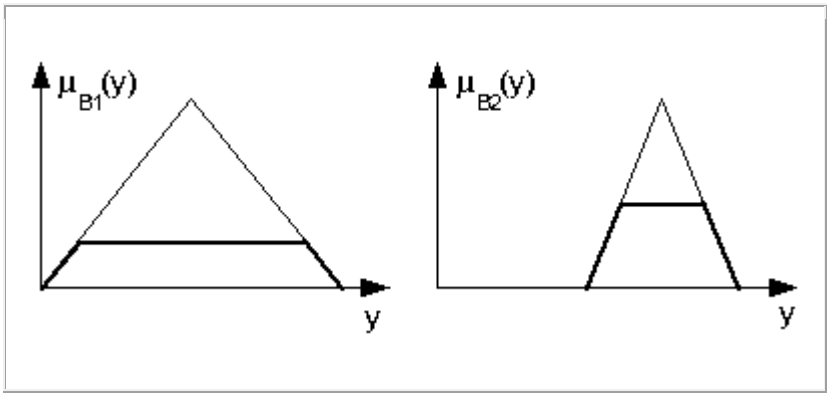


Рис. 5.6 - графики функций принадлежности, являющихся результатом агрегирования на уровне продукций

Результат агрегирования — функции принадлежности, графики которых выделены жирной линией.

Далее агрегатор продукций формирует итоговую функцию принадлежности в виде логической суммы функций принадлежности отдельных продукций, как это показано на рис. 5.7

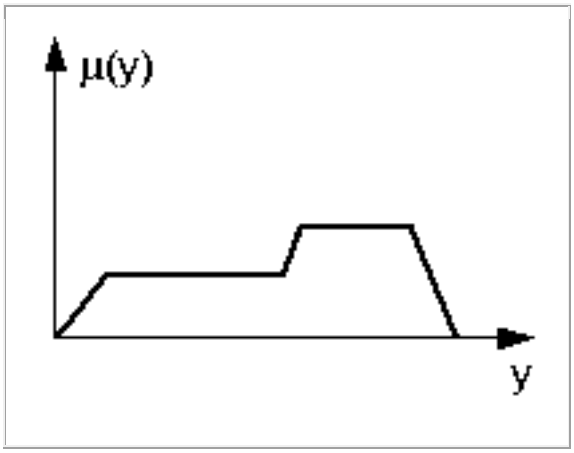


Рис. 5.7 - график итоговой функции принадлежности

Затем дефаззификатор превращает итоговую функцию принадлежности в конкретное значение числовой величины y.

Фаззификатор осуществляет преобразование четкого множества $\mathbf{X}$ в нечеткое множество $\mathbf{A}$, характеризующееся функцией принадлежности $\mu_{\mathbf{A}}(x)$. Наибольшее распространение на практике получили функции принадлежности гауссова типа, а также треугольные и трапецеидальные функции.

Дефаззификатор преобразует нечеткое множество, заданное функцией принадлежности $\mu_{\mathbf{A}}(x)$, в конкретную величину. Для такого преобразования могут быть использованы многие способы, наиболее популярны следующие.

- Дефаззификация относительно центра области

$$y_C = \frac{\int_y y \cdot \mu(y) \cdot dy}{\int_y \mu(y) \cdot dy}$$

или в дискретной форме

$$y_{\mathrm{c}} = \frac{\sum_i y_i \cdot \mu(y_i)}{\sum_i \mu(y_i)}.$$

- Дефаззификация относительно среднего центра

$$y_{\mathrm{a}} = \frac{\sum_i c_i \cdot \mu(c_i)}{\sum_i \mu(c_i)},$$

где c_i — центр i-ой одиночной функции принадлежности, участвующей в итоговой агрегированной функции.

- Дефаззификация относительно среднего максимума

$$y_{\mathrm{m}} = \frac{\sum_i y_i}{m},$$

где m — количество максимумов функции принадлежности, y_i — значение, в котором функция принадлежности имеет максимум. Если $\mu(y)$ одинаковые максимальные значения на интервале между y_l и y_h, то $y_i = \frac{y_l + y_h}{2}$.

- Дефаззификация в виде минимального из максимальных

$$y_{\mathrm{l}} = \min\{y_i\},$$

где y_i — значения, доставляющие функции принадлежности максимум.

- Дефаззификация в виде максимального из максимальных

$$y_{\mathrm{h}} = \max\{y_i\}.$$

На рис. 5.8 показано применение дефаззификации.

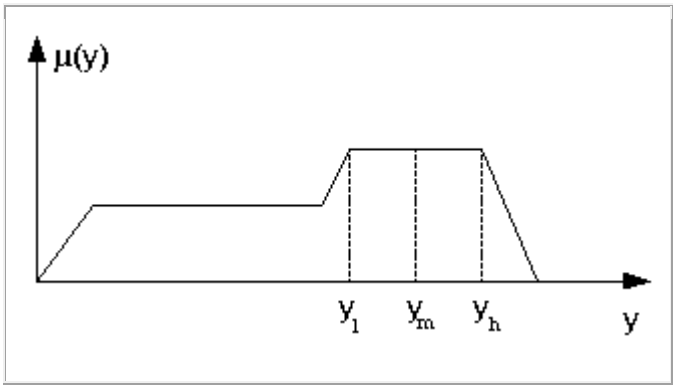


Рис. 5.8 - пример применения дефаззификации

Нечеткая модель Такаги – Сугено

Нечеткие СИИ на основе продукционной модели Такаги – Сугено выглядят следующим образом:

ЕСЛИ x_1 это $\mathbf{A}_{i1}$ И ... И x_N это $\mathbf{A}_{iN}$, ТО $y_i = \varphi(x_1, ..., x_N)$.

Поскольку в правых частях продукций определяются конкретные значения выходов y_i, дефаззификатор на выходе системы не требуется.

Функция в правой части продукции — это, чаще всего, полином первой степени

$$y_i = \varphi(\mathbf{X}) = p_{i0} + \sum_{j=1}^{N} p_{ij} \cdot x_j,$$

где p_{ij} — веса, подбираемые в процессе обучения продукционной системы.

При использовании M продукций итоговый выход y системы определяется как средневзвешенная сумма в виде

$$y = \frac{\sum_{i=1}^{M} w_i \cdot y_i}{\sum_{i=1}^{M} w_i} = \sum_{i=1}^{M} \frac{w_i}{\sum_{i=1}^{M} w_i} \cdot y_i = \sum_{i=1}^{M} w_i' \cdot y_i = \sum_{i=1}^{M} w_i' \cdot (p_{i0} + \sum_{j=1}^{N} p_{ij} \cdot x_j).$$

Для вычисления веса w_i i-ой продукции в ПС используется агрегирование (в виде логического или алгебраического произведения) условий нечетких правил.

Контрольные вопросы

1. В чем смысл функции принадлежности?
2. Чем нечеткий вывод отличается от четкого?
3. Что представляет собой нечеткая импликация?
4. Чем модель Такаги – Сугено отличается от модели Мамдани?
5. Сформулируйте правило композиции Заде?

Глава 6. Методы вывода и поиска решений в системах искусственного интеллекта

Вывод в логических моделях. Логические модели удобны для формального описания мышления человека, так как часто его рассуждения при решении задач носят дедуктивный характер. Для построения подобных рассуждений СИИ должна выводить новые факты исходя из известных ей. Новые логические конструкции и формы создаются применением правил вывода к имеющимся логическим формулам [2].

Пусть A и B - это суждения, причем запись $A=1$ ($B=1$) означает, что логическим значением суждения A (B) считается «истина», тогда как запись $A=0$ ($B=0$) означает, что логическим значением суждения A (B) считается «ложь». Правило вывода modus ponens

Условие	A	A есть истинно
Импликация	$A \to B$	Если A, то B – истинно
Вывод	B	B есть истинно

Пусть суждение A имеет вид «Ян является водителем», а суждение B - «У Яна есть водительское удостоверение». В соответствии с правилом modus ponens, если $A=1$,то и $B=1$, поскольку если истинно, что «Ян является водителем», то также истинно, что «У Яна есть водительское удостоверение». Из истинности предпосылки и импликации следует истинность вывода.

Правило вывода modus tollens

Условие	$\overline{B}$
Импликация	$A \to B$
Вывод	$\overline{A}$

Понимается, что если «У Яна нет водительского удостоверения», т. е. $B=0$ ($\overline{B}=1$), то «Ян не является водителем», т.е. $A=0$ ($\overline{A}=1$). В этом примере из истинности предпосылки и импликации тоже следует истинность вывода.

Формула является выводимой, если получена из конечной совокупности исходных формул путем конечного количества шагов применения правил вывода.

При использовании для вывода теорем дедукции или противоречивости, сначала задачу описывают формулой, а затем доказывают, что эти формулы общезначимы или противоречивы [11].

Для поиска доказательства **методом Эрбрана** необходимо выполнить процедуру стандартизации, т.е. преобразования формул в предложения. Предложение – это множество дизъюнктов, а дизъюнкт – это дизъюнкция литер. В результате стандартизации получается стандартная форма в виде множества дизъюнктов. При использовании метода Эрбрана к стандартной форме формулы применяют процедуры поиска опровержения. Это делается для удобства реали-

зации поиска, т.е. вместо доказательства общезначимости формулы доказывают, что отрицание формулы противоречиво [6].

Пусть S – стандартная форма формулы F, представленная в виде множества дизъюнктов. Тогда F противоречива в том случае, когда противоречива S.

Множество дизъюнктов невыполнимо тогда, когда оно ложно при всех интерпретациях на всех областях. Из-за невозможности рассмотрения интерпретаций на всех областях, строится такая область H, что если не существует удовлетворяющей интерпретации в этой области, то ее вообще не существует, т.е. S невыполнимо тогда, когда оно ложно при всех интерпретациях в этой области. Такая область называется универсумом Эрбрана и обозначается H(S). Недостатком метода Эрбрана является экспоненциальный рост множества дизъюнктов..

Для ограничения порождения множества, предложен метод резолюций, разработанный Робинсоном. Процедура поиска доказательства методом резолюций может применяться к любому множеству дизъюнктов с целью проверки его невыполнимости. Идея метода резолюций заключается в проверке наличия в множестве S пустого (ложного) дизъюнкта (противоречия). Если S содержит противоречие, то оно невыполнимо; если не содержит – то выводятся новые дизъюнкты до тех пор, пока не будет получено противоречие (это всегда имеет место для невыполнимого S).

Недостатком метода резолюций является то, что неограниченное применение правила резолюций вызывает порождение большого числа дизъюнктов, многие из которых лишние, хотя и являются корректными. Кроме того, для выполнимого множества дизъюнктов процедура, основанная на методе резолюций, может продолжаться бесконечно.

Методы резолюций неприемлемы для сложных проблем, так как пространство поиска возрастает экспоненциально. Из-за указанных недостатков дедуктивные методы не находят широкого применения.

Вывод в семантических сетях. Семантическая сеть обладает свойством целостности, что не позволяет разделить БЗ и механизм вывода. Поэтому эффективный способ поиска решений и вывода знаний заключается в сопоставлении частей сетевой структуры. Он основан на построении подсети, соответствующей вопросу, и сопоставлении ее с базой данных сети.

В семантических отношениях узлов и дуг такой сети не должно быть противоречий. Вершины семантической сети обычно показывают объект проблемной области, концепт, ситуацию и т.п., а дуги – отношения между ними. Факты, в явном виде не содержащиеся в системе, могут быть, выведены из других знаний. Выводы в семантических сетях отличаются полнотой, и они сравнимы с нестандартными выводами процедурного представления и имеют ясную концептуальную интерпретацию. Последовательное применение подобных правил вывода может привести к образованию так называемых «цепочек вывода», которые в отдельных случаях могут достигать значительной длины..

В иерархической структуре понятий существуют предикаты отношений двух типов: ЯВЛЯЕТСЯ (IS-A) и ЧАСТЬ (PART-OF). Иерархия показывает отношения включения понятия. Способ включения можно назвать понятием верхнего уровня, а способ удаления – понятием нижнего уровня. Экземпляр нижнего уровня содержит в основном все атрибуты, которые имеет экземпляр понятия верхнего уровня (прототип). Это свойство называется наследованием атрибутов между уровнями иерархии IS-A.

Выражение PART-OF показывает отношение «целое – часть». Этот способ показывает отношения между экземплярами класса, причем основная часть показывает внутреннюю структуру предиката.

Семантическими сетями можно также представлять знания, касающиеся атрибутов объекта. Используя отношения IS-A и PART-OF, можно вывести факт, что объект обладает определенной характеристикой или свойством. Другими словами, факт, объявляемый для вершин на верхнем уровне иерархической структуры, на основе предпосылки, говорящей о справедливости его для узлов нижнего уровня, показывает возможность вывода множества фактов с помощью отношения IS-A.

Проблемой для семантических сетей является наследование атрибутов между иерархическими уровнями. Результат вывода, получаемого с помощью семантической сети, не гарантирует достоверность как логический формализм. Это обусловлено тем, что процедура вывода по определению не более как наследование свойств ветви IS-A. Вследствие этого требуются также способы представления данных и вывода, которые обеспечивали бы одновременно управление наследованием.

Методы выводов в пространстве состояний. Решение многих задач в интеллектуальных системах можно определить как проблему поиска, где искомое решение – это цель поиска, а множество возможных путей достижения цели представляет собой пространство поиска (или пространство состояний). Поиск решений в пространстве состоит в определении последовательности операторов, которые преобразуют начальное состояние в целевое.

Пусть исходная задача описывается тройкой (S, F, T), где S – множество начальных состояний; F – множество операторов, отображающих одни состояния в другие; T – множество целевых состояний. Решение задачи состоит в нахождении последовательности операторов $f_1, f_2, \ldots, f_k$ ($f_i \in F$), преобразующих начальные состояния в конечные. Задача поиска в пространстве состояний описывается с помощью понятий теории графов. Задается начальная вершина (исходное состояние) и множество целевых вершин T. Последовательно от корня к

вершинам дерева применяются операторы, относящиеся к этому уровню, для построения вершин-преемников следующего уровня (раскрытие вершин). Дуги идентифицируются как операторы преобразования. Получаемые вершины - преемники проверяются: не являются ли они целевыми. Далее переходят к следующему уровню. Терминальные вершины – это те, к которым нельзя применить никаких операторов. Раскрытие продолжается до тех пор, пока не получена целевая или терминальная вершина. Поиск на графе состояний – это процесс построения графа *G*, содержащего целевую вершину. Этот граф называется графом поиска. Существуют нескольких вариантов реализации метода перебора, отличающихся заданным порядком, в котором будут перебираться вершины [6].

Поиск в глубину. При поиске в глубину прежде всего раскрывается та вершина, которая имеет наибольшую глубину. Из вершин, расположенных на одинаковой глубине, выбор вершины для раскрытия определяется произвольно. Для сдерживания возможности следования по бесконечному пути вводится ограничение на глубину. Вершины, находящиеся на граничной глубине, не раскрываются.

Поиск в ширину. Вершины раскрываются в последовательности их порождения. Поиск идет по ширине дерева, т. к. раскрытие вершины происходит вдоль одного уровня. Целевая вершина выбирается сразу же после порождения. При поиске в ширину возможно нахождение наиболее короткого пути к целевой вершине, если такой путь есть.

Эвристические методы поиска. Эти методы поиска возможно использовать тогда, когда имеются эмпирические правилами, сокращающие объем вариантов решений. Эвристическая информация основывается на опыте, здравом смысле, допущениях разработчика.

При использовании эвристических методов поиска открытые вершины стремятся упорядочить таким образом, чтобы процесс поиска распространился

в перспективных направлениях. Для определения направления поиска используется некоторая мера, характеризующая перспективность вершины или пути, где эта вершина находится. Эту меру называют оценочной функцией f(n*)*. Эта функция является оценкой стоимости кратчайшего пути из начальной вершины в целевую при условии, что он проходит через вершину *n*. При раскрытии вершины или определении пути выбирается вершина с минимальным значением оценочной функции. Оценочная функция должна адекватно характеризовать пространство поиска, т.е. необходим достаточно большой объем знаний о проблемной области и тщательный анализ пространства состояний.

Вывод во фреймовой системе. Во фреймовой системе знания представляются в виде отдельных кластеров (фреймов), содержащих сведения об общих характеристиках данного класса объектов или ситуаций. Система выполняет поиск структуры, наилучшим способом описывающую рассматриваемую ситуацию. При этом слоты заполняются некоторой информацией и заполненный фрейм проверяется на адекватность данной ситуации. В случае несовпадения ищется новый фрейм и процесс продолжается.

Организация вывода, заключающаяся в последовательном поиске активации в сети фреймов до нахождения наиболее соответствующего и построения на его основе экземпляра фрейма.

Для сохранения преимуществ декларативного и процедурного представления, знания, касающиеся функций непосредственного представления их с помощью фреймов, должны храниться в декларативной форме, а знания об использовании фреймов — в процедурной.

В частности, процедуры могут хранить знания, позволяющие давать ответ на следующие вопросы:

1. Подобно «демонам» фреймы могут активировать сами себя в случае, если распознана соответствующая ситуация.

2. Фрейм может автоматически передать управление другому фрейму или деактивировать себя.

3. Заполнение слотов происходит в момент вызова или позднее, по мере необходимости. Реализация этих функций возложена на присоединенные процедуры. Процедуры могут также реализовывать эвристики, направленные на поиск необходимой для заполнения слотов информации.

Вероятностный вывод. Неопределённость знаний приводит к тому, что решающие правила даже в простых случаях могут не дать корректных результатов или имея неполные знания, невозможно уверенно предсказать результат вывода. Эксперты пользуются неточными методами по двум главным причинам:

- точных методов не существует;

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

Четыре источника неопределенных знаний обычно существуют в СИИ: неизвестные данные, неточный язык, неявное смысловое содержание и трудности, вызванные несовпадением взглядов различных экспертов. Для управления неопределенностью, в СИИ используют Байесовское вероятностное рассуждение и теорию уверенности.

Пусть *A* есть событие в реальной действительности, а *B* – другое событие, причем события *A* и *B* не являются взаимно исключающими, но происходят условно при проявлении другого. Условная вероятность математически обозначается P(A|B), в которой вертикальная черта изображает «имело место», а полное выражение вероятности интерпретируется как «Условная вероятность того, что произойдет событие *A* при условии, что имело место событие *B*».

$$p(A|B)=\frac{количество\quad взможных\quad совместных\quad проявлений\, A\quad и\quad B}{количество\quad возможных\quad проявлений\quad B}$$

Количество возможных совместный проявлений *A* и *B*, или вероятность того, что *A* и *B* произойдут совместно, называется совместной вероятностью *A* и *B*. Математически это представляется $P(A\cap B)$.

Количество способов возможного проявления *B* является вероятностью *B*, $P(B)$, и тогда

$$p(A|B)=\frac{p(A\cap B)}{p(B)}$$

Так же, условная вероятность того, что произойдет событие *B* при условии, что имело место событие *A* определяется

$$p(B|A)=\frac{p(B\cap A)}{p(A)}$$

Отсюда,

$p(B\cap A)=p(B|A)\times p(A)$

Совместная вероятность является коммутативной, таким образом

$p(A\cap B)=p(B\cap A)$

Следовательно

$p(A\cap B)=p(B|A)\times P(A)$

Подстановка предыдущих уравнений приводит к следующему уравнению:

$$p(A|B)=\frac{p(B|A)\times p(A)}{p(B)}$$, где

$p(A|B)$ – условная вероятность того, что произойдет событие *A* при условии, что имело место событие *B*;

$p(B|A)$ – условная вероятность того, что произойдет событие *B* при условии, что имело место событие *A*;

$p(A)$ - вероятность того, что произойдет событие *A;*

$p(B)$ - вероятность того, что произойдет событие *B*.

Это уравнение известно как правило (или формула) Байеса, которое названо именем Томаса Байеса, британского математика XVIII века, предложившего это правило [6].

Понятие условной вероятности предлагается, когда считается, что событие *A* было зависимым от события *B*. Этот принцип может быть расширен на случай, когда событие *A* зависит от некоторого числа несовместимых событий B_1, B_2, …, B_n.

Следующее множество уравнений может быть выведено:

$p(A \cap B_1) = p(A|B_1) \times p(B_1)$

$p(A \cap B_2) = p(A|B_2) \times p(B_2)$

. . .

$p(A \cap B_n) = p(A|B_n) \times p(B_n)$

Или после объединения.

$$\sum_{i=1}^{n} p(A \cap B_i) = \sum_{i=1}^{n} p(A \mid B_i) \times p(B_i)$$

Если просуммировать по всему полному перечню событий B_i, получаем

$$\sum_{i=1}^{n} p(A \cap B_i) = p(A)$$

Это приводит к следующему уравнению условной вероятности, т.е. *P(A)* выражается с помощью формулы полной вероятности:

$$p(A) = \sum_{i=1}^{n} p(A \mid B_i) \times p(B_i)$$

Если проявление события *A* зависит только от двух взаимно исключающих событий, например *B* и *НЕ B*, тогда () будет выглядеть так:

$p(A) = p(A|B) \times p(B) + p(A \setminus \neg B) \times p(\neg B)$

Где ¬ - логическое НЕ.

Также

$p(B)=p(B|A)\times p(A)+p(B\backslash\neg A)\times p(\neg A)$

Подставим $p(B)$ в формулу Байеса, что приводит к

$$p(A|B)=\frac{p(B|A)\times p(A)}{p(B|A)\times p(A)+p(B|\neg A)\times p(\neg A)}$$

Уравнение обеспечивает основу использования теории вероятности для управления неопределенностью в интеллектуальных системах.

Байесовский вывод. Представим правила в БЗ в следующей форме:

Если H истинно Тогда C истинно {с вероятностью p}. Это правило подразумевает, что если событие H происходит, тогда вероятность, что событие C произойдет, равно p. Вычисления вероятности того, что событие H также произошло.

Заменим H и C вместо A и B. В интеллектуальных системах H обычно обозначает гипотезы, а C – доказательства (свидетельства) в поддержку этих гипотез.

Таким образом, уравнение, выраженное в терминах гипотез и свидетельств, выглядит так

$$p(H|C)=\frac{p(C|H)\times p(H)}{p(C|H)\times p(\neg H)+p(C|\neg H)\times p(\neg H)}$$

$p(H)$ – априорная (безусловная) вероятность того, что гипотеза H является истинной;

$p(C|H)$ – вероятность того, что гипотеза H является истинной, будет результатом доказательства C;

$p(\neg H)$ - априорная (безусловная) вероятность того, что гипотеза H является ложной;

$p(C|\neg H)$ – вероятность нахождения доказательства C даже когда гипотеза H ложна.

В СИИ для решения задачи, которые ставят эксперты, требуются значения вероятностей. Эксперт определяет априорные вероятности для возможной гипотезы $p(H)$ и $p(\neg H)$, а также условные вероятности для наблюдаемого доказательства C, если гипотеза H истинна, $p(C|H)$, и если гипотеза H ложна, $p(C|\neg H)$. Пользователи обеспечивают информацию о наблюдаемом доказательстве, и интеллектуальная система вычисляет $p(C|H)$ для гипотезы H в свете представленного пользователем доказательства C. Может быть ситуация, когда эксперт, основываясь на простом доказательстве C, не может выбрать простую гипотезу, но обеспечивает многочисленные гипотезы $H_1, H_2, \ldots, H_m$ Это требует получения условных вероятностей всех возможных комбинаций свидетельств для всех гипотез, что требование делает задачу эксперта невыполнимой. Поэтому в СИИ, допускается условная независимость между различными доказательствами.

Следовательно, получаем

$$p(H_i \mid C_1, C_2, ..., C_n) = \frac{p(C_1 \mid H_i) \times p(C_2 \mid H_i) \times ... \times p(C_n \mid H_i) \times p(H_i)}{p(C_1 \mid H_k) \times p(C_2 \mid H_k) \times ... \times p(C_n \mid H_k) \times p(H_k)}$$

Для вычисления все апостериорных вероятностей СИИ ранжирует потенциально истинную гипотезу.

Пример. Эксперт, имея три условно независимых доказательства C_1, C_2 и C_3, создает три взаимно исключающие гипотезы H_1, H_2 и H_3 и обеспечивает априорные вероятности для этих гипотез – $p(H_1)$, $p(H_2)$ и $p(H_3)$ соответственно. Эксперт также определяет условные вероятности каждого отмеченного доказательства для всех возможных гипотез.

В Таблице 6.1 показаны априорные и условные вероятности, обеспеченные экспертом.

Сначала наблюдаем свидетельство C_3, и вычисляются апостериорные вероятности для всех гипотез:

$$p(H_i \mid C_3) = \frac{p(C_3 \mid H_i) \times p(H_i)}{\sum_{k=1}^{3} p(E_3 \mid H_i) \times p(H_k)}, i = 1,2,3$$

Табл 6.1 - априорные и условные вероятности.

Вероятности	Гипотезы		
	i=1	*i=2*	*i=3*
$P(H_i)$	0.4	0.35	0.25
$P(C_1\|H_i)$	0.3	0.8	0.5
$P(C_2\|H_i)$	0.9	0.0	0.7
$P(C_3\|H_i)$	0.6	0.7	0.9

Таким образом

$$p(H_1 \mid C_3) = \frac{0,6 \times 0,4}{0,6 \times 0,4 + 0,7 \times 0,35 + 0,9 \times 0,25} = 0,34$$

$$p(H_2 \mid C_3) = \frac{0,7 \times 0,35}{0,6 \times 0,4 + 0,7 \times 0,35 + 0,9 \times 0,25} = 0,34$$

$$p(H_3 \mid C_3) = \frac{0,9 \times 0,25}{0,6 \times 0,4 + 0,7 \times 0,35 + 0,9 \times 0,25} = 0,32$$

После того, как наблюдается доказательства C_3, доверие гипотезе H_2 и становится равным доверию гипотезе H_1. Доверие гипотезе H_3 также возрастает и даже приблизительно достигает доверию гипотезам H_1 и H_2.

Наблюдаем свидетельство C_1, и рассчитываются апостериорные вероятности

$$p(H_i \mid C_1 C_3) = \frac{p(C_1 \mid H_i) \times p(C_3 \mid H_i) \times p(H_i)}{\sum_{k=1}^{3} p(C_1 \mid H_i) \times p(C_3 \mid H_i) \times p(H_k)}, i = 1,2,3$$

Таким образом

$$P(H_i \mid C_1C_3) = \frac{0{,}3\times 0{,}6\times 0{,}4}{0{,}3\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}9\times 0{,}25} = 0{,}19$$

$$P(H_2 \mid C_1C_3) = \frac{0{,}8\times 0{,}7\times 0{,}35}{0{,}3\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}9\times 0{,}25} = 0{,}52$$

$$P(H_3 \mid C_1C_3) = \frac{0{,}5\times 0{,}9\times 0{,}25}{0{,}3\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}9\times 0{,}25} = 0{,}29$$

Гипотеза H_2 теперь рассматривается как наиболее вероятная, т.к. доверие гипотезе резко уменьшается.

После наблюдения доказательства C_2 также вычисляется последние апостериорные вероятности для всех гипотез:

$$p(H_i \mid C_1C_2C_3) = \frac{p(C_1 \mid H_i)\times p(C_2 \mid H_i)\times p(C_3 \mid H_i)\times p(H_i)}{\sum_{k=1}^{3} p(E_1 \mid H_i)\times p(C_2 \mid H_i)\times p(C_3 \mid H_i)\times p(H_k)}, i = 1,2,3$$

Таким образом

$$p(H_i \mid C_1C_2C_3) = \frac{0{,}3\times 0{,}9\times 0{,}6\times 0{,}4}{0{,}9\times 0{,}3\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}0\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}7\times 0{,}9\times 0{,}25} = 0{,}45$$

$$p(H_2 \mid C_1C_2C_3) = \frac{0{,}8\times 0{,}0\times 0{,}7\times 0{,}35}{0{,}3\times 0{,}9\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}0\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}7\times 0{,}9\times 0{,}25} = 0$$

$$p(H_3 \mid C_1C_2C_3) = \frac{0{,}5\times 0{,}7\times 0{,}9\times 0{,}25}{0{,}3\times 0{,}9\times 0{,}6\times 0{,}4 + 0{,}8\times 0{,}0\times 0{,}7\times 0{,}35 + 0{,}5\times 0{,}7\times 0{,}9\times 0{,}25} = 0{,}55$$

Хотя первоначальное ранжирование, предложенное экспертом было H_1, H_2 *и* H_3, только гипотезы H_1 и H_3 остались для рассмотрения после всех свидетельств (C_1, C_2 и C_3), которые наблюдались. Гипотеза H_2 теперь может быть полностью исключена.

Гипотеза H_3 теперь рассматривается как более предпочтительная, чем гипотеза H_1. При применении методов на базе байесовской логики возникают трудности с получением большого числа данных, необходимых для определения условных вероятностей.

Логический вывод Демпстера – Шеффера. Для представления субъектной ненадежности, которую не способна выразить байесовская вероятность,

была введена теория вероятностей Демпстера – Шеффера, отличающаяся тем, что она не фиксирует значения вероятности, а может представлять и незнание.

Демпстер предложил такие понятие, как нижняя и верхняя вероятности. Шеффер, совершенствуя теорию Демпстера, переименовал их соответственно в функцию доверия и меру правдоподобия с целью придания этим понятиям субъективного смысла.

При использовании байесовского подхода отобранные гипотезы об искомых элементах множеств, имеют равные априорные вероятности и, следовательно, равномерно распределены веса доказательств между этими гипотезами. Для байесовской вероятности требуется соотношение $p(A)+p(\neg A)=1$.

В теории Демпстера – Шеффера предполагается:

- гипотезы (компоненты пространства гипотез Θ) являются взаимно исключающими,

- набор гипотез представляется исчерпывающим.

- имеется средство получения доказательств в пользу подмножеств гипотез $h_1, \ldots, h_m$, принадлежащих Θ,

- имеется средство получения доказательств в пользу подмножеств гипотез $A_1, \ldots, A_k$, которые могут перекрываться.

Эти доказательства рассматриваются, как элементы множества Ψ и формируется отображение $\Gamma: \Psi \rightarrow 2^{\Theta}$, связывающее каждый элемент в Ψ с подмножеством пространства Θ.

1. В теории Демпстера-Шеффера сужается пространство рассматриваемых гипотез до определённого подмножества. Правила, порождающие такое сужение пространства гипотез, ничего не говорят об относительном правдоподобии отобранных гипотез. Функция присвоения базовых вероятностей в теории Демпстера-Шеффера не делает различия между априорными и апостериорными вероятностями, а потому и не приводит к такому распределению вероятностей.

2. В теории Демпстера-Шеффера изменение степени доверия к подмножеству гипотез A не принуждает к изменению степени доверия к остальным гипотезам.

3. При использовании модели Демпстера-Шеффера появление нового свидетельства оказывает большее влияние, чем использование модели, основанной на коэффициентах уверенности.

Контрольные вопросы

1. Почему выводы на логических моделях имеют ограниченное применение?
2. Что такое дизъюнкт?
3. В чем суть вывода modus ponens?
4. Какое новшество появилось в модели Демпстера – Шеффера?
5. Какие источники неопределенностей существуют в СИИ?
6. В чем заключаются ограничения метода резолюций?

Глава 7. Языки программирования СИИ

Развитие искусственного интеллекта вызвало необходимость создания языков программирования для представления знаний и оболочек, ориентированных на разработку и эксплуатацию систем искусственного интеллекта.

Язык программирования Лисп создан в 1960-х гг. американским ученым Дж. Маккарти и его учениками. Наиболее известными диалектами этого языка являются InterLisp, QLisp, CommonLisp. На языке Лисп написаны многие СИИ (Mycin, Internist, Kee), системы естественно-языкового общения (Margie, Shrdlu, Дилос), интеллектуальные ОС (Flex).

Достоинства Лиспа заключаются в том, что посредством простых конструкций можно представлять сложные системы обработки символьной информации в виде списков. Но при этом Лисп - системы имеют низкую вычислительную эффективность.

В языке Лисп "данные" и "программы" внешне не отличаются друг от друга, что позволяет писать на Лиспе программы, манипулирующие не только "данными", но и "программами", и это свойство позволяет Лиспу стать средством программирования систем ИИ.

Язык программирования FRL (Frame Representation Language). Относится к классу фрейм - ориентированных языков. Фрейм в FRL - это совокупность поименованных, ассоциативных списков, содержащая до пяти уровней подструктур. Подструктурами фреймов могут быть слоты, аспекты, данные, комментарии и сообщения.

Важным свойством FRL является наличие в нем встроенного механизма "наследования свойств". Иначе говоря, все понятия предметной области в БЗ организовываются в виде иерархической классификационной системы, где каждое общее (родовое) понятие связывается с более конкретным (видом). На сегодняшний день большинство FRL - систем написаны на Лиспе.

Язык программирования OPS, реализующий продукционную модель знаний, предназначен для разработки СИИ, и, в частности экспертных систем. Архитектура языка OPS типичная для продукционных систем включает базу правил, рабочую память и механизм вывода. Отличительные черты семейства языков OPS: программное управление стратегией вывода решений, развитая структура данных и принципиальная эффективность реализации.

Язык программирования Рефал (рекурсивных функций алгоритмический язык). Это машинно-независимый алгоритмический язык, ориентированный на так называемые "символьные преобразования": перевод с одного языка на другой, алгебраические выкладки. Особенностью Рефала является его использование в качестве метаязыка для построения системных макрокоманд и специализированных языков. Конкретная область применения Рефала заключается в разработке СИИ, создание специализированных языков общения с ЭВМ, автоматическую генерацию программ, перенос программ на языки высокого уровня и их адаптацию при переходе от одного типа ЭВМ к другому.

Язык программирования Пролог. Название “Пролог” есть сокращение для "ПРОграммирование в терминах ЛОГики". Пролог может быть использован в различных приложениях, относящихся к искусственному интеллекту:

– общение с ЭВМ на естественном языке;

– символьные вычисления;

– написание компиляторов;

– базы данных;

– экспертные системы и т.д.

Язык программирования Пролог относится к языкам, используемых в символьной обработке и в исследованиях по искусственному интеллекту, задачи которого рекурсивны по своей природе. Эти задачи, имеют свои особенности реализации:

1. Используют символьное представление информации и символьные вычисления, включая не только операции обработки цепочки символов, но и достаточно сложные графовые структуры.

2. Требование поиска, для которого Пролог, используя набор условий, производит поиск объекта с помощью встроенного недетерминированного механизма

бэктрекинга. Одно из важных применений недетерминированного поиска представляет логический вывод.

Язык программирования основан на ограниченном наборе механизмов, включающих в себя сопоставление образцов, древовидное представление структур данных и автоматический возврат. Такой небольшой набор образует мощный и гибкий программный аппарат. Пролог идеально подходит для создания баз данных и баз знаний, а также экспертных систем, основанных на базах знаний.

Диалект Пролога язык PDCProlog разработан фирмой PrologDevelopmentCenter (Копенгаген, Дания) и предназначен для программирования задач из области искусственного интеллекта. Достоинства реализации:

- является компиляторно-ориентированным языком программирования высокого уровня,
- имеет средства создавать автономно компилируемые файлы.
- Включает в состав интегрированную среду для разработки, отладки и выполнения программ.

Программирование на Прологе – программирование в терминах логики, и программы на Прологе записываются в декларативном стиле. Задача описывается в терминах фактов и правил, а поиск решения Пролог берет на себя посредством встроенного механизма логического вывода. Идентификаторы в Прологе (имена предикатов, переменных) несут большую смысловую нагрузку, чем в традиционных языках программирования [5].

Программа на Прологе включает в себя постанову задачи в виде множества формул логики предикатов первого порядка и описания цели, которую нужно достичь, исходя из множества фактов и правил, содержащихся в этой постановке. Поэтому формализм исчисления предикатов первого порядка оказался удобным для описания постановки задачи на языке, близком к естественному.

Подмножество формальной логики (предикаты), представлено хорновскими дизъюнктами, имеющими вид:

$\neg P1 \vee \neg P2 \vee \ldots \vee \neg Pm \vee Pn$,

где P1, P2, … Pm, Pn – предикаты (термы, логические формулы, атомы). В последнем выражении только формула Pn без отрицания, в чем главная особенность хорновского дизъюнкта. Это выражение может быть переписано в виде:

$P1 \,\&\, P2 \,\&\, \ldots \,\&\, Pm \Rightarrow Pn$, и оно имеет следующее объяснение:

из P1 и P2 и … и Pm следует Pn.

Последнее выражение также называется дизъюнктом, хотя от знаков отрицания и дизъюнкции перешли к импликации и конъюнкции. Особенность хорновского дизъюнкта: справа от знака следования стоит только одно следствие из некоторого множества дизъюнктов, записанных слева. В синтаксисе Пролога последнее выражение записывается:

Pn :- P1, P2, … , Pm.

Следствие оказалось записанным слева, а поскольку Пролог базируется на хорновских дизъюнктах, в левой части импликации (заключения) содержится единственный литерал. Последнюю запись утверждения Пролога можно трактовать двумя способами:

1) декларативный смысл: Pn истинно, если одновременно истинны P1, P2, … , Pm.
2) процедурный смысл: для того, чтобы выполнить Pn, надо последовательно выполнить P1, P2, … , Pm.

Логика хорновских дизъюнктов эквивалентна теории предикатов первого порядка, применяемой для доказательства на основе опровержения. Механизм логического вывода использует метод резолюций. Процесс поиска доказательства на основе метода резолюции и составляет суть выполнения программы на Прологе.

Структура программы на Прологе содержит набор процедур, каждая из которых представляет собой определенный предикат. Предикат имеет общую форму:

A :– B1, B2, ... , Bn. ,

которая интерпретируется : "A является истинным, если B1, B2, ..., Bn являются истинными". Таким образом, значок “ :- “ соответствует обозначению «если». Предикат, содержащий условия истинности, является правилом.

При n=0 отсутствуют условия истинности, и такое предложение выражает факт, и это записывается "A." (точка в конце записи предиката обязательна). Поскольку факт не содержит условий истинности, в Прологе факт всегда является истинным.

В данной форме записи часть выражения, стоящая слева от знака “ :- “, называется головой дизъюнкта (в нашем примере "A"), а выражение, стоящее после этого значка, называется телом дизъюнкта, т.е. B1, B2, ... , Bn – это тело, или хвост дизъюнкта.

Пролог-программа должна иметь цель, поскольку вычисление такой программы всегда начинается от цели. Достичь цели всегда означает, что она логически следует из правил программы. Если получен запрос (т.е. цель, которую нужно удовлетворить), Пролог пытается определить его истинность двумя способами.

- цель успешно удовлетворяется (т.е. считается истинной), если она сопоставляется с существующим фактом (так как факты всегда являются истинными).
- цель считается истинной, если она сопоставляется с головой “A” правила " A, если B1, ... , Bn " и если подцели B1, ... , Bn могут быть завершены успешно. В случае успешного сопоставления Пролог выдает ответ "yes", т.е. цель согласована. Термин «*сопоставление*» обозначает совпадение цели с головой факта или правила и относится к одному из двух базовых механизмов логического вывода.

Если попытка сопоставить подцель с фактами в базе данных завершается неудачей и остаются альтернативные (еще не применявшиеся правила или факты), Пролог будет осуществлять возврат и использовать эти альтернативы. Если все альтернативы закончились неудачно, то считается, что начальная цель неудовлетворительна. В этом случае выдается ответ "no".

В Пролог-программе нет никаких выражений, кроме предикатов, и цель тоже представляет собой некоторый предикат (или последовательность предикатов). В качестве аргументов цель может содержать константы или переменные. Переменная представляет собой неконкретизированную величину, которую Пролог должен конкретизировать, т.е. найти объекты-константы, которые при их подстановке вместо переменных обеспечивают достижение цели. Неконкретизированная переменная считается свободной, а если выполнена подстановка константой, то связанной. После того, как цель, содержащая переменные, будет согласована, Пролог выдаст значения, которыми будут заменены переменные, входящие в цель.

Пример: родитель(«царь Петр I», «царевич Алексей»). Зададим вопрос программе, чьим родителем является царь Петр I:

? родитель(«царь Петр I», X).

Чтобы ответить на вопрос, нужно вместо X подставит константу «царевич Алексей». В этом случае переменная X конкретизируется константой. В этом примере имела место унификация, т.е. подстановка значений вместо переменных при выводе логической формулы. В результате унификации переменная X получила конкретное значение.

В Прологе отсутствуют глобальные переменные, одни и те же имена переменных можно употреблять во многих правилах программы, при этом между такими переменными будет отсутствовать какая-либо связь.

Имена предикатов выбираются согласно смыслу программы. В синтаксисе Пролога предикат имеет следующий вид:

functor(d1,d2,...) :– cond1, cond2... Здесь functor – имя предиката;

functor(d1,d2,...) – голова дизъюнкта; d1, d2,... – аргументы предиката;

cond1, cond2,... – условия истинности дизъюнкта (в свою очередь, предикаты), где запятые обозначают связку “ И ”;

":–" - обозначение "ЕСЛИ".

Имена предикатов и их размерность («арность») произвольны и зависят только от целей их использования.

Пусть конкретное предложение:

P :- Q, R.

где P, Q, R – некоторые предикаты. Для выяснения истинности P следует сначала выяснить истинность Q, затем истинность R. Таким образом, запятая между целями обозначает конъюнкцию целей. Однако в Прологе возможна и дизъюнкция целей: должна быть истинной, по крайней мере, одна из целей. Дизъюнкция обозначается точкой с запятой:

P :- Q; R.

Смысл такого предложения тот же, что и смысл следующей пары предложений:

P :- Q.

P :- R.

Таким образом, логическая функция «ИЛИ» может быть записана двумя способами.

Если опущены условия, то functor(d1,d2,...) описывает факт.

Пример. Опишем мир увлечений круга лиц.

Ellen likes reading.

John likes football.

Tom likes baseball.

Eric likes reading.

Mark likes tennis.

соответствуют факты Пролога:

likes(ellen, reading).

likes(john, football).

likes(tom, baseball).

likes(eric, reading).

likes(mark, tennis).

Имена людей записали строчными буквами, чтобы отличить их от имен переменных, которые в Прологе начинаются с прописной буквы.Некоторое множество фактов, задающих объекты и отношения между ними, уже образует простую программу на Прологе. Таким образом, вопрос: "Что любит Джон?" сводится к поиску объекта X, который отношение likes (нравится) связывает с Джоном; этот объект и будет служить ответом на запрос:

? likes(john, X).

X = football.

(В языке PDCProlog вместо знака вопроса, обозначающего цель, записывается ключевое слово Goal). При определении отношения между объектами существенен порядок, в котором эти объекты входят в отношение:

likes(john, football) отличается от likes(football, john).

Факты – важная часть программы искусственного интеллекта, без них нельзя достичь цели. Можно сказать, что каждая цель опирается на свои факты. После записи фактов обычно следуют правила. Правило всегда выражает некоторый общий закон.

: «Всякий X, любящий читать – умный». Его запись в виде правила:

clever (X) :- likes (X, reading).

Если значением факта всегда является «истина», то правило содержит условие истинности, записанное в его правой части. Если цель найти всех умных людей среди имеющихся фактов, введем запрос:

? clever (X).

Система найдет два решения:

X=ellen

X= eric

Правило Пролога можно представить как небольшую процедуру с локальными переменными. Предикат может выполнять несколько функций, в зависимости от состояния аргументов (конкретизированы они или нет). Одно и то же отношение может быть использовано несколькими способами: если его аргументы известны, то проверяется выполнимость отношения на этих аргументах. Если некоторые аргументы неизвестны, то вычисляется множество наборов значений этих переменных, удовлетворяющих отношению [5].

Пример. Имеется предикат square («площадь»), связывающий длину и высоту геометрической фигуры с третьим значением (например, площадью прямоугольника):

square(A, B, S),

этот предикат вычисляет площадь S:

? square(5, 4, S),

Можно задавать и другие цели:

? square(X, 4, S),

? square(X, Y, 20).

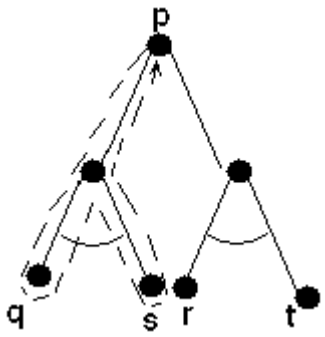
Система Пролог обладает встроенным механизмом логического вывода (перебора), благодаря чему от пользователя требуется только описание своей задачи с помощью аппарата логики предикатов первого порядка, а поиск решения система берет на себя. При этом Пролог – строго последовательный язык. Проце-

дуры выполняются в нем строго по очереди, в том порядке, в котором они записаны. Если вызов процедуры завершен успешно, Пролог продвигается вперед, чтобы попытаться выполнить следующую процедуру. При этом в стеке запоминаются все возможные разветвления. Если вызов неудачен, происходит отход назад к ближайшему из предыдущих вызовов процедуры с целью найти другое решение, при этом отменяются результаты всех подстановок, сделанных после последнего разветвления. Такой же возврат осуществляется и после завершения успешного вычисления – для получения других ответов. При поиске новых решений теряются решения, сгенерированные на предыдущих шагах.

Пример. Программа на Прологе в виде предикатов без аргументов:

p :– q, s.

p :– r, t.



Эту программу можно прочитать следующим образом. При выполнении процедуры p выполняются процедуры q и s. Если они завершаются успешно, то и процедура p считается успешно завершенной. Если это не так, то выполняются процедуры r и t. В этом случае процедура успешно завершается, если r и t завершены успешно. В противном случае процедура p терпит неудачу. Этот процесс возврата к поиску других решений называется бэктрекинг.

Алгоритм работы можно представить в виде И-ИЛИ-дерева. В прологе применяется стратегия поиска в глубину, т.е. обход дерева сверху вниз слева направо. Если при обходе вершины ИЛИ какой-либо потомок разрешается успешно, то обработка остальных дочерних вершин приостанавливается, и считается, что

вершина ИЛИ разрешается успешно. Если какой-либо из потомков И–вершины терпит неудачу, то обработка остальных ее потомков полностью прекращается и данная вершина терпит неудачу.

Приостановка обработки ИЛИ-вершины означает, что впоследствии возможно повторное обращение к этой вершине для поиска других решений (бэктрекинг).

Основными механизмами поиска решения задачи в Прологе является сопоставление и унификация, что позволяет решать широкий класс задач наиболее простым способом: путем перебора всех возможных состояний. Сначала находится предикат, который сопоставляется с целью, а затем выполняется унификация аргументов предиката. Для поиска всех решений используется предикат fail

Имена в языке Prolog используются для обозначения переменных, символьных констант, объявлений типов и предикатов. Имя может начинаться с любой латинской буквы или символа подчеркивания "_", затем следует любая комбинация букв, цифр и символа "_".

При образовании имен, необходимо учитывать следующие правила:

- имена строковых констант должны начинаться с маленькой буквы либо они должны быть заключены в кавычки;
- имена переменных должны начинаться с большой буквы или символа подчеркивания "_".

Имена предикатов могут быть записаны с любой буквы или символа подчеркивания. Имеется следующие основные стандартные типы данных:

symbol	Символьная строка начинается со строчной буквы или заключена в кавычки
string	символьная строка, но имеющая другое внутреннее представление
char	Отдельный символ, заключенный в апострофы
integer	Целое число в диапазоне от -32768 до 32767

real	Вещественное число в диапазоне степеней 10 от -307 до +308

Константы Пролога должны быть записаны:

а) либо с маленькой буквы (исключая кириллицу):

fact1, summa, person ;

б) либо стоять в одинарных кавычках (отдельный символ) или бинарных кавычках (строковая константа):

'c' , "summa=", "сумма" ;

в) либо они являются числами, целыми или вещественными:

25, 0.5, 3.2e-4 .

Таким образом, константы могут быть любого из стандартных типов Пролога. В программе тип констант явно не указывается.

Переменные – это цепочки, состоящие из букв, цифр и символа подчеркивания. Имя переменной всегда начинается с прописной буквы или с символа подчеркивания:

X, Summa, List_of_members, _x23.

Переменная может иметь один из стандартных типов, или тип ее определяется в секции описания областей определения (типов) domaims. Можно также использовать так называемую анонимную переменную, которая записывается в виде одного символа подчеркивания.

Программа на Прологе состоит из нескольких программных секций (разделов), каждой из которых предшествует ключевое слово:

Domains	Объявление областей определение данных (типов)
Database	Объявление предикатов базы данных
Predicates	Объявление предикатов
Clauses	Определение фактов и правил

Goal	Цель

В программе необязательно наличие всех секций. Обычно в небольшой программе, находящейся в стадии отладки, присутствуют разделы Predicates и Clauses, иногда можно обойтись без них (например, когда проверяется работа стандартного предиката). Для больших программ появляется необходимость в секции Domains для объявления собственных имен типов, списков или сложных структур данных.

Порядок следования программных секций, как правило, соответствует порядку, приведенному в таблице, т.е. предикаты должны быть прежде объявлены (в секции Predicates), а затем следует их реализация (программирование) в секции Clauses. Цель в программе, если она есть, может стоять в любом месте, Пролог перебирает весь код в поисках цели, и только если ее не найдет, запросит цель отдельно. Обычно цель записывают в конце программы как логическое ее завершение, но при записи цели в середине, нужно записать ключевое слово, соответствующее продолжению программы, например Clauses.

В программе может использоваться только одна цель; если цель не указана, Пролог цель запросит. Прочих программных секций может быть сколько угодно. Можно описать несколько динамических баз данных или несколько объявлений предикатов с дальнейшей их реализацией.

Ключевые слова разделов можно записывать прописными и строчными буквами. Комментарии к программе записываются либо парами символов слэш-звездочка:

```
/* это комментарий, он занимает несколько строк */
```

либо остаток строки записывается после знака процента:

```
% это тоже комментарий
```

Контрольные вопросы

1. Что такое bektreking?
2. В чем смысл хорновского дизъюнкта?
3. На чем основывается вывод в языке Пролог?
4. С какой области программы начинается ее выполнение?
5. Какие секции содержит программа на языке Пролог?

Глава 8. Инструментальные средства разработки СИИ

Необходимость использования средств автоматизации для создания СИИ осознана разработчиками таких систем уже давно. Средства поддержки разработки интеллектуальных систем в развитии прошли стадии, подобные для систем автоматизации программирования.

Первая из них как бы повторяет «классический» путь развития средств автоматизации программирования: автокоды — языки высокого уровня — языки сверхвысокого уровня — языки спецификаций. Эту стадию называют восходящей стратегией в области разработки интеллектуальных систем.

Вторая стадия, нисходящая, связывается со специальными средствами, уже изначально ориентированными на определенные классы задач и методов ИИ.

Интенсивно развиваются работы по созданию инструментальных систем, предназначенных для быстрого проектирования и разработки разнообразных интеллектуальных систем. Общая идея тут заключается в том, чтобы создать некоторую систему-прототип, затем улучшать ее, развивая, и затем использовать для решения задач в конкретной предметной области.

Существуют следующие направления развития средств автоматизации и разработки СИИ:

Разработка "Оболочек", пустых версий существующих экспертных систем без базы знаний. Примером такой оболочки может служить EMYCIN (Empty MYCIN - пустой MYC1N), представляющая собой незаполненную экспертную систему MYCIN.

В системе-прототипе должны быть заранее зафиксированы все средства заполнения базы знаний и манипулирования знаниями, но база знаний не заполнена. Для настройки оболочки на некоторую предметную область, инженер по знаниям должен, используя готовую форму представления знаний, ввести в базу знаний всю необходимую информацию о предметной области. После этого система-прототип превращается в готовую систему искусственного интеллекта (СИИ).

Достоинство оболочек в том, что они не требуют работы программистов для создания готовой СИИ, нужны специалисты в предметной области для заполнения базы знаний. Однако если некоторая предметная область плохо укладывается в модель, используемую в некоторой оболочке, заполнять базу знаний в этом случае затруднительно, поэтому интерес к "пустым" системам уменьшается. Даже для однотипных предметных областей переход из одной области к другой может потребовать модификации тех или иных средств для манипулирования знаниями, а иногда и формы представления знаний. Поэтому основные усилия разработчиков направлены на создание таких инструментальных систем, в которых имеется возможность при переходе к конкретной системе варьировать в достаточно широких пределах формой представления знаний и способов манипулирования ими.

Инструментальные пакеты для СИИ. Анализ существующих инструментальных систем показывает, что сначала в области ИИ активно велись работы по созданию интеллектуальных систем автоматизированного синтеза исполнительных программ, потому что инструментарий ИИ является, по существу, эволю-

ционным развитием систем автоматизации программирования. При этом основная доля мощности и интеллектуальности такого инструментария связывалась не с его архитектурой, а с функциональными возможностями отдельных компонентов той или иной технологической среды. Большое значение при разработке инструментария для СИИ уделялось и удобству сопряжения отдельных компонентов.

Эти системы отличает:

- интеграция поддержки разработки СИИ.
- сбалансированный выбор необходимых блоков, что приводит к возможности автоматизации программирования СИИ

В середине 80-х годов разработана интегрированная среда **ART**, поддерживающая технологию проектирования систем, основанных на правилах, имеющая разнообразие типов правил. Различные типы правил вводятся с помощью мощного механизма Viewpoin, являющийся близким к системе, основанной на истинности предположений. Возможные ограничения в использовании ART вызваны не свойствами самой системы, а базовой методикой представления знаний.

ART часто представляют лучшей программной средой для создания экспертных систем, но следует понимать, что хорошо эта среда отвечает только требованиям поверхностных знаний. Благодаря компилятору правил, система вывода в ART является быстрой по своей природе. Стратегии выбора структуры управления поиском решений обеспечиваются, а гибкость в управлении поиском остается инженеру по знаниям. Чисто декларативные таксономические фреймы языка интегрируются с системой правил, но в ART не существует действительно процедурных фреймов, которые могли бы позволить объединить предметные описания с продукционными.

Первые версии ART опирались на язык ЛИСП, последние реализованы непосредственно на С, что увеличивает эффективность периода исполнения ART.

CLIPS как среда построения СИИ. Инструментальной средой построения СИИ, основанных на модели представления знаний, является среда CLIPS (C Language Integrated Production System – язык С, интегрированный с продукционной системой).

Появилась CLIPS в 1984 г., когда в отделе искусственного интеллекта Центра космических исследований NASA начались работы по созданию компьютерных программ, моделирующих человека-эксперта, во время мониторинга и диагностики космических систем и комплексов различного назначения. С момента разработки он был неоднократно модернизирован, введены процедурные и объектно-ориентированные парадигмы, поддержка модульной структуры программ и многое другое.

На текущий момент CLIPS является мощным инструментом создания экспертных систем, распространяется бесплатно, а также доступен его исходный программный код. С домашней страница проекта (http://sourceforge.net/projects/clipsrules/files/CLIPS/) можно свободно загрузить текущую версию среды разработки. Язык CLIPS и его среда разработки хорошо документированы. Суммарный объем технических описаний приближается к тысяче pdf-страниц [8].

Особенность CLIPS состоит в том, что она имеет встроенный механизм вывода на продукционной базе знаний. Представляя логически полную среду, содержащую встроенный редактор и средства отладки, CLIPS является оболочкой ЭС. CLIPS использует продукционную модель представления поверхностных-знаний и поэтому содержит три основных элемента:

1. список фактов
2. базу знаний
3. блок вывода

Принципиальным отличием данной системы от аналогов является то, что она полностью реализована на языке С. В CLIPS используется оригинальный подход в программировании, ориентированный на разработку ЭС. Кроме того, CLIPS поддерживает еще две парадигмы программирования: объектно-ориентированную и процедурную.

Для представления информации в CLIPS предусмотрено восемь простых типов данных: float, integer, symbol, string, external-address, fact-address, instance-name и instance-address. Для представления числовой информации используются типы float и integer, символьной - symbol и string.

При записи числа могут использоваться только цифры (0-9), десятичная точка (.), знак (+) или (-) и (e) при экспоненциальном представлении. Число сохраняется либо как целое, либо как действительное. Любое число, состоящее только из цифр, перед которыми может стоять знак, сохраняется как целое (тип integer представляется внутри CLIPS как тип языка C long integer). Все остальные числа сохраняются как действительные (float - C double float). Последовательность символов, не удовлетворяющих числовым типам, обрабатывается как тип данных symbol.

Тип данных symbol в CLIPS - последовательность символов, состоящая из одного или нескольких любых печатных символов кода ASCII. Как только в последовательности символов встречается символ-разделитель, symbol заканчивается. Следующие символы служат разделителями: любой непечатный ASCII символ (включая пробел, символ табуляции, CR, LF), двойные кавычки,"(",")",

"&", "|", "<","~",";". Символы-разделители не могут включаться в symbol за исключением символа "<", который может быть первым символом в symbol. CLIPS различает регистр символов.

Тип данных string - это последовательность символов, состоящая из нуля и более печатных символов и заключенная в двойные кавычки. Если внутри строки встречаются двойные кавычки, то перед ними необходимо поместить символ (\). То же справедливо и для самого (\).

Под функцией в CLIPS понимается фрагмент исполняемого кода, с которым связано уникальное имя и который возвращает полезное значение или имеет полезный побочный эффект (например, вывод информации на экран).

Существует несколько типов функций. Пользовательские и системные функции - это фрагменты кода, написанные на внешних языках (например, на C) и связанные со средой CLIPS. Системными называются те функции, которые были определены изначально внутри среды CLIPS. Пользовательскими называются функции, которые были определены вне CLIPS.

Хотя CLIPS и не ориентирована на численные вычисления, в ней предусмотрен ряд стандартных арифметических и математических функций. Среди них:

+ Сложение

- Вычитание

* Умножение

/ Деление

* * Возведение в степень

Abs Определение абсолютного значения

Sqrt Вычисление квадратного корня

Mod Взятие по модулю

Min Нахождение минимума

Max Нахождение максимума

Конструкция deffunction позволяет пользователю определять новые функции непосредственно в среде CLIPS., которые выглядят и работают подобно остальным функциям, однако они выполняются не напрямую, а интерпретируются средой CLIPS. Вызовы функций в CLIPS имеют префиксную форму: аргументы функции могут стоять только после ее названия. Вызов функции начинается с открывающейся скобки, за которой следует имя функции, затем идут аргументы, каждый из которых отделен одним или несколькими пробелами. Аргументами функции могут быть данные простых типов, переменные или вызовы других функций. В конце вызова ставится закрывающаяся скобка.

В CLIPS существует несколько описывающих конструкций:

defmodule, defrule, deffacts, deftemplate, defglobal, deffunction, defclass, definstances, defmessage-handler, defgeneric.

При записи все они заключаются в скобки. Определение конструкции отличается от вызова функции главным образом по производимому эффекту. Обычно вызов функции оставляет состояние среды CLIPS без изменений (за рядом исключений, когда речь идет о функциях сброса, очистки, открытия файла и т.п.). Определение конструкции направлено на изменение состояния среды путем

внесения изменений в базу знаний CLIPS. В отличие от функций конструкции никогда не возвращают значений.

Все конструкции (за исключением defglobal) позволяют размещать комментарии сразу вслед за именем конструкции. Кроме того, комментарии могут вставляться в код CLIPS при помощи точки с запятой (;). Все, что следует за (;) до конца строки, будет игнорироваться CLIPS. Если (;) стоит первым символом в строке, то вся строка считается комментарием.

Факты являются одной из основных форм представления информации в системе CLIPS. Каждый факт представляет фрагмент информации, который был помещен в текущий список фактов, называемый fact-list. Факт представляет собой основную единицу данных, используемую правилами. Ввод фактов выполняется командой **accert.** Количество фактов в списке и объем информации, который может быть сохранен в факте, ограничивается только размером памяти компьютера. Если при добавлении нового факта к списку обнаруживается, что он полностью совпадает с одним из уже включенных в список фактов, то эта операция игнорируется (хотя такое поведение можно изменить).

Факт может описываться индексом или адресом. Всякий раз, когда факт добавляется (изменяется), ему присваивается уникальный целочисленный индекс. Индексы фактов начинаются с нуля и для каждого нового или измененного факта увеличиваются на единицу. Каждый раз после выполнения команд reset и clear выделение индексов начинается с нуля. Факт также может задаваться при помощи адреса. Адрес факта может быть получен путем сохранения возвращаемого значения команд, которые возвращают в качестве результата адрес факта (таких как assert, modify и duplicate), или путем связывания переменной с адресом факта в левой части правила.

Идентификатор факта - это короткая запись для отображения факта на экране. Она состоит из символа f и записанного через тире индекса факта. Например, запись f-10 служит для обозначения факта с индексом 10. Существует два формата представления фактов: позиционный и непозиционный.

Позиционные факты состоят из выражения символьного типа, за которым следует последовательность (возможно, пустая) из полей, разделенных пробелами. Вся запись заключается в скобки. Обычно первое поле определяет "отношение", которое применяется к оставшимся полям. Например: (the pump is on) (altitude is 10000 feet) (grocery_list bread milk eggs).

Поля в позиционных фактах могут быть любого простого типа (за исключением первого поля, которое всегда должно быть типа symbol), на порядок полей также не накладывается никаких ограничений. Следующие символьные выражения зарезервированы и не должны использоваться как первое поле любого факта (позиционного или нет): test, and, or, not, declare, logical, object, exists u forall.

Для того чтобы обратиться к информации, содержащейся в позиционном факте, пользователь должен знать не только какие данные содержатся в факте, но и то, в каком поле они хранятся. Непозиционные (шаблонные) факты дают возможность пользователю абстрагироваться от структуры факта, задавая имена каждому из полей факта. Для задания шаблона, который затем может использоваться при доступе к полям по именам, используется конструкция deftemplate.

Конструкция deftemplate позволяет наряду с определением именованных полей, или слотов, вводить имя шаблона. В отличие от позиционных фактов слоты шаблонного факта могут быть ограничены по типу, значению, числовому диапазону. Кроме того, для любого слота можно определить значения по умолча-

нию. Слот состоит из открывающейся скобки, за которой следует имя слота, полей (могут отсутствовать) и закрывающейся скобки. Заметим, что слоты не могут использоваться в позиционных фактах, так же как позиционные поля не могут использоваться в шаблонных фактах. Общая структура конструкции deftemplate такова:

(deftemplate)

(slot-1)

(slot-2)

…

(slot-N)

Шаблонные факты отличаются от позиционных по первому полю в факте. Первое поле всех фактов должно быть типа symbol, но если это символьное выражение соответствует имени шаблона, то этот факт - шаблонный. За первым полем шаблонного факта следует список из нуля или более слотов. Как и позиционные, шаблонные факты заключаются в скобки. Далее приведено несколько примеров шаблонных фактов:

(client (name "Joe Brown") (id X9345A))

(point-mass (x-velocity 100) (y-velocity -200))

(class (teacher "Martha Jones") (#-students 30)

(room "37A"))

(grocery-list (#-of-items 3) (items bread milk eggs))

Порядок следования слотов в шаблонном факте не важен.

Факты могут добавляться к списку фактов (с помощью команды assert), удаляться из него (с помощью команды retract), изменяться (с помощью команды modify) и дублироваться (с помощью команды duplicate) самим пользователем или программой. Например:

(assert (light green))

Кроме того, конструкция deffacts позволяет определить множество исходных или априорных знаний в виде набора фактов. Например:

(deffacts walk "Some facts about walking"

(status walking)

(walk-sign walk))

Когда производится сброс состояния среды CLIPS (с помощью команды reset) все факты, описанные в конструкции deffacts, добавляются к списку фактов. Кроме того, по этой команде в список фактов заносится исходный факт (initial-fact). Этот факт включается в список фактов всегда с идентификатором f-0.

Одним из основных методов представления знаний в CLIPS являются правила. Правила используются для представления эвристик, определяющих ряд действий, которые необходимо выполнить в определенной ситуации. Разработчик СИИ определяет совокупность правил, используемых для решения проблемы. Правило состоит из двух частей: антицедента (условия), который является аналогом условия в if-then операторе и записывается слева, и консеквента (заключения), который является аналогом then части этого оператора и записывается справа.

Левая часть правила представляет собой ряд условий (условных элементов), которые должны выполняться, чтобы правило было применимо. В CLIPS принято считать, что условие выполняется, если соответствующий ему факт присутствует в списке фактов. Одним из типов условных элементов может быть образец. Образцы состоят из набора ограничений, используемых для описания того, какие факты удовлетворяют условию, определяемому образцом. Процесс сопоставления фактов и образцов выполняется блоком вывода CLIPS, автоматически сопоставляющий образцы, исходя из текущего состояния списка фактов, и определяет, какие из правил являются применимыми. Если все условия правила выполняются, то оно активируется и помещается в список активированных правил.

Если левая часть правила пуста, то для его активации необходимо наличие в списке фактов исходного факта (initial-fact). Такие безусловные правила часто используются для того, чтобы инициировать работу программы. Поэтому перед запуском таких программ необходимо произвести сброс состояния среды CLIPS.

Правая часть правила представляет собой совокупность действий, выполняемых, если правило применимо. Действия, описанные в применимых правилах, выполняются тогда, когда блок вывода CLIPS получает команду начать выполнение применимых правил. Если существует множество применимых правил, то для того, чтобы выбрать правило, действия которого должны быть выполнены, блок вывода использует глубинную стратегию разрешения конфликтов. Действия, описанные в выбранном правиле, выполняются (при этом список применимых правил может измениться), а затем блок вывода выбирает другое правило и т. д. Этот процесс продолжается до тех пор, пока не остается ни одного применимого правила, т. е. пока список активированных правил не окажется

пуст. Для выбра приоритета правил используется параметр salience, которому может быть присвоено любое целочисленное значение в диапазоне [-10 000, 10 000]

Блок вывода постоянно отслеживает все правила, условия которых выполняются, и, таким образом, правило может быть выполнено в любой момент, как только оно становится применимым. Для определения правил используется конструкция defrule:

(defrule rule_name "optional_comment"

(patten_1)

(patten 2)

…….

(patten_N)

=>

(action_1)

(action_2)

(action_M))

В CLIPS для хранения значений используются переменные. В отличие от фактов, являющихся статическими, или неизменными, содержание переменной динамично и изменяется по мере того, как изменяется присвоенное ей значение. Идентификатор переменной всегда начинается с вопросительного знака, за которым следует ее имя. В общем случае формат переменной выглядит следующим образом:

?

Примеры переменных:

?x ?sensor ?noun ?color

Перед использованием переменной ей необходимо присвоить значение. Все переменные, кроме глобальных, считаются локальными и могут использоваться только в рамках описания конструкции. К этим локальным переменным можно обращаться внутри описания, но они не определены вне него.

Чаще всего переменные описываются и получают значения в левой части правила. Например:

```
(defrule make-quack

(duck-sound ?sound)

=>

(assert (sounds-is ?sound) )
```

Получив значение, переменная сохраняет его неизменным при использовании как в левой, так и в правой части правила, если только это значение не изменяется в правой части при помощи функции bind.

```
(defrule addition

(numbers ?x ?y)

=>

(assert (answer (+ ?x ?y)))

(bind ?answer (+ ?x ?y))
```

```
(printout t "answer is " ?answer crlf))
```

Кроме значения самого факта, переменной может быть присвоено значение адреса факта. Это может оказаться удобным при необходимости манипулировать фактами непосредственно из правила. Для такого присвоения используется комбинация "<-". Следующий пример иллюстрирует присвоение переменной значения адреса факта и ее последующее использование:

```
(defrule get-married

?duck <- (bachelor Dopey)

=>

(retract ?duck))
```

Для определения глобальных переменных, которые видны всюду в среде CLIPS, используется конструкция defglobal. К глобальной переменной можно обратиться в любом месте, и ее значение остается независимым от других конструкций. Глобальные переменные CLIPS подобны глобальным переменным в процедурных языках программирования, но они значительно слабее типизированы (на них не налагается ограничения хранения данных только одного типа. CLIPS не ориентирован на численные вычисления, но в нём предусмотрены стандартные математические и арифметические функции: +, –, *, /, ** (возведение в степень), **Abs**, **Sqrt**, **Mod**, **Min**, **Max**.

Контрольные вопросы

1. Какая модель знаний поддерживается в среде CLIPS?
2. Как создаются правила в CLIPS?
3. Какие языки поддерживаются в среде ART?

4. Перечислите функции конструктора deffunction.
5. Какие инструментальные системы для разработки СИИ существуют?

Глава 9. Разработка систем искусственного интеллекта

Проектирование систем искусственного интеллекта— это итеративный и эволюционный процесс, в котором участвуют такие специалисты, как: эксперт, обладающий знаниями о предметной области, а также специалисты в области искусственного интеллекта — инженеры знаний, аналитик и программист. В зависимости от объема и трудоемкости работ группа может состоять из трех — шести человек.

При оценке проблемной области на этапе проектирования СИИ необходимо учитывать следующие факторы: легкость сбора данных, возможность представления данных, затраты на разработку СИИ, наличие экспертов, наличие необходимых ресурсов (компьютеров, программистов, программного обеспечения и т. д.).

Процесс построения СИИ разделяют на 4 основных этапа (рис. 9.1)

1. Идентификация определения задач и идентификация их характеристик. На этом этапе устанавливаются задачи, которые предполагается решать, их характеристики и особенности. Разрабатывается техническое задание на проектируемую систему и очерчивается круг пользователей системы. В результате вырабатываются определенные требования.

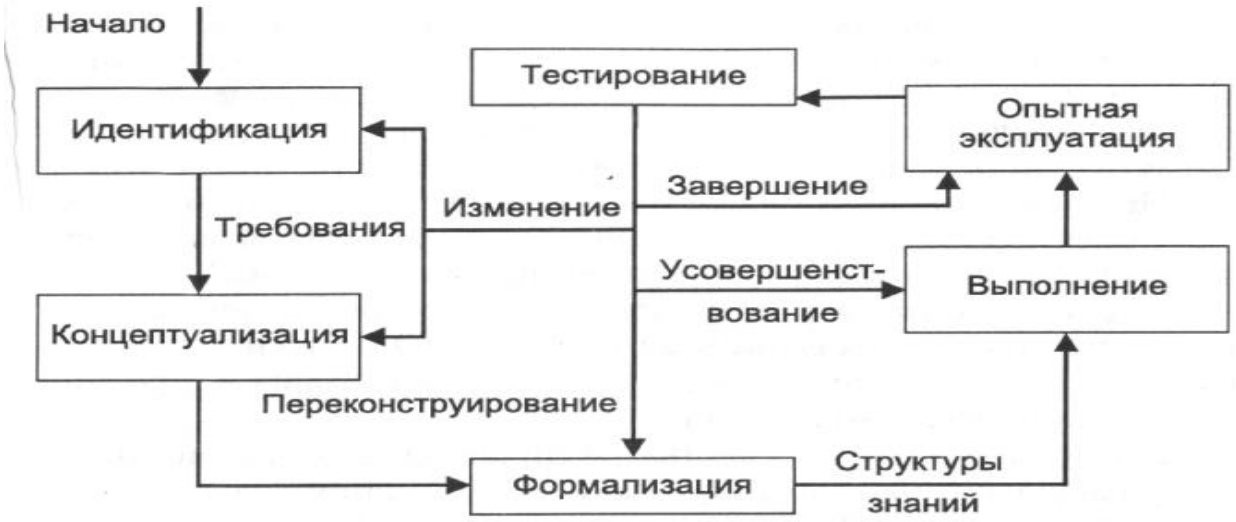


Рис. 9.1 - этапы разработки СИИ

2. Концептуализация предметной области, отражающая знания экспертов, позволяет анализировать тип знаний, которыми оперирует эксперт в процессе принятия решений. Инженер знаний определяет формальные средства представления знаний и процедуры получения решений, соответствующие характеру рассуждений эксперта при выводе решения.

3 . Формализация представления знаний (модели) и определение механизма вывода решений. Эти компоненты моделирования в значительной степени влияют на успешное решение поставленной задачи по проектированию системы. После того как правила сформулированы и представлены на выбранном языке представления, они заносятся инженером знаний в БЗ.

4. Тестирование системы.

Работоспособность системы определяется путем на этапах выполнения и опытной эксплуатации путем решения конкретных проверочных задач. При выявлении различных недостатков происходит обращение к тому или иному этапу разработки в зависимости от характера недостатков. При отсутствии каких-либо знаний в СИИ или их недостаточной определенности возвращаются к этапу 4 и по возможности вносят поправки. В случае если какие-либо знания, представ-

ленные экспертом практически невозможно представить в пределах формализма выбранной модели представления знаний, то возвращаются к этапу 3 и выбирают альтернативные модели или схемы представления знаний. 5.

Стадии существования СИИ (жизненные циклы системы) соответствуют уровню готовности системы, завершенности ее функциональных возможностей, реализуемых инструментарием. Определяют следующие стадии существования СИИ: демонстрационный прототип; исследовательский прототип; действующий прототип; промышленная система; коммерческая система.

Демонстрационный прототип — это состояние разработанности системы, когда она решает часть проблемных задач. При разработке демонстрационного прототипа стремятся достичь противоречивых целей: с одной стороны, система должна выполнять задачи, которые бы характеризовали ее возможности, с другой стороны, эту стадию стремятся пройти как можно быстрее. Работа демонстрационного прототипа может быть признана удовлетворительной, если он оперирует минимальным набором правил, достаточным для решения некоторых задач. Время разработки колеблется от двух месяцев до года.

Исследовательский прототип проектируется в течение 1,5 ...2 лет. На этой стадии развития системы ее БЗ уже содержит несколько сотен правил, которые достаточно адекватно описывают предметную область.

Действующий прототип систем искусственного интеллекта осуществляет качественный вывод решений на расширившемся пространстве правил, достигшем порядка 1000. Поэтому для вывода сложных решений требуются большие ресурсы времени и памяти.

Промышленные системы обеспечивают высокий уровень качества решения проблем предметной области при значительных уменьшениях времени решения и требуемой памяти. Количество правил возрастает не столь значительно по сравнению с действующим прототипом. На этой стадии происходит преобразо-

вание действующего прототипа за счет расширения числа правил и совершенствования систем искусственного интеллекта на базе использования более эффективных, инструментальных средств. Это требует примерно 3 ... 4 года.

Коммерческая система предназначена в основном для продажи. Она является либо проблемно-ориентированной, либо проблемно-независимой.

Анализ предметной области и методы приобретения знаний. Предметную область можно определить как сферу человеческой деятельности, выделенную и описанную согласно установленным критериям. Специфика предметной области оказывает влияние на функционирование СИИ, выбор метода представления знаний, способов рассуждения о знаниях, и т. д.

Приобретение знаний реализуется с помощью двух функций: получения информации и ее систематизация. При этом возможны различные формы приобретения знаний, а также различные формы получаемой информации.

Форма представления знаний для их использования определяется внутри системы, поэтому форма информации, которую она может принимать зависит от того, какие способности имеет система для формализации информации до уровня знаний.

Предложена следующая классификация этапов обучения СИИ, соответствующих возможностям компьютеров к формализации знаний.

- получение информации без логических выводов. Это обучение без выводов или механическое запоминание для получения информации, при котором необязательны функции выводов, а полученная информация в виде программ или данных используется для решения задач в неизменном виде.

- получение извне информации, уже представленной в виде знаний, т.е. в форме, которую можно использовать для выводов. Обучающейся системе необходимо иметь функцию преобразования входной информации в формат, удобный для дальнейшего использования и включения в базу знаний.

- обучение по примерам.

1. Параметрическое обучение.
2. Обучение на основе выводов по аналогии.
3. Обучение на основе выводов по индукции – эвристическое обучение.

Параметрическое обучение. Наиболее простая форма обучения по примерам или наблюдениям состоит в определении общего вида правила, которое должно стать результатом вывода, и последующей корректировки входящих в это правило параметров в зависимости от данных. При этом используются психологические модели обучения, системы управления обучением и другие системы.

Обучение на основе выводов по аналогии (прецендентам). Приобретение новых понятий возможно путем преобразования существующих знаний, похожих на те, которые собираются получить. Много примеров, когда новые понятия или технические приемы приобретаются с помощью аналогий. Выводы по аналогии – один из важных объектов исследования ИИ.

Обучение на основе выводов по индукции – эвристическое обучение. Обучение по индукции с использованием выводов высокого уровня, как при обучении по аналогии. В процессе этого обучения путем обобщения совокупности имеющихся данных выводятся общие правила. Обучение этой категории включает открытие новых правил, построение теорий, создание структур и другие действия, причем модель теории или структуры, которые следует создать, заранее не задаются, поэтому их необходимо разработать. Индуктивные выводы возможны в случае, когда представление результата вывода частично определяется из представления входной информации. Приобретение знаний на метауровне, знаний, основой которых является информация по управлению решением задач с использованием знаний на объектном уровне. Для знаний на метауровне

пока не установлены ни формы представления и использования, ни связь со знаниями на объектном уровне, ни другая техника их систематизации.

Источники знаний. Выявление источников знаний и работа с ними - основная задача инженера знаний, выполняющего основные функции при разработке БЗ. Он должен хорошо ориентироваться в проблемной области и быть хорошим психологом, чтобы общаться с экспертом в процессе приобретения знаний. Вместе с тем он должен знать и возможности программного обеспечения компьютеров, чтобы структурировать знания для хранения и работы с ними.

Основным источником знаний о проблемной области является человек-эксперт (стратегический источник). Инженер знаний работает с ним в режиме диалога или интервью и формирует необходимый объем знаний и сведений для работы с объектом. Для некоторых задач источниками дополнительной информации являются книги, технологические описания, инструкции, документы (фактографические источники знаний). Используются также методы так называемого «мозгового штурма», когда группа специалистов в определенной обстановке в оперативные сроки генерирует необходимую информацию, помогающую разрешение проблемы и лучшему исследованию предметной области. Для некоторых областей применения СИИ знания об объекте можно формировать путем использования статистической обработки информации и информации о результатах имитационных экспериментов. В последнее время все чаще начинают использовать методы автоматизированного заполнения БЗ.

Другим важным источником знаний является Интернет. Помимо традиционного поиска необходимой информации и знаний в Интернет, в настоящее время в процесс поиска знаний вовлекаются интеллектуальные агенты. Интеллектуальные агенты (программные роботы) для нахождения или доступа к знаниям могут работать внутри документов или искать в Web, а затем возвращаться со знаниями в нужном формате.

Работа с экспертами и проблема извлечения знаний

Характеристики СИИ в контексте приобретения знаний

1. Уровень языка, в котором осуществляется приобретение знаний:

– формализованный язык;

– ограниченный естественный язык (ЕЯ);

– язык пиктограмм и изображений;

– ЕЯ и язык изображений

2. Тип приобретаемых знаний:

– данные в виде таблиц, содержащих значения входных и выходного атрибутов (по которым индуктивными методами строится дерево вывода);

– специализированные правила;

– общие и специализированные правила.

3. Вид приобретаемых данных:

– атрибуты со значениями;

– объекты;

– классы структурированных объектов и их экземпляры, получающие значения атрибутов по наследованию

Некоторые методы извлечения знаний

Мозговой штурм широко применяется для генерирования новых идей путем творческого сотрудничества группы специалистов, пытающихся разрешить проблему. Участники выдвигают и развивают собственные идеи, стимулируя появление новых и комбинируя их. Для обеспечения максимального эффекта «мозговой штурм» должен подчиняться определенным правилам и основываться на строгом разделении во времени процесса выдвижения идей и процесса их обсуждения и оценки.

На первой стадии штурма запрещается осуждать выдвинутые идеи и предложения (считается, что критические замечания уводят к частностям, прерывают

творческий процесс, мешают выдвижению идей). Роль аналитика состоит в том, чтобы активизировать творческое мышление участников заседания и обеспечить выдвижение возможно большего числа идей.

Извлечение знаний из текстов включает следующие шаги.

1. Составление «базового» списка литературы для ознакомления с предметной областью.

2. Выбор текста для извлечения знаний.

3. Беглое знакомство с текстом. Проведение консультации со специалистами для определения значений незнакомых слов.

4. Формирование первой гипотезы о макроструктуре текста.

5. Внимательное прочтение текста, выбор ключевых слов и выражений, определяющих «смысловые вехи».

6. Определение связи между ключевыми словами, разработка макроструктуры текста в форме графа или реферата.

7. Формирование нового представления знаний на основании макроструктуры текста.

Интеллектуальные системы создаются совместно со специалистами, передающих свои знания о процессах и объектах, поясняющих схему рассуждений по выбору решений конкретных задач, приводящих неформализуемые факторы, которые необходимо учитывать. Процесс работы с экспертами или специалистом состоит в извлечении или приобретении знаний. Процесс этот сложный, трудоемкий, содержит факторы технического, психологического, производственного и социального характера. В течение долгого времени когнитолог работает совместно с экспертом, определяя задачи, выявляя наиболее важные понятия, определяя и формулируя правила отношений между понятиями.

Инженеру знаний обычно приходится работать с плохо определенными задачами, в которых отсутствует детерминированная внутренняя структура. В

своей практической деятельности эксперты при решении этих задач используют эвристики - эмпирические правила, применяемые в случаях, когда условия оперативности решения задачи или недостаточное понимание существа проблемы делают невозможным анализ всех параметров задачи. Поэтому в СИИ стремятся заложить эвристические процедуры.

Трудным моментом в работе инженера знаний является оказание помощи эксперту при попытках структурировать предметные знания, определить и формализовать предметные концепции.

Основная особенность извлечения знаний для интеллектуальных и экспертных систем в том «Откуда, что и как извлекать?». Изучение особенностей эксперта – это не только одна из возможных целей, к которой следует стремиться в человеко-машинных системах и пользователя, это важно для того, чтобы заранее знать особенности партнера, от которого мы будем получать знания.

Знания эксперта систематизируются вместе с внешней моделью задачи и мгновенно выбираются из памяти в ответ на каждую конкретную ситуацию и лишь в ситуации, с которой эксперт сталкивается впервые, порядок их выборки будет определяться сознательно но наиболее общим закономерностям.

Если функции системы уже определены, то естественно, самое важное – получить правила выводов, которые необходимы для реализации этих функций.

Прежде всего, это базовая структура. Главное перечислить объекты, понятия и атрибуты, формирующие базовую структуру проблемной области, и знать свойства области. Связь между объектами, понятиями и атрибутами организуется через правила вывода.

Немаловажны и критерии разумности, заключающиеся в том, как эксперт решает некоторую проблему именно данным способом, может ли этот способ имеет высокую эвристическую ценность.

Средства, используемые экспертом, например модели принятия решений, используемые им при принятии решений (Табл.9.1).

Таблица 9.1 - методы извлечения знаний из предметного эксперта

Метод	**Описание**
Наблюдение на рабочем месте	Наблюдать за экспертом, решающим реальные задачи на своем рабочем месте.
Обсуждение задач	Выявить виды данных, знаний и процедур, необходимых для решения конкретных задач.
Описание задач	Попросить эксперта описать прототипную задачу для каждой категории возможных ответов.
Анализ задачи	Представить эксперту ряд реалистических задач для решения вслух с целью выявить логические основания конкретных шагов рассуждения.
Доводка системы	Попросить эксперта предоставить вам несколько задач для решения и с использованием правил, выявленных во время интервью.
Оценивание системы	Попросить эксперта проверить работу системы и подвергнуть критике правила и структуру управления прототипной системой.
Проверка системы	Предоставить примеры, решенные экспертом и прототипом системы, другим независимым экспертам для сравнения и оценки.

Технику извлечения знаний можно разделить на шесть основных классов: опрос с наводящими вопросами, структурированный опрос, самонаблюдение, самоотчет, диалог, критический обзор. Каждый класс, в свою очередь состоит из нескольких технических методов.

Способом приобретения знаний системой является автоматизация извлечения знаний и запись их в БЗ. Неавтоматизированный сбор знаний специалистов — трудоемкий процесс. В связи с этим, в развитых СИИ предусматриваются вспомогательные средства для приобретения знаний.

Интернет используются для облегчения процесса извлечения знаний. Электронное интервьюирование может проводиться, если инженер знаний и эксперты находятся в различных местах или эксперты могут утверждать и сопровождать базы знаний на расстоянии. Посредством Интернет могут быть достигнуты документированные знания. Проблемой является идентификация знаний: задача, которая может быть облечена интеллектуальными агентами.

Актуальной является также задача автоматической структуризации неформальных знаний, доступных в Интернет через распределенную гипермедиа систему – Web или semantic WEB. Технология гипермедиа через Web обеспечивает идеальный подход для развития систем, основанных на знаниях путем расширения возможностей каналов человеко-машинного взаимодействия. Этот новый подход к интеграции технологии гипермедиа с извлечением знаний имеет дело со знаниями до того, как они будут формализованы.

Многие web-механизмы поиска включают интеллектуальных агентов для идентификации и поставки требуемой информации по индивидуальным потребностям и запросам.

Причина экспоненциального роста количества информации, обеспечиваемого через web-механизмы, вызывает развитие методов структуризации информации в распределенных гипермедиа системах.

Практический опыт решения задачи приобретения знаний привел к развитию методов и программных средств, призванных упростить процесс приобретения знаний. Эти средства и методы для приобретения знаний могут быть разделены на три категории: редакторы и интерфейсы для формирования баз зна-

ний, средства для объяснения различных аспектов работы, средства для модификации баз знаний.

Оценка пространства поиска решений. На основе изучения проблемной области и характера получаемых знаний оценивают пространство поиска решений. Размеры пространства поиска решений определяются многими факторами, и прежде всего - характером данных и знаний предметной области и спецификой решаемых задач.

Надежные и определенные знания и данные позволяют реализовывать монотонные БЗ, и новые знания и правила включаются в нее без пересмотра и удаления некоторых хранящихся знаний, в том числе неизменяющиеся во времени. БЗ в этом случае представляется в виде списков выведенных заключений или использование декларативных форм представления.

Для структуризации, формализации и работы с неточными неопределенными знаниями и данными используется вероятностный подход нечеткие множества и нечеткая логика.

Для представления ситуационных знаний (изменяющихся во времени) разработаны ситуационное исчисление и язык ситуационного управления.

При создании БЗ необходим выбор способа представления знаний в такой форме, чтобы имелся доступ к ней для принятия решений, планирования, узнавания объектов и ситуаций, анализа сцен, вывода заключений и других когнитивных функций.

При использовании логических моделей БЗ рассматривается как совокупность логических формул, которые обеспечивают частичное описание проблемной среды.

Семантические сети, позволяющие описывать свойства и отношения объектов событий, понятий, ситуаций или действий с помощью направленного графа,

разделяются на экстенсиональные и интенсиональные. Экстенсиональная сеть является основой БД, а интенсиональная — БЗ.

Фреймы, как и семантические сети, представляют собой декларативно-процедуральные структуры. Во многих фреймовых структурах возможна реализация наследственных отношений, при которых объекты могут наследовать атрибуты более абстрактных объектов [36]. Такая форма организации знаний позволяет экономить объем памяти.

Продукционные модели, вероятно, являются сейчас наиболее популярным способом представления знаний.

При организации знаний с использованием продукционных моделей в БЗ содержатся правила продукций, а в БД содержится информация, которая отображает текущее состояние решаемой задачи. Инициализацию необходимого правила осуществляет блок управления (в другой терминологии: интерпретатор).

Применение онтологических моделей позволяет унифицировать представление знаний, используя совокупности логики, семантики и нечеткого подхода.

При организации БЗ исходят из характера той информации, которую она должна содержать. Это некие факты, данные, представляющие собой быстро меняющуюся информацию. Другой тип информации — это модели знаний или правила, которые изменяются значительно реже данных. Кроме того, правила несут в себе содержательные сведения об объекте. Они активны и могут порождать новые факты или гипотезы из тех сведений, которыми располагает БЗ в текущий момент. В связи с этим структурно БЗ и базы данных (БД) должны взаимодействовать.

В БД хранится фактографическая информация о решаемых на объекте задачах и данные, которые относятся к указанной предметной области. представления знаний.

Помимо знаний о предметной области в БЗ должны храниться и другие типы знаний: модель мира системы, знания о пользователе, целях и т. д. Эти знания в основном содержатся на втором уровне представления в виде блоков или органических частей БП.

Контрольные вопросы

1. Назовите этапы разработки СИИ.
2. В чем заключается оценка пространства поиска решений?
3. Почему нежелательно, чтобы эксперт формировал модель знаний?
4. В чем заключается идентификация задач для СИИ?
5. В чем заключаются основные функции инженера-когнитолога?

Глава 10. Архитектура и основные составные части систем искусственного интеллекта

Существуют различные подходы к построению архитектуры систем ИИ.

Логический подход. Основой для логического подхода служит Булева алгебра, представленная в виде исчисления предикатов, в котором она расширена за счет введения предметных символов, отношений между ними, кванторов существования и всеобщности. Практически каждая система ИИ, построенная на логическом принципе, представляет собой машину вывода, где исходные данные хранятся в базе данных в виде фактов, правил, и логического вывода как отношения между ними. Каждая такая машина имеет блок генерации цели, и система вывода пытается доказать данную цель как теорему. Если цель доказана, то трассировка примененных правил позволяет получить цепочку действий, необходимых для реализации поставленной цели. Мощность такой системы определяется возможностями генератора целей и машиной доказательства теорем.

Расширить логический подход позволяет **нечеткая логика** (глава 5). Основным ее отличием является то, что правдивость высказывания может принимать в ней кроме да/нет (1/0) еще и промежуточные значения от 0 до 1.

Для большинства логических методов характерна большая трудоемкость, поскольку во время поиска доказательства возможен полный перебор вариантов. Поэтому данный подход требует эффективной реализации вычислительного процесса, и эффективная работа гарантируется при небольшом размере базы данных.

Структурный (бионический) подход заключается в моделировании структуры человеческого мозга. Основной структурной единицей в большинстве вариантов моделирования мозга является нейрон. Затем возникли и другие модели нейронных сетей (Глава 4). Эти модели различаются по строению отдельных нейронов, по топологии связей между ними и по алгоритмам обучения. Среди наиболее известных сейчас вариантов НС для реализации СИИ можно назвать НС с обратным распространением ошибки, сети Хопфилда, стохастические нейронные сети.

Эволюционный подход. При построении СИИ по этому подходу, основное внимание уделяется построению начальной модели, и правилам, по которым она может изменяться (эволюционировать). Модель может быть составлена по самым различным методам, это может быть и НС и набор логических правил и любая другая модель. После проверки моделей отбираются самые лучшие из них, на основании которых по самым различным правилам генерируются новые модели, из которых опять выбираются самые лучшие и т. д.

При построению СИИ применяется **имитационный подход** с представлением системы в виде “черного ящика". При этом моделируется интеллектуальное свойство человека копировать то, что делают другие, не вдаваясь в подробно-

сти, как это происходит. Основным недостатком имитационного подхода является низкая информационная способность большинства моделей, построенных с его помощью.

Основу СИИ составляют база знаний и механизм вывода решений. Эти компоненты определяют две основные интеллектуальные характеристики системы: способность хранить знания о чем-то и умение оперировать этими знаниями, в том числе способность обучаться, приобретать новые знания, пополнять БЗ, корректировать знания в соответствии с изменяющимися условиями и ситуацией в предметной области.

Для этого необходимы следующие процедуры:

1) накопление знаний о предметной области;
2) классификация знаний по критерию полезности и непротиворечивости;
3) структурирование знаний в конкретной области;
4) инициализация процессов получения новых знаний;
5) соотнесение новых знаний со старыми;
6) формированием логического вывода, отражающего закономерности в предметной области и накопленных знаниях;
7) логическое планирование своей деятельности;
8) осуществление вывода на основе рассуждений по прецендентам.
9) общение на естественном языке (или подмножестве профессионального языка);
10) обучение;
11) введение знаний о целях и возможностях пользователя, а также о собственных возможностях;
12) формирование по запросу пользователя объяснений своей деятельности;

13) документирование информации в форме, необходимой пользователю.

Эти процедуры реализуются архитектурой системы в виде обобщенной структурно-функциональной схемы СИИ (рис. 10.1), и определяется функциями конкретного состава интеллектуальных задач.

База знаний (БЗ) реализуют первую функцию СИИ – функцию представления и обработки знаний и состоит из блоков:

База фактов содержит факты, носящие конкретный характер:

а) факты, характеризующие текущую ситуацию, текущее состояние;

б) факты, характеризующие уже имевшие место ситуации (опыт).

База правил содержит элементарные выражения в виде часто используемых моделей, называемые **продукциями**, и содержащая закономерности, представляющие, причинно-следственные связи предметной области типа ЕСЛИ – ТО – ИНАЧЕ.

База процедур хранит прикладные программы, с помощью которых выполняются все необходимые вычисления, преобразования и другие нужные системе последовательности внутренних действий.

База закономерностей содержит различные сведения, относящиеся к особенностям той предметной области, в которой будет функционировать система (законы предметной области, эмпирические зависимости).

База знаний о себе содержит списки того, что хранится в текущий момент в остальных базах:

- сведения о том, как представляются единицы информации различного типа;
- сведения о том, как взаимодействуют отдельные части системы;
- сведения о том, как получено решение любой конкретной задачи.

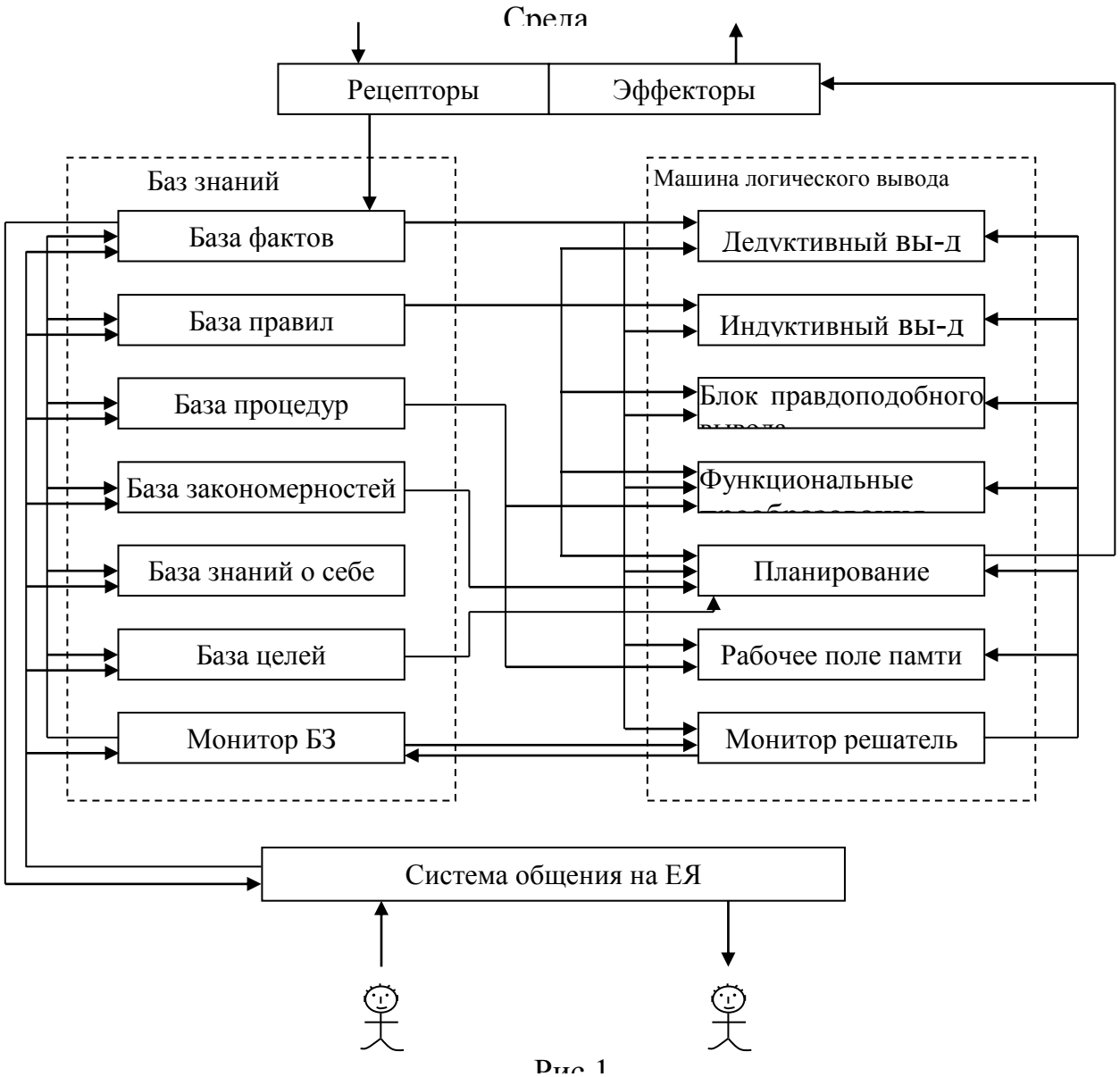


Рис. 10.1 - структурно-функциональная схема архитектуры СИИ

База целей содержит целевые структуры, позволяющие организовать процессы движения от исходных фактов, закономерностей, правил и процедур к достижению той цели, которая поступила в систему от пользователя или сформирована в самой системе в процессе ее функционирования. База целей – это способ представления знаний, как и база правил, но ориентирована на связи

объектов между собой через сообщения, что позволяет реализовывать стратегию вывода по сценариям.

Этот момент является одним из существенных отличий СИИ от систем обработки данных (СОД). В обычной СОД схема передачи управления и использования данных предопределена в программе. Обработка информации осуществляется последовательными шагами, а ветвление имеет место в заранее выбранных точках.

В СИИ ходом рассуждения управляют данные и существуют ветвления в ходе рассуждения, и правила на любом шаге оценивают ситуацию для выполнения соответствующих действий.

Монитор баз знаний – это программа управления всеми базами, входящими в базу знаний. Эта программа организует их взаимодействие между собой.

МДВ (модуль дедуктивного вывода) реализует вторую функцию СИИ – функцию рассуждений, состоящую из 7 элементов:

БДВ (блок дедуктивного вывода) реализует дедуктивные рассуждения, с помощью которых на основании общих закономерностей из базы закономерностей, конкретных фактов из базы фактов и правил вывода из базы правил выводятся новые факты.

Строгий формальный вывод существует не всегда, потому что может отсутствовать необходимая информация, могут существовать ограничения на принятые решения и т.п. Поэтому **БИВ** (блок индуктивного вывода) нужен в СИИ для организации вывода новых знаний на основе обобщений отдельных понятий и фактов, замены операций с отдельными понятиями и фактами операциями с их множествами.

В процессе индуктивного и дедуктивного выводов возможны ошибки. Для их устранения используют определенные указатели правдоподобия сформированных правил, реализуемых в **БПВ** (блок вывода правдоподобий).

Поскольку СИИ должна работать на естественном языке или его подмножестве из области профессионального языка, то этот блок и должен отражать те степени правдоподобия правил, фактов, которые имеются в естественном языке (ЕЯ) в виде выражений: «вероятно», «часто», «много раз», … В этом блоке целесообразно моделировать и правдоподобные рассуждения, т. е. элементы переноса свойств, выявленных для одних фактов и ситуаций, на другие, кажущиеся по набору признаков похожими на уже изученные (рассуждения по аналогии и ассоциации).

Для выявления необходимых знаний используют образы, т. е. форматы, определяющие условия активизации различных структурированных знаний. На каждой итерации происходят анализ и сопоставление текущей ситуации и образов с целью нахождения блоков, для которых выполняются условия активизации для действий в этой ситуации. Для организации и описания подобной процедуры введено понятие модуля, управляемого по образам.

В системах, основанных на правилах, в качестве модулей, управляемых по образам, выступают либо отдельные правила, либо блоки правил, отражающие определенный уровень или тип эвристических знаний в БЗ.

Процедуры сопоставления с образами и определения правил, соответствующих текущему состоянию рабочей области, определены логикой работы управляющей структуры (монитора-решателя). В связи с этим весь процесс реализации стратегии вывода проходит через четыре основные стадии: выбор, сопоставление, разрешение конфликтов, выполнение.

На стадии выбора выбираются модули БЗ и данные из рабочей памяти, относящиеся к рассматриваемой ситуации, и за счет этого происходит сокращение

пространства поиска, так как рассматриваются только активные модулей и данных.

На стадии сравнения с образцами активных правил и действующих данных определяются модули, называемых означенными, для которых удовлетворяются условия выполнения. Это множество модулей называется конфликтным, так как только некоторые означенные модули могут быть задействованы в текущей ситуации. На стадии разрешения конфликтов принимается решение, какие из означенных модулей будут выполняться в действующем цикле.

На стадии выполнения запускаются модули, выбранные на предыдущей стадии. В результате выполнения модифицируются элементы и структуры данных рабочей памяти, выдаются необходимые рекомендации или решения пользователю, возможны изменения в самой БЗ и т. д.

Другими общими стратегиями рассуждения являются стратегии, реализующие поиск от целей или от данных. Стратегия, реализующая поиск от целей (или поиск, направляемый целью), производит поиск в обратном направлении - от искомого конечного состояния к начальным условиям. При обратной цепочке рассуждений выбирают правила, которые могут привести к поставленной цели, и стремятся удовлетворить правые части этих правил — следствия. В следствиях обычно представлены переменные, с которыми работают в процессе решения. Для определения значений этих переменных находят значения исходных посылок соответствующих им правил, т. е. цель (следствие) испытывается на истинность путем рекурсивной проверки на истинность ее подцелей (посылок). При истинности значений этих подцелей снова переходят к следствиям уже предыдущих правил для выстраивания последовательности причинных связей, которая приводит к поставленной итоговой цели.

Процесс заканчивается либо когда определяется, что значения всех подцелей построенной цепочки истинны, либо когда БЗ исследована и результаты отсутствуют.

Система объяснений (СОб) функционально предназначена для формирования ответов на вопросы пользователя относительно поведения интеллектуальной системы в процессе получения ею заключения или решения. Способность объяснять свои действия должна быть главным отличительных свойств СИИ, что повышает доверие пользователя к системе, к представляемым ею рекомендациям и решениям. Кроме того, СОб возможно использовать в процессе модификации и развития интеллектуальной системы, выявлении противоречивых знаний, а также при обучении менее подготовленных пользователей.

Системы искусственного интеллекта различных типов, ориентированные на разные проблемные области, должны иметь специфичные для них СОб (некоторые типы ИС могут вообще не иметь СОб). На практике все СОб реализуются на одних и тех же принципах в основном двумя способами:

–фиксацией событий и состояний с помощью заготовленных текстов на естественном языке;

–трассировкой рассуждений, обратным развертыванием дерева целей с указанием подцелей.

При реализации каждого из этих способов предварительно выделяются ситуации, факты и узлы перехода в новые состояния, требующие объяснений. Им ставится в соответствие некоторый текст объяснения.

При способе фиксации событий объяснения составляются из кратких текстов на естественном языке, которые хранятся вместе с правилами и фактами. Эти тексты предварительно помещаются в программу и инициируются в том случае, когда задан вопрос по соответствующей ситуации и необходимо их представление. Несмотря на некоторые преимущества, связанные с возможно-

стью, формирования удобных и простых для восприятия объяснений, этот способ имеет два важных ограничения, препятствующих широкому применению:

−объяснения должны исправляться каждый раз, когда меняется БЗ или соответствующие эвристики;

−объяснение может быть адаптировано к индивидуальному пользователю только с большим трудом. Кроме того, очень часто пользователя интересует именно ход рассуждения, цепочка логических выводов, приведших к заключению.

Способ трассировки рассуждений при объяснении предусматривает пересечение дерева целей для ответа на вопросы. Основываясь на дереве целей, СОб может объяснять, как было получено заключение. Это достигается путем прохождения подцелей, которые были удовлетворены при движении к цели. Если требуется более детальное объяснение, то СОб может повторить каждое из задействованных правил, представив их в краткой формулировке на естественном языке.

Система объяснения отвечает преимущественно на два типа вопросов: «Почему?» и «Как?» Оба вопроса должны интерпретироваться на различных уровнях, которые образуются при обосновании поведения программы исходя из действующего уровня, приоритета и компетентности пользователя.

Механизм обхода пространства состояний обычно реализуется в виде графа И/ИЛИ, и формирование объяснений связывается с обработкой вершин этого графа. При ответе на вопрос «Почему?» вероятнее движение вверх по графу состояний к ближайшей подцели, которая объясняет причину достижения текущей подцели. При ответе на вопрос «Как?» возможно движение вниз по графу с объяснением способа достижения текущей подцели. Поскольку дерево целей является И/ИЛИ графом, то при движении вниз обычно образуется несколько подцелей.

СИИ моделируют в определенной мере человеческие рассуждения. В них каждый из сделанных по определенной программе выводов должен соответствовать основным этапам, которые бы сделал эксперт. При этом он может проверить программу и при необходимости подкорректировать ее. По аналогии с черным ящиком СИИ можно рассматривать как прозрачный ящик: будучи прозрачным, они могут дать описание способа получения результатов.

Практические СИИ функционируют обычно в интерактивном режиме с пользователями, поэтому они должны обладать дружественным интерфейсом, позволяющим человеку взаимодействовать с компонентами. Подсистема интеллектуального интерфейса управляется программным обеспечением, называемым управляющая система интеллектуального интерфейса. Эта управляющая система состоит из нескольких программ, перечисленных ниже:

- обеспечение графического пользовательского интерфейса;
- организация взаимодействия пользователя с различными входными устройствами;
- представление данных с различными форматами и на разные входные устройства;
- представление пользователю помощи, подсказок, советов, диагностического режима работы или другой гибкой поддержки;
- обеспечение взаимодействия с БД и базой моделей;
- хранение входных и выходных данных;
- обеспечение цветной графики, трехмерной графики и плоттинга данных;
- окна, позволяющие отображать множество функций одновременно;
- поддержка взаимодействия между пользователями и разработчиками системы;
- обеспечение обучения на примерах;

- обеспечение гибкости и адаптивности, что позволяет интеллектуальной СПР вмещать различные задачи и технологии;

- взаимодействовать во многих различных стилях диалога и др.

Блок планирования планирует процесс вывода в зависимости от конкретной ситуации.

Монитор решателя – программа, управляющая всеми блоками решателя.

Блок рабочего поля памяти отражает реальную ситуацию использования памяти компьютера при решении интеллектуальных задач. В нее блоки индуктивного и дедуктивного вывода вызывают из БЗ необходимые знания, чтобы не искажать БЗ различными преобразованиями, нужными для решения задач.

При проектировании архитектуры СИИ много усилий и времени затрачиваются на разработку БЗ: накопление знаний, создание модели представления знаний, их структурирование, заполнение БЗ и дальнейшее поддержание ее в актуальном состоянии. Перед проектированием БЗ необходимо осмыслить и разрешить ряд вопросов, непосредственно связанных с процессом создания БЗ и СИИ в целом.

–изучение проблемной области (объекта, задач, целей), т. е. «что представлять в БЗ» и «для чего представлять»;

–определение понятия «знание» в контексте исследуемой проблемной области и их типы;

–выявление источников знаний, и работа с ними;

–оценка на основе исследования проблемной области и характера знаний пространства поиска решений с целью выбора способа структуризации знаний и метода поиска решений (механизма вывода);

–определение способа структуризации и представления знаний, т. е. того, «как представлять знания»;

– определение структуры БЗ.

Разработка БЗ, выбор ее структуры, способ представления знаний, работа с экспертами являются важными задачами при проектировании СИИ.

Метазнания определяются как знания интеллектуальной системы о себе, знания о своей работе и своей структуре. Они позволяют осознавать, исследовать и анализировать линии рассуждения самой интеллектуальной системы при решении задачи.

Кроме того, на верхнем уровне БЗ приведены метазнания, необходимые для выработки рациональной стратегии поиска. Однако вывод решения либо генерация новых правил и знаний осуществляется здесь с помощью блока вывода, который взаимодействует с метауровнем БЗ при интерпретации правил и данных БЗ.

Решение задачи и работа с правилами и данными осуществляются в специальном блоке - рабочей области. В рабочей области представляются описания запроса - или решаемой задачи, данные и правила из БЗ, процедуры или стратегия механизма вывода.

Вид интерфейса определяет, как информация введена и отображена и включает следующие его виды: взаимодействие на основе меню, командный язык, вопросно-ответный, формирование взаимодействия, обработка естественного языка и графический пользовательский интерфейс.

Взаимодействие на основе меню. Пользователь выбирает позицию или пункт из списка возможных выборов (меню) для того, чтобы функция была выполнена. Меню появляются в логическом порядке, начиная с главного меню и продвигаясь к локальным меню.

Пункты меню могут включать команды, которые появляются в отдельных локальных меню или в меню с не командными пунктами. Командный язык. Пользователь вводит команды. Многие команды включают комбинации глагол-существительное. Некоторые команды могут исполняться с функциональными

ключами. Другим способом упрощения команд является использование макросов. Команды могут также вводиться голосом.

Вопросно-ответный вид интерфейса начинается с вопросов компьютера пользователю. Пользователь отвечает на вопросы фразой или предложением (или выбором пункта меню). Компьютер может подсказывать пользователю для прояснения или дополнительного ввода информации. Формирование взаимодействия. Пользователь вводит данные или команды в обозначенные формы (поля). Заголовки формы (или отчета, или таблицы) служат подсказками для входа. Компьютер может представлять какой-то выход как результат, и пользователь может быть спрошен о продолжении интерактивного процесса.

Естественный язык. Взаимодействие человек – компьютер, которое подобно диалогу человека с человеком называется естественным языком. Сегодня диалог на естественном языке выполняется главным образом посредством клавиатуры. Такой диалог будет проводиться в будущем с использованием голоса для ввода и вывода информации. Главным ограничением использования естественного языка является по существу неспособность компьютера понимать естественный язык. Однако, достижения ИИ все больше повышают уровень диалога на естественном языке.

Графический пользовательский интерфейс. В графическом пользовательском интерфейсе объекты обычно представляются как пиктограммы (или символы) и пользователь непосредственно ими манипулирует. Новейшие операционные системы компьютеров и их приложения исключительно основаны на графике.

Контрольные вопросы

1. Какие основные подходы к проектированию СИИ существуют?
2. Какие блоки составляют основу СИИ?

3. Для чего предназначен блок индуктивного вывода в архитектуре СИИ?
4. Поясните функции блока метазнаний.
5. Чем рецепторы отличаются от эффекторов?

Глава 11. Экспертные системы

В начале 1980-х гг. сформировалось направление разработки СИИ, получившее название «экспертные системы» (ЭС), в его основе – разработка программ для хранения знаний предметных областей и получение результатов, не уступающих по качеству и эффективности решениям, получаемым экспертом. ЭС предназначены для решения практических задач, возникающих в слабо структурированной и трудно формализуемой предметной области. Технология экспертных систем расширяет круг практически задач, решаемых на компьютерах и эриносящих экономический эффект. Первая экспертная система MYCIN была предназначена для медицинских целей и разработана начале 1970-х годов в Стэнфордском университете. Там же была разработана и экспертная система PROSPECTOR, созданная для содействия поиску коммерчески оправданных месторождений полезных ископаемых.

Экспертная система (ЭС) – это сложный программный комплекс, аккумулирующий знания специалистов в конкретных предметных областях и использующий эти знании для консультаций менее квалифицированных пользователей.

Экспертные системы отличаются от систем обработки данных следующими основными чертами:

- используют символьный (а не числовой) способ представления;
- используются символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма);

- применяются для решения таких практических задач, для решения которых нужны экспертные знания;

- формируют пользователю решение, которое по качеству и эффективности не уступает решению эксперта-человека;

- формируемые решения могут быть объяснены пользователю, что обеспечивается способностью ЭС рассуждать о своих знаниях и умозаключениях;

- экспертные системы пополняют свои знания в ходе взаимодействия с экспертом, а также в процессе самообучения (т.н. машинное обучение);

Предметная область – это совокупность реальных или абстрактных объектов, связей и отношений между этими объектами, а также процедур преобразования этих объектов для решения возникающих задач. Традиционно знания существуют в двух видах – коллективные знания, которыми обладают большинство людей, и личные знания, которыми обладают специалисты (эксперты).

Если большая часть знаний в предметной области представлена в виде коллективного опыта (например, высшая математика), и эта предметная область не нуждается в экспертных системах (рис. 11.1)

Большая часть знаний предметной области является личным опытом специалистов (экспертов), и если эти знания по каким-либо причинам слабо структурированы, то такая предметная область требует работы когнитолога для формирования модели знаний. На рис.39 представлены соотношения личных и коллективных знаний, которые могут в различной степени пересекаться и объединяться, в зависимости от вида предметной области.

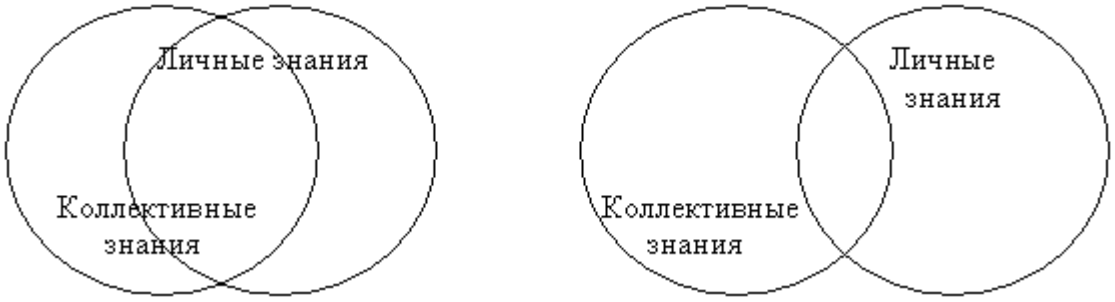


Рис.11.1 - соотношение коллективных и личных знаний в предметной области

ЭС используются для решения различных задач (интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление) в разнообразных проблемных областях. Вместе с тем, степень охвата решаемых задач отличается для различных предметных областей. Поэтому применение ЭС реально только для узких предметных областей и решения конкретной задачи в предметной области.

Класс "экспертные системы" сегодня объединяет множество различных программных комплексов, классифицируемых по различным признакам (рис. 11.2).

- **По задаче**

Интерпретация данных – традиционная задача для экспертных систем. Под интерпретацией понимается определение смысла данных, результаты которого должны быть согласованны и корректны. Обычно предусматривается многовариантный анализ данных. Например, определение основных свойств объекта по результатам тестирования [10].

Рис. 11.2 - классификация экспертных систем

Диагностика. Под диагностикой понимается обнаружение неисправности в некоторой системе, т. е. отклонение от нормы. Такая постановка позволяет с единых теоретических позиций рассматривать неисправность оборудования в технических системах, и заболевания живых организмов, и всевозможные природные аномалии. Важной спецификой является необходимость понимания функциональной структуры диагностирующей системы. Например, диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ.

Мониторинг. Основная задача мониторинга - непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы - "пропуск" тревожной ситуации и инверсная задача "ложного" срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимость учета временного

контекста. Примером может служить система контроля аварийных датчиков на опасных промышленных производствах.

Проектирование. Проектирование состоит в подготовке спецификаций на создание "объектов" с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов чертеж, пояснительная записка и т.д. Основные проблемы здесь - получение четкого структурного описания знаний об объекте и проблема "следа". Для организации эффективного проектирования и, в еще большей степени, перепроектирования, необходимо формировать не только сами проектные решения, но и мотивы их принятия. В задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ЭС: процесс вывода решения и процесс объяснения.

Прогнозирование. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров "подгоняются" под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками. Например, предсказание погоды.

Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности.

Обучение. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом "ученике" и его характерных ошибках, затем в работе способны диагностировать слабости в знаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют

акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

- **По связи с реальным временем**

Статические ЭС разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны. Например, диагностика неисправностей в автомобиле.

Квазидинамические ЭС интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Например, лабораторные ЭС, в которых выполняются анализы технологического процесса и анализируется динамика полученных показателей по отношению к предыдущему измерению.

Д**инамические** ЭС работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступаемых данных. Например, управление гибкими производственными комплексами, мониторинга в реанимационных палатах и т.д.

- **По типу ЭВМ**

На сегодняшний день существуют:

ЭС для уникальных стратегически важных задач на суперЭВМ;

ЭС на ЭВМ средней производительности;

ЭС на символьных процессорах и рабочих станциях;

ЭС на мини- и супермини-ЭВМ;

ЭС на персональных компьютерах.

- **По степени интеграции с другими программами**

Автономные ЭС работают непосредственно в режиме консультаций с пользователем для специфически "экспертных" задач, для решения которых не требу-

ется привлекать традиционные методы обработки данных (расчеты, моделирование и т. д.).

Гибридные ЭС представляют программный комплекс, агрегирующий стандартные пакеты прикладных программ (например, математическую статистику, линейное программирование или системы управления базами данных) и средства манипулирования знаниями. Разработка таких систем на порядок более сложная, чем разработка автономной ЭС, так как необходимо согласовывать разные методологии и это порождает комплекс теоретических и практических трудностей.

Обобщенная структура экспертной системы

Обобщенная структура экспертной системы представлена на рисунке 11.3.

Интерфейс пользователя (диалоговый процессор) - комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, получения результатов и «объяснения» решения.

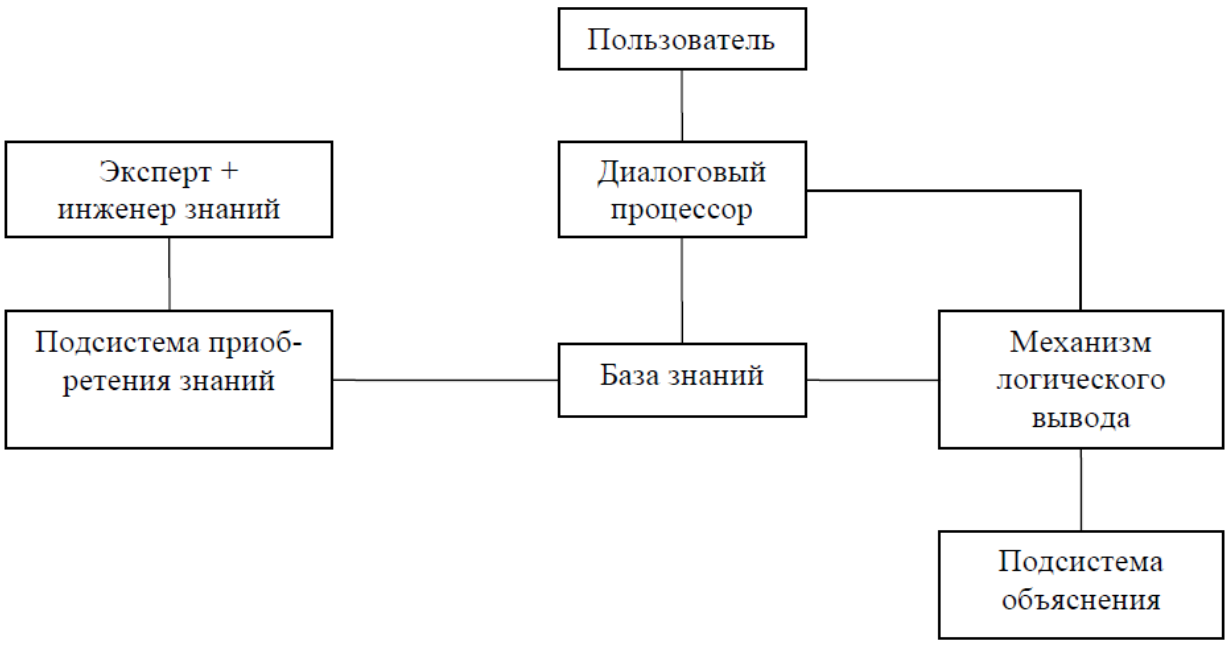


Рис.11.3 - обобщенная структура ЭС

База знаний(БЗ) - ядро ЭС, совокупность формализованных знаний предметной области на машинном носителе в форме, понятной эксперту и инженеру по знаниям. Параллельно такому представлению существует БЗ во внутреннем "машинном" представлении-

Решатель (машина логического вывода, интерпретатор) - программа, моделирующая ход рассуждений эксперта на основании формализованных знаний, имеющихся в БЗ и исходных данных (фактах), получаемых от пользователя.

Подсистема объяснения - программа, протоколирующая работу решателя в виде «цепочки логических выводов». Она позволяет пользователю получить ответы на вопросы; "Как был получен совет?" и "Почему система приняла такое решение?" Ответ на вопрос "как" - это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, т.е. всех шагов цепи умозаключений. Ответ на вопрос "почему"- ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад.

Редактор БЗ - программа, представляющая инженеру по знаниям возможность создавать и пополнять БЗ в диалоговом режиме. Осуществляет ввод формализованных знаний, например, правил продукционной модели представления знаний. Включает систему вложенных меню, шаблонов языка представления знаний, подсказок ("help" - режим) и других сервисных средств, облегчающих работу с базой знаний.

Статическая экспертная система используются в тех случаях, когда можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими, и работали в двух режимах:

в режиме приобретения знаний;

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 (редактор БЗ), наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

В режиме консультации общение с ЭС осуществляет пользователь, которого интересует результат и способ его получения.

В зависимости от назначения ЭС пользователь может не быть специалистом в данной проблемной области (ему нужен результат, который он не умеет получить сам), или быть специалистом (может сам получить результат, но нужно ускорить процесс получения результата, проконсультироваться, либо возложить на ЭС рутинную работу).

В режиме консультации исходные данные (факты) о задаче от пользователя поступают через интерфейс в решатель. Решатель на основе входных данных, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения, которое выдается из системы объяснений.

Динамическая экспертная система используются в условиях изменения окружающего мира, происходящие за время решения задачи. Пользователь в режиме консультации не вводит исходные данные, а лишь отслеживает результат рабо-

ты. Исходные данные непрерывно поступают с датчиков. Такие системы используют, например, для контроля технологических процессов, контроля работы опасных систем (атомных станций).

Кроме основных компонентов существуют дополнительные, позволяющие создавать интегрированные системы в соответствии с современной технологией использования ЭС. Наибольшее распространение в ЭС получила продукционная модель представления знаний, состоящая из набора правил.

Машина вывода (интерпретатор) – это программа, управляющая перебором правил в базе знаний. Машина вывода (интерпретатор правил) выполняет две функции:

- просмотр существующих фактов из рабочей памяти и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов.

- определение порядка просмотра и применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных.

Действие компонента вывода наиболее часто основано на применении правила modus ponens:. «Если известно, что истинно утверждение А и существует правило вида «ЕСЛИ А, ТО В», тогда утверждение В также истинно».

Правила срабатывают, когда находятся 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—правила сопоставляются с имеющимися фактами, полученными от пользователя или в результате срабатывания правила..

Выбор — если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

Срабатывание — выполнение части «ТО» выбранного правила.

Действие — рабочая память подвергается изменению путем добавления в нее заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие (воздействие), то оно выполняется.

Машина вывода (Интерпретатор продукций) работает циклически. Цикл работы интерпретатора схематически представлен на рисунке 11.4.

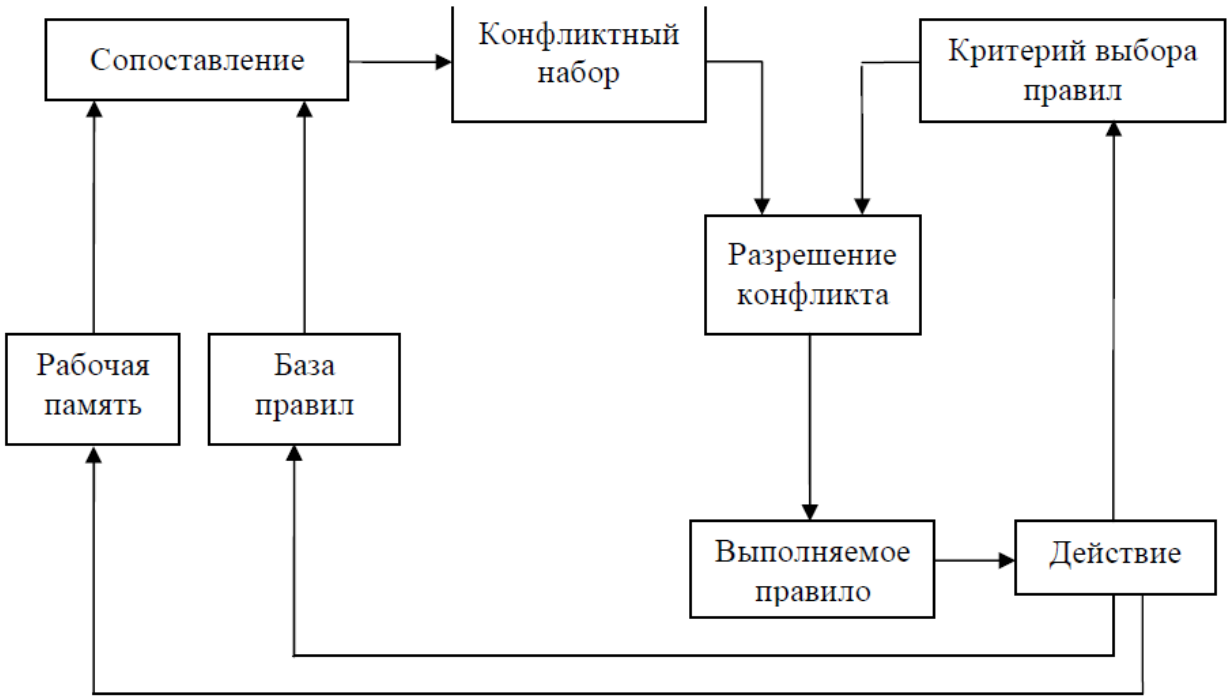


Рис. 11.3 - цикл работы машины вывода

Информация из рабочей памяти (базы данных) последовательно сопоставляется с условными частями правил из базы правил для выявления успешного сопоставления. Совокупность успешно отобранных правил составляет так называе-

мое конфликтное множество. Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он выбирает единственное выполняемое правило. После чего оно срабатывает и выполняется действие. Действие может выражаться:

- в занесении нового факта (заключения сработавшего правила) в рабочую память;

- в изменении критерия выбора конфликтующих правил,

- в воздействии на что- либо (например, отключение аварийного блока).

Затем цикл повторяется снова, до тех пор, пока не перестанет образовываться конфликтное множество. Заключение последнего сработавшего правила выдается пользователю в качестве результата решения задачи.

Работа машины вывода зависит только от состояния рабочей памяти и от состава базы знаний [11].

В каждом цикле просматриваются все правила, но в одном цикле может сработать только одно правило

Технология разработки ЭС включает следующие этапы:

- идентификация,
- концептуализация,
- формализация,
- выполнение,
- тестирование
- опытная эксплуатация.

На этапе **идентификации** определяются задачи, подлежащие решению, выявляются цели разработки, определяются эксперты и типы пользователей. Идентификация задачи заключается в составлении неформального (вербального)

описания, в котором указываются: общие характеристики задачи; подзадачи, выделяемые внутри данной задачи; ключевые понятия (объекты), их входные (выходные) данные; предположительный вид решения, а также знания, относящиеся к решаемой задаче.

Результат этапа заключается в том, что требуется выполнить и какие ресурсы необходимо задействовать (идентификация задачи, определение участников процесса проектирования и их роли, выявление ресурсов и целей). На выходе этапа формируются требования к экспертной системе.

Этап **концептуализации** представляет содержательный анализ проблемной области и решаемой задачи, выявляются используемые понятия и их взаимосвязи, определяются методы решения задачи.

Этот этап завершается созданием модели предметной области, включающей основные концепты и отношения. На этапе концептуализации определяются следующие особенности задачи:

-типы доступных данных;

-исходные и выводимые данные, подзадачи общей задачи; используемые стратегии и гипотезы;

-виды взаимосвязей между объектами, типы используемых отношений (иерархия, причина — следствие, часть — целое и т.п.);

- процессы, используемые в ходе решения;

- состав знаний, используемых при решении задачи; типы ограничений, накладываемых на процессы, используемые в ходе решения;

-состав знаний, используемых для обоснования решений.

На этапе **формализации** выбираются инструментальные средства разработки, определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, разрабатывает-

ся программная оболочка (прототип), моделируется работа системы, оценивается адекватность системы.

Выходом этапа формализации является описание того, как рассматриваемая задача может быть представлена в выбранном или разработанном формализме. Сюда относится указание способов представления знаний (фреймы, сценарии, семантические сети и т.д.) и определение способов манипулирования этими знаниями (логический вывод, аналитическая модель, статистическая модель и др.) и интерпретации знаний. В конце этапа имеем структуру системы.

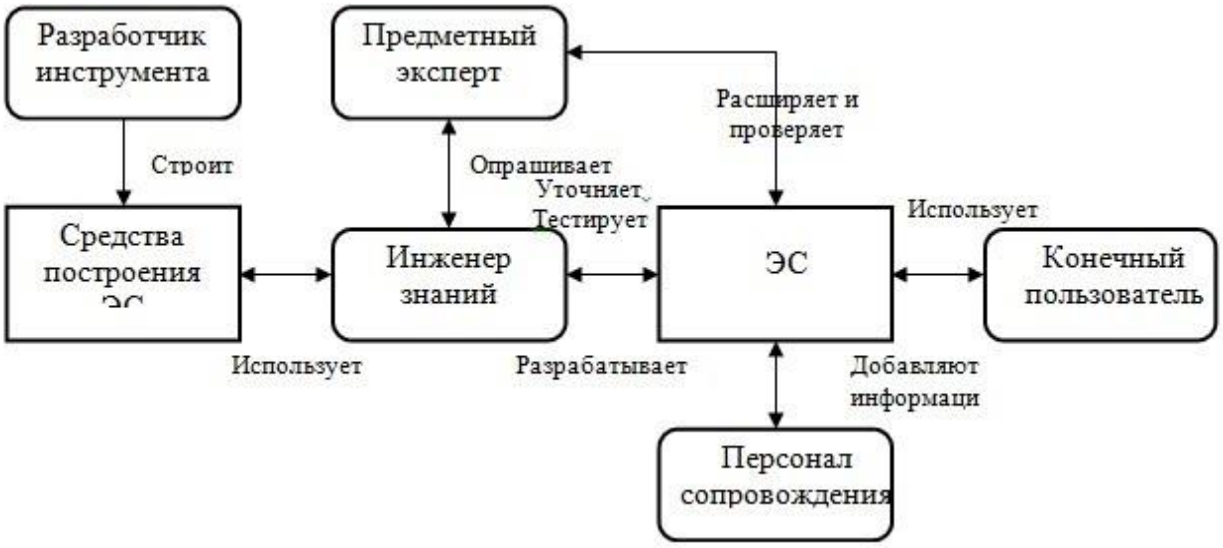
На этапе **выполнения** создаются один или нескольких прототипов ЭС, решающих требуемые задачи, базы знаний наполняются экспертом. Этот этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, формализацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач. На выходе этапа – модели программной реализация всех структурных элементов экспертной системы.

Этап **тестирования.** В ходе данного этапа производится оценка выбранного способа представления знаний в ЭС в целом. Для этого инженер по знаниям совместно с экспертом подбирает тестовые примеры, обеспечивающие проверку возможностей разработанной ЭС. При подготовке тестовых примеров следует классифицировать их по подпроблемам предметной области, выделяя стандартные случаи, определяя границы трудных ситуаций и т.п. Степень соответствия решения тестового примера полученного экспертной системой и известного решения характеризует качество разработки системы. Главным в оценке работы системы является полнота и безошибочность работы правил вывода.

На этапе **опытной эксплуатации** проверяется пригодность ЭС для конечного пользователя. Пригодность ЭС для пользователя определяется в основном удобством работы с ней и ее полезностью. Под полезностью ЭС понимается ее способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неудач в работе, а также удовлетворять указанные потребности пользователя (решать поставленные задачи). Удобство работы с ЭС определяется естественностью взаимодействия с ней, гибкостью ЭС (способность системы настраиваться на различных пользователей, а также учетом в изменении квалификации пользователя) и устойчивости системы к ошибкам (способность не выходить из строя при ошибочных действиях неопытного пользователях).

Коллектив разработчиков ЭС

Под коллективом разработчиков понимается группа специалистов, ответственных за создание ЭС. В разработке ЭС участвуют представители следующих специальностей (рис. 11.5).



. Рис.11.5 - взаимодействие коллектива разработчиков при построении экспертной системы

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

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний. Руководит коллективом.

Программист(ы) осуществляет программную реализацию ЭС, сопрягает все основные компоненты ЭС.

Пользователь осуществляет проверку удобства работы с ЭС на заключительных этапах разработки.

Минимальный состав коллектива включает четыре человека; реально же он разрастается до 8-10 человек. Численное увеличение коллектива разработчиков происходит по следующим причинам: необходимость учета мнения нескольких пользователей, помощи нескольких экспертов; потребность как в проблемных, так и системных программистах.

Для обеспечения эффективности работы любой творческой группы, в том числе и группы разработчиков ЭС, необходимо возникновение атмосферы взаимопонимания и доверия, которое, в свою очередь, обусловлено психологической совместимостью членов группы.

Прототипы и жизненный цикл экспертной системы

По степени готовности к использованию и распространению различают четыре прототипа экспертных систем:

1) демонстрационный. Предназначен для демонстрации возможностей будущей экспертной системы, основных архитектурных решений, пользовательского интерфейса, для уточнения требований к пользовательскому интерфейсу и функциям, выполняемым экспертной системой, содержит демонстрационную далеко неполную базу знаний;

2) исследовательский. Предназначен для исследования направлений дальнейшего совершенствования экспертной системы и для пополнения базы знаний, может использоваться для решения реальных задач в ограниченных пределах;

3) промышленный. Предназначен для использования, как правило, в организации, где был разработан, в нем возможны ограничения, условности, специализация, свойственные для данной организации;

4) коммерческий. Предназначен для широкого распространения, обладает гибкостью, удобством в эксплуатации, адаптируемостью к конкретным задачам и требованиям пользователя.

Жизненный цикл экспертной системы состоит из этапов разработки и сопровождения. На этапе разработки создается программное обеспечение и база знаний экспертной системы, на этапе сопровождения происходит исправление выявленных ошибок и пополнение базы знаний без участия разработчиков (если последнее допускается архитектурой экспертной системы).

ЭС, достигшая стадии промышленной системы, обеспечивает высокое качество решений всех задач при минимуме времени и памяти. Обычно процесс преобразования действующего прототипа в промышленную систему состоит в расширении базы знаний (до 150 исполняемых утверждений) и ее

тщательной отладке. Доведение ЭС от начала разработки до стадии промышленной системы на развитом ИС требует примерно 12 – 18 месяцев.

Обобщение задач, решаемых ЭС на стадии промышленной системы, позволяет перейти к стадии коммерческой системы, т.е. к системе, пригодной не только для собственного использования, но и для продажи различным потребителям. Доведение системы до коммерческой стадии требует примерно 1,5–2 года. В таблице 11.1 приведены данные о этапах и сроках разработках ЭС, их стоимости.

Табл. 11.1 – этапы разработки экспертных систем

Этап разработки	Характер прототипа	Количество правил	Срок разработки	Стоимость
Идентификация	Демонстрацион-ный	50 - 100	1 - 2 мес.	
Концептуализация Формализация	Исследователь-ский	200 - 500	3 - 6 мес.	25 - 50т.$
Реализация	Действующий	500 - 1000	6 - 12 мес.	
Тестирование	Промышленный	1000 - 1500	1 - 1,5 года	300т.$
Опытная эксплуатация	Коммерческий	1500 - 3000	1,5 - 3 года	2 - 5 млн.$

Контрольные вопросы

1. Каково основное назначение экспертных систем?
2. Назовите основные элементы экспертных систем?
3. Чем определяется жизненный цикл экспертной системы?
4. Какие основные процедуры включает цикл работы машины вывода?
5. Зачем в экспертной системе нужна подсистема объяснений?

Глава 12. Прикладные системы искусственного интеллекта

Интеллектуальные естественно-языковые системы (ИЕЯС) – это совокупность программных и аппаратных средств, обеспечивающих общение пользователя с СИИ на ограниченном рамками предметной области естественном языке. В состав ИЕЯС входят словари, отражающие словарный состав и лексику языка,

а также лингвистический процессор, осуществляющий анализ текстов (морфологический, синтаксический, семантический и прагматический) и синтез ответов пользователю.

ИЕЯС должно удовлетворять двум основным требованиям:

- обеспечивать вход и выход системы на естественном языке;

- обеспечивать обработку входных данных и генерацию выходных, основываясь на знании относительно синтаксических, семантических и прагматических аспектов естественного языка.

Понимание естественного языка зависит от знания предметной области и требует знаний о целях говорящего и о контексте. Необходимо также учитывать недосказанность или иносказательность.

Естественно-языковый интерфейс содержит две взаимодействующие подсистемы.

Первая – для обеспечения взаимодействия интерфейса с пользователем. При этом должен быть создан алгоритм понимания смысла того, что пользователь может сказать, а система - должна понять.

Вторая – для преобразования смысла сказанной фразы в какое-либо внутреннее представление (модель), понимание смысла и трансляции этого смысла в запросы SQL на языке, понятном для базы данных. Возврат информации осуществляется в обратном порядке.

База данных на запрос отвечает множеством записей из базы данных. Результат транслируется во внутреннее представление, затем во фразы естественного языка и выдается пользователю как ответ пользователем на естественном языке.

Существуют пользовательские интерфейсы, называемые традиционными:

- интерфейсы с формальным языком запросов;
- интерфейсы, основанные на заполнении форм запросов.

В интерфейсах с формальным языком запросов пользователь для того, чтобы правильно задать запрос, должен знать синтаксис языка запросов (например, SQL) и устройство структурированного источника данных (например, реляционную схему базы данных). При работе с этим типом интерфейсов пользователь должен обладать достаточно высокой квалификацией. ИЕЯС-интерфейс должен обладать большей гибкостью, чтобы один и тот же запрос формулировать различными способами.

Интерфейсы, основанные на заполнении форм запросов, являются более дружественными, по сравнению с формальными языками. Пользователь сразу видит набор критериев и параметров поиска, а иногда и список возможных значений полей формы, это сводит к минимуму ошибки при вводе запроса.

ИЕЯС должны удовлетворять следующим критериям качества:

- надежность, под которой понимается способность ЕЯ-интерфейса правильно понимать намерения пользователя по получению информации из источника, при условии, что пользователь корректно выразил потребности в виде ЕЯ-запроса. Надежность отражает правильность принципов, лежащих в методе ЕЯ-анализа, а также правильность (корректность) построения ЕЯ-интерфейса к конкретному СИД;

- полнота. Эта характеристика, тесно связанная с гибкостью интерфейса. Любой ЕЯ-интерфейс имеет некоторое пространство правильно понимаемых запросов. Чем больше это пространство, тем большей полнотой обладает ЕЯ-интерфейс;

- дружественность интерфейса, определяемую как меру того, насколько ЕЯ-интерфейс удобен в работе, насколько корректно он может сообщать о проблемах понимания, может ли он помогать в переформулировке неберущихся запросов и т. д.;

- портируемость, которая может быть представлена переводом:

1. На другой язык,
2. На другую СУБД,
3. На другие платформы и языки программирования.

Структура ИЕЯС (рис. 12.1) содержит лексический анализатор ЕЯ как компонент, реализующий тот или иной метод анализа естественного языка, и от принципов построения которого, зависит архитектура системы и основные характеристики интерфейсов на основе данного компонента.

Лингвистическое ядро является универсальным элементом, словарь содержит универсальную лексику, используемую во многих ЕЯ-интерфейсах, модели предметной области (МПО) содержат шаблоны, общие для нескольких предметных областей и т.д. Часто ЕЯ-интерфейс собирается из готовых компонентов, которые настраиваются под конкретную базу данных.

Для процесса анализа сначала проводится лексический анализ (преданализ), преобразующий входной текст как последовательность символов, в цепочку лексем, поступающей на вход анализатора. Затем работа анализатора заключается в построении внутреннего представления входного ЕЯ-текста либо запроса, обычно в виде синтаксического дерева, семантической сети, фреймовой структуры и т. д.

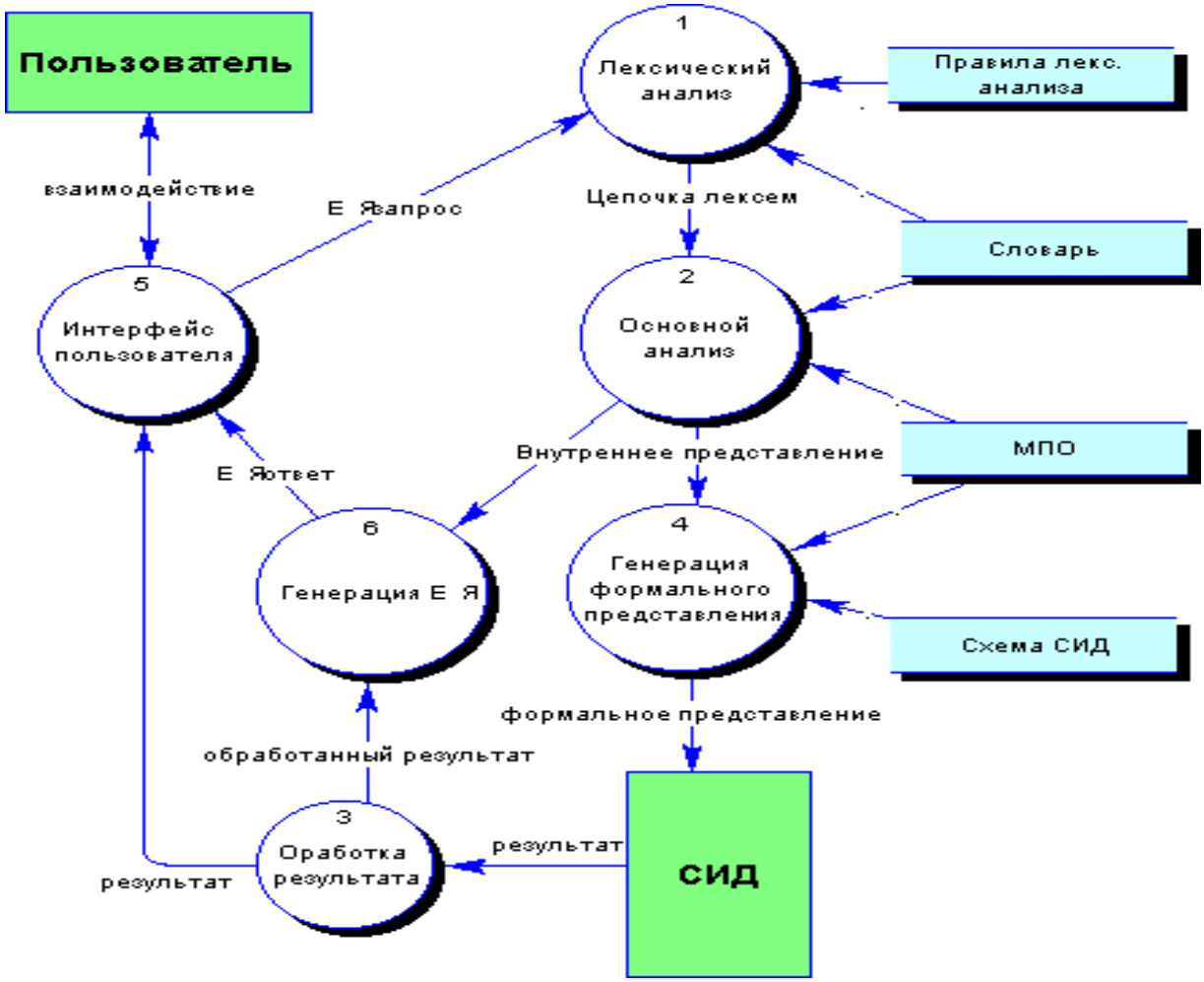


Рис. 12.1 - Основные составляющие ИЕЯС и взаимосвязи

Необходимым компонентом работы анализатора является словарь, содержащий слова и фразы, обычно с привязкой к ним определенной информации, связанной с семантикой, морфологией и т.д., в зависимости от подхода анализа ЕЯ. Важным компонентом многих систем является модель предметной области, структура которой варьируется в очень больших пределах от системы к системе.

Для построения запроса на формальном языке источника данных используется модель источника данных, отражающая основную структуру СИД, ее части, существенные для данного ЕЯИ.

Для перевода запроса из внутреннего представления системы в формальный язык источника данных предназначен процесс генерации формального запроса. Некоторые системы имеют также модуль синтеза ЕЯ, применяемый для генерации естественно-языкового представления запроса, например, для верификации понимания запроса системой, а также для генерации уточняющих вопросов.

Модель предметной области является одной из важнейших составляющих понимания ЕЯ. Пользователь, обращаясь с запросом к информационной системе, как правило, имеет некоторое представление о предметной области, в которой работает информационная система. Модель предметной области в некоторых системах дополняется базой знаний со средствами вывода новых знаний.

Способы понимания запросов естественного языка:

- синтаксически-ориентированные. В результате парсинга (синтаксического анализа) запроса дерево синтаксического разбора непосредственно отображается в выражение на языке запросов к базе данных. Синтаксически-ориентированные системы используют грамматику, описывающую возможные синтаксические структуры пользовательских запросов. Отображение дерева в выражение запроса производится с помощью правил и целиком основывается на синтаксической информации дерева разбора;

- семантически-ориентированные. Предложены А. С. Нариньяни, обеспечивают надежность понимания ЕЯ в ограниченных регистрах естественного языка, используют закономерности выражения семантики в естественном языке. Суть способа заключается в оперировании компонентами, являющимися концептами, имеющими смысл в достаточно узком регистре ЕЯ. Компоненты образуют семиотическую (знаковую) систему, а процесс анализа представляет собой движение снизу вверх, при котором на основе компонентов входной цепочки обра-

зуются вторичные компоненты, имеющие, например, семантику предикатов, логических операций и т. д. Поскольку система компонентов мало зависит от конкретного естественного языка, данный способ позволяет построение достаточно универсальных лингвистических схем.

Интеллектуальные мультиагентные системы (МАС). Интеллектуальные мультиагентные системы – новое перспективное направление развития систем искусственного интеллекта, сформированное на основе распределённых информационных систем, сетевых технологий и параллельных вычислений. В мультиагентных технологиях заложен принцип автономности отдельных частей программы (агентов), совместно функционирующих в распределённой системе, где одновременно протекает множество взаимосвязанных процессов. Под агентом подразумевают автономный объект (компьютерную программу), обладающий активным мотивированным поведением и способный к взаимодействию с другими объектами в динамических виртуальных средах. Каждый агент может принимать сообщения, интерпретировать их содержание и формировать новые сообщения, которые либо передаются на «доску объявлений», либо направляются другим агентам [8].

Два класса задач МАС:

- задачи, где усилия разных агентов направлены на решение общей проблемы, и необходимо обеспечение эффективного способа кооперации их деятельности.
- задачи, где агенты самостоятельно достигают своих локальные целей, используя общие, как правило, ограниченные ресурсы.

В мультиагентных системах (МАС) множество автономных агентов действуют в интересах различных пользователей и взаимодействуют между собой в процессе решения определённых задач. Агентами названы искусственные модули, способные воспринимать и интерпретировать сигналы, поступающие из внешней среды, и формировать ответные сигналы. В роли таких модулей сначала вы-

ступали конечные автоматы, не имеющие априорных знаний о свойствах окружающей среды и о наличии в ней других существ. Единственным знанием, которым они обладали, была цель их деятельности и способность оценивать поступающие сигналы относительно достижения этой цели. Даже такие простые структуры, как конечные автоматы, демонстрируют хорошие способности к адаптации в стационарных вероятностных средах. Одной из главных характеристик агентов–автоматов была рациональность, которая определялась как сумма положительных откликов среды, накопленных агентом за некоторый период его существования.

В дальнейших исследованиях структуры усложнялись. Сначала появились вероятностные автоматы с переменной структурой, адаптирующейся к характеристикам среды, затем появились агенты, способные изменять свои реакции на основании предыстории и анализа состояния окружения. Серьёзным шагом в развитии мультиагентных технологий стала реализация способности агентов к рассуждениям. Простейшие модели взаимодействия агентов предусматривали их общение через среду. При этом на каждом шаге функционирования агенты совершают выбор возможных для них действий. Множество действий всех агентов обусловливает распределение откликов среды для всех участников, которые могут его использовать либо не использовать при формировании своих ответных реакций.

Дальнейший шаг к пониманию агентов сделан при переходе к коллективной работе в распределённых системах искусственного интеллекта. Этот шаг стал началом бурного развития мультиагентных технологий. Предложены разнообразные модели агентов и способы их реализации, решены практические задачи и созданы инструментальные средства для разработки мультиагентных систем, сформулированы различные принципы взаимодействия агентов.

Интеллектуальным агентам присущи следующие основные свойства [8]:

- автономность – это независимость агента от окружающей среды, т.е. наличие свободы, обусловливающей собственное поведение, обеспеченное необходимыми ресурсами;
- активность – способность к организации и реализации действий;
- общительность – взаимодействие и коммуникация с другими агентами;
- реактивность – адекватное восприятие состояния среды и реакция на его изменение;
- целенаправленность, предполагающая наличие собственных источников мотивации;
- наличие базовых знаний о себе, о других агентах и об окружающей среде;
- убеждения – переменная часть базовых знаний, меняющихся во времени;
- желания – стремление к определённым состояниям;
- намерения – действия, которые планируются агентом для выполнения своих обязательств и/или желаний;
- обязательства – задачи, которые выполняет один агент по просьбе и/или поручению других агентов.

Иногда к этому списку добавляются другие качества, в том числе:
- правдивость – неспособность к подмене истинной информации заведомо ложной;
- благожелательность – готовность к сотрудничеству с другими агентами в процессе решения собственных задач, что обычно предполагает отсутствие конфликтующих целей, поставленных перед агентами;
- альтруизм – приоритетность общих целей по сравнению с личными;
- мобильность – способность агента мигрировать по сети в поисках необходимой информации.

Также для классификации агентных модулей используются два основных признака:

1) степень развития внутреннего представления о внешнем мире;

2) способ поведения.

По первому признаку выделяются интеллектуальные (когнитивные, рассуждающие) и реактивные агенты. Интеллектуальные агенты обладают хорошо развитой и пополняемой символьной моделью внешнего мира благодаря наличию у них БЗ, механизмов рассуждения и анализа действий. Реактивные агенты не имеют развитого представления о внешней среде. Они не используют рассуждений и могут не иметь собственных ресурсов. Их поведение определяется целью, в соответствии с которой формируются реакции на предъявляемые ситуации (табл. 12.1).

В связи с этим реактивные агенты не имеют внутренних источников мотивации и не способны планировать свои действия (реактивность в чистом виде – это обратная связь без прогноза).

Интеллектуальная мультиагентная система представляет собой множество интеллектуальных агентов, распределённых в сети, которые мигрируют по ней в поисках релевантных данных, знаний, процедур и кооперируются для достижения поставленных перед ними целей [7].

В зависимости от концепции, принятой при разработке МАС, возможны различные варианты её архитектуры, среди которых выделяют три базовых типа:

1) архитектуры, основанные на методах работы со знаниями;

2) архитектуры, в которых используются поведенческие модели «стимул–реакция»;

3) гибридные архитектуры.

В архитектурах первого типа для представления и обработки знаний используются традиционные модели, методы и средства искусственного интеллекта, а принятие решений осуществляется на основе механизмов формальных рассуждений (логика предикатов первого порядка).

Табл. 12.1 – свойства агентов МАС

Признак	Тип агента			
	простой	смышлёный	ителлектуальный	действительно интеллектуальный
Автономность	+		+	+
Взаимодействие с другими агентами и/или пользователями	+	+	+	+
Реактивность	+	+	+	+
Способность использования абстракции		+	+	+
Адаптивное поведение		+	+	+
Обучение на основе взаимодействия с окружением			+	+
Толерантность к ошибкам и/или неверным входным сигналам			+	
Функционирование в режиме реального времени			+	
Взаимодействие на естественном языке			+	

Основной недостаток архитектур первого типа – сложность или принципиальная невозможность построения достаточно полных баз знаний, являющихся необходимой частью создаваемых систем. Интеллектуальный агент может иметь архитектуру типичной продукционной системы, способную воспринимать ин-

формацию из внешней среды и осуществлять те или иные действия в результате обработки этой информации.

Главные отличия агентной программы от обычной продукционной ЭС связаны с наличием механизма формирования целей и модуля коммуникации, который обеспечивает взаимодействие с другими агентами. Агент с такой архитектурой способен к рассуждениям, но не способен к обучению.

В архитектурах второго типа, называемых реактивными, модели поведения агентов представлены либо наборами правил, позволяющих выбрать действие, соответствующее ситуации, либо конечными автоматами, либо другими средствами, обеспечивающими формирование адекватных реакций агента на возникающие в системе стимулы. Системы этого типа, как правило, имеют высокую степень специализации и строгие ограничения на сложность решаемых задач.

Гибридные интеллектуальные мультиагентные системы, наиболее перспективные, используют возможности интеллектуальных и реактивных архитектур. Агент с подобной архитектурой обладает способностью к рассуждениям и к реактивному поведению. Его БЗ содержит три уровня: 1) знания предметной области; 2) знания о взаимодействии, которые позволяют принимать решения в условиях неопределённости; 3) управляющие знания.

Интеллектуальное поведение агента обеспечивается способностью принимать решения, а реактивное – системой контроля за содержимым рабочей памяти, функционирующей по принципу глобальной доски объявлений. Агент взаимодействует с пользователем, используя человеко-машинный интерфейс. В общем случае гибридные архитектуры являются многоуровневыми и отличаются друг от друга структурой и содержанием уровней, соответствующие различным уровням управления, абстракции либо отдельным функциональным свойствам агента.

Главная черта ИМАС, отличающая их от других интеллектуальных систем, – взаимодействие между агентами, означающее установление многосторонних динамических отношений между субъектами. Оно является не только следствием деятельности агентов, но и необходимым условием формирования виртуальных сообществ. Взаимодействие – не просто связь между сосуществующими агентами, но и предпосылка для взаимных превращений самих агентов и отношений между ними.

Системы поддержки принятия решений – это системы искусственного интеллекта, предназначенные для выбора лучших вариантов, и лицо, принимающее решение, обеспечивается не только информационной, но в первую очередь технологической поддержкой процедуры, вплоть до выбора лучшего решения.

Понятие о поддержке в принятии решений сформулировали П. Кин и Ч. Стэйбел. Ранние определения систем поддержки принятия решений (в начале 70-х годов прошлого века) отражали следующие три свойства систем:

- возможность оперировать с неструктурированными или слабоструктурированными задачами, в отличие от задач, с которыми имеет дело исследование операций;
- интерактивные автоматизированные (то есть реализованные на базе компьютера) системы;
- разделение данных и моделей.

В современном представлении идеальная система поддержки принятия решений:

- оперирует со слабоструктурированными решениями;
- предназначена для лица, принимающего решения, различного уровня;
- может быть адаптирована для группового и индивидуального использования;

- поддерживает как взаимозависимые, так и последовательные решения;
- поддерживает 3 фазы процесса решения: интеллектуальную часть, проектирование и выбор;
- поддерживает разнообразные стили и методы решения, что может быть полезно при решении задачи группой лиц, принимающих решения;
- является гибкой и адаптируется к изменениям как организации, так и ее окружения;
- проста в использовании и модификации;
- улучшает эффективность процесса принятия решений;
- позволяет человеку управлять процессом принятия решений с помощью компьютера, а не наоборот;
- поддерживает эволюционное использование и легко адаптируется к изменяющимся требованиям;
- может быть легко построена, если возможно сформулировать логику конструкции системы поддержки принятия решений;
- поддерживает моделирование;
- позволяет использовать знания.

Систему поддержки принятия решений можно представить в виде процессов (рис. 12.3).

Система проводит сбор запрашиваемых у пользователя или внешних датчиков данных и вложенных в нее при создании данных и знаний. После этого определяет состояние, в котором находится система и решаемая задача, критерии и цели (может запрашивать и уточнять у пользователя). На основе полученных данных, которые содержатся в памяти, и имеющейся модели системы или задачи с учетом сформированных критериев и целей генерируется множество решений, которые проверяются на модели, и выбирается лучшее. После реали-

зации решения производится оценка результатов: если она неудовлетворительная, то процессы генерации и выбора повторяются с учетом новых данных.

С информационно-аналитической точки зрения, задачей системы поддержки принятия решений является агрегирование (сжатие) многокритериальной информации об анализируемых объектах до объема и формы представления, воспринимаемых лицом, принимающим решение.

С программно-технологической точки зрения, варианты решений являются для системы принятия решений просто анализируемыми объектами, характеризуемыми наборами количественных и качественных характеристик.

Классификацию СППР целесообразно разделить по нескольким уровням.

На уровне пользователя выделяют следующие СППР:

- пассивные;
- активные;
- кооперативные.

Рис.12.3 - процессы в системах поддержки принятия решений

Пассивной системой поддержки принятия решений называется система, помогающая принять решение, но СППР не может выбрать или отклонить решение.

Активная система может сделать предложение, о выборе решения.

Кооперативная система позволяет ЛПР изменять, пополнять или улучшать решения, предлагаемые системой, посылая затем эти изменения в систему для проверки. СППР в свою очередь изменяет пополняет или улучшает эти решения и посылает их опять пользователю. Процесс продолжается до получения согласованного решения.

На концептуальном уровне выделяют следующие СППР:

- управляемые сообщениями;
- управляемые данными;
- управляемые документами;
- управляемые знаниями;
- управляемые моделями.

Система, управляемая сообщениями, поддерживает группу пользователей, работающих над выполнением общей задачи.

Системы, управляемые данными, ориентируются на доступ и манипуляции с данными.

Системы, управляемые документами, осуществляют поиск и манипулируют неструктурированной информацией, заданной в различных форматах.

Системы, управляемые знаниями, обеспечивают решение задач в виде фактов, правил, процедур.

Системы, управляемые моделями, базируются на математических моделях (статистических, финансовых, оптимизационных, имитационных).

На уровне данных, с которыми эти системы работают, условно можно выделить:

- оперативные;
- стратегические.

Оперативные системы поддержки принятия решений предназначены для немедленного реагирования на изменения текущей ситуации в управлении бизнес-процессами компании.

Стратегические системы ориентированы на анализ значительных объемов разнородной информации, собираемой из различных источников.

На уровне решаемой задачи и области применения выделяют системы поддержки принятия решений:

- первого класса;
- второго класса;
- третьего класса.

Системы поддержки принятия решений первого класса, обладающие наибольшими функциональными возможностями, предназначены для применения в органах государственного управления высшего уровня (администрация президента, министерства) и органах управления больших компаний. Системы поддержки принятия решений этого класса являются системами коллективного пользования, базы знаний которых формируются многими экспертами – специалистами в различных областях знаний.

Системы поддержки принятия решений второго класса являются системами индивидуального пользования, базы знаний которых формируются непосредственным пользователем. Они предназначены для использования государственными служащими среднего ранга, а также руководителями малых и средних фирм для решения оперативных задач управления.

Системы поддержки принятия решений третьего класса являются системами индивидуального пользования, адаптирующимися к опыту пользователя. Они предназначены для решения часто встречающихся прикладных задач си-

стемного анализа и управления (например, выбор субъекта кредитования, выбор исполнителя работы, назначение на должность и пр.). Такие системы обеспечивают получение решения текущей задачи на основе информации о результатах практического использования решений этой же задачи, принятых в прошлом.

На уровне архитектуры системы поддержки принятия решений делятся:

- на функциональные системы поддержки принятия решений;
- независимые витрины данных;
- двухуровневое хранилище данных;
- трехуровневое хранилище данных.

Они отличаются организацией серверной стороны системы поддержки принятия решений. Характерной чертой функциональной системы является то, что анализ осуществляется с использованием данных из оперативных систем. На клиентской стороне располагается пользовательский интерфейс системы поддержки принятия решений, а на серверной – источники данных для задач принятия решений.

Системы поддержки генерации решений можно разделить на эвристические и оптимизационные. Эвристические технологии стимулируют и дисциплинируют мышление (например, структурный и морфологический анализ), помогают находить варианты решений на базе известных правил, принципов и аналогов. Оптимизационные системы поддержки принятия решений основаны на методах оптимального структурного синтеза и параметрической оптимизации.

Обобщенная архитектура системы поддержки принятия решений состоит из пяти частей (рис. 12.4): источники данных (часто используется база данных), система управления данными (если источников несколько, подсистема объединяет, проверяет и синхронизирует их), модели управления (включают в себя модели решаемой задачи и внешнего мира), машина вывода (позволяет с помо-

щью имеющихся данных и моделей получить и обосновать решение) и интерфейс пользователя.

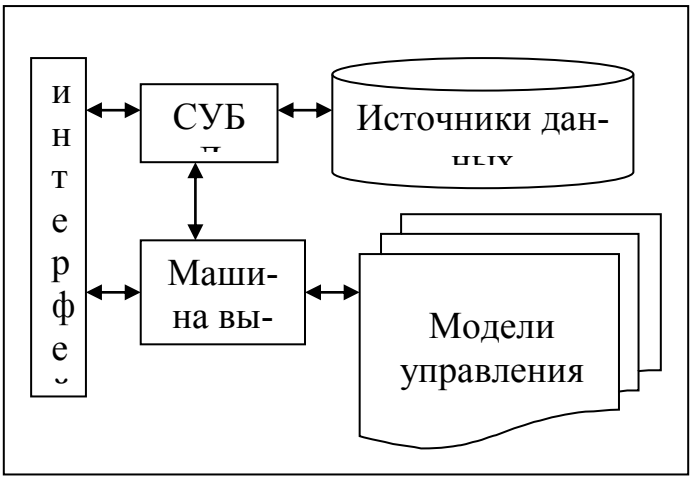


Рис. 12.4 - компоненты системы поддержки принятия решений

В зависимости от решаемой задачи в системах поддержки принятия решений могут использоваться различные методы принятия решений, привлекаться модели и методы, разработанные в рамках предметной области.

Контрольные вопросы

1. Назовите критерии качества естественно-языковых систем.
2. Какова роль агентов в мультиагентных системах?
3. В чем отличие оперативных и стратегических СППР?
4. Какие существуют способы понимания запросов естественного языка?
5. В чем заключается главная черта, отличающая ИМАС от других систем?

Заключение

Существует множество действующих СИИ в виде экспертных систем для различных предметных областей, начиная от космических приложений или управ-

ления боем до ухода за ребенком или людьми с ограниченными возможностями. При этом СИИ используют различные модели представления знаний, но наиболее эффективны модели в виде продукционных систем. СИИ уже продемонстрировали свою работоспособность, имеют уникальные потенциальные возможности, но остается много ограничений и открытых вопросов. Для улучшения существующих СИИ развиваются новые технологии представления знаний и декларативного программирования, расширены существующие методы и теоретические основы представления знаний. Особенно перспективно развитие интеллектуальных мультиагентных систем.

Как и людям, структуру рассуждений которых они копируют, СИИ в определенной мере свойственна неадекватность выводов. Выходом является исследование всех возможных вариантов для вывода знаний и поиска решений. В большой предметной области такая полная проверка практически неосуществима и должны применяться статистические методы для оценки функционирования.

СИИ обещают создание систем, выполняющих функции, бывшие ранее исключительной прерогативой человека, способные выполнять скучные, монотонные и опасные задания, а с развитием технологий представления и обработки знаний могут возникнуть совершенно новые приложения. Потенциальными приложениями СИИ также являются области, где человеческий интеллект малоэффективен, например в интеллектуальных системах, где обычные вычисления, трудоемки или неадекватны. В этом классе приложений, скорее всего, СИИ займут свое место наряду с обычными вычислениями в качестве дополнения.

Литература

1. Болотова Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях : учебник для вузов / Л. С. Болотова ; Министерство образования и науки Российской Федерации, Российский государственный уни-

верситет инновационных технологий и предпринимательства, Государственный научно-исследовательский институт информационных технологий и телекоммуникаций «Информатика». – М. : Финансы и статистика, 2012. – 664 с

2. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы: пер. с пол. И. Д. Рудинского [Электронный ресурс] / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия-Телеком, 2013. – 384 с. – Режим доступа http://e.lanbook.com/book/

3. Замятин Н. В. Нечеткая логика и нейронные сети : учеб. пособие / Н. В. Замятин ; рец.: И. А. Ходашинский, С. Н. Ливенцов ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники. – Томск : Эль Контент, 2014. – 146 с.

4. Гаврилов А. В. Системы искусственного интеллекта : учеб. пособие : в 2 ч. / А. В. Гаврилов. – Новосибирск : НГТУ, 2001. – Ч. 1. – 67 с.

5. Цуканова Н. И. Теория и практика логического программирования на языке Visual Prolog 7 : учеб. пособие [Электронный ресурс] / Н. И. Цуканова, Т. А. Дмитриева. – М. : Горячая линия-Телеком, 2013. – 232 с. – Режим доступа: http://e.lanbook.com/book/11847.

6. Абдикеев Н. М. Проектирование интеллектуальных систем в экономике [Электронный ресурс] / Н. М. Абдикеев. – Режим доступа: www.dgybga.ru/.../proektirovanie-intellektualnih-sistem-v-ekonomike-uchebnik-abdigeev...

7. Козлов А. Н. Интеллектуальные информационные системы : учебник [Электронный ресурс] / А. Н. Козлов ; Министерство сельского хозяйства РФ, ФГБОУ ВПО Пермская ГСХА. – Пермь : Изд-во ФГБОУ ВПО Пермская ГСХА, 2013. – 278 с. – Режим доступа: www.metodichka.x-pdf.ru/.../322646-1-ankozlov-intellektualnie-informacionnie-sistem

8. Интеллектуальные информационные системы и технологии : учеб. пособие [Электронный ресурс] / Ю. Ю. Громов, О. Г. Иванова, В. В. Алексеев и др. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2013. – 244 с.

9. Смагин А. А. Интеллектуальные информационные системы : учеб. пособие [Электронный ресурс] / А. А. Смагин, С. В. Липатова, А. С. Мельниченко. – Ульяновск : УлГУ, 2010. – 136 с.

10. Гущин А. Н. Экспертные системы : учеб. пособие [Электронный ресурс] / А. Н. Гущин, И. А. Радченко ; Балт. гос. техн. ун-т. – СПб., 2007. – Режим доступа: bezogr.ru/uchebnoe-posobie-sankt-peterburg-2007-gushin-a-n-radchenko-i-a.html.

11. Новиков Ф. А. Искусственный интеллект: представление знаний и методы поиска решений : учеб. пособие [Электронный ресурс] / Ф. А. Новиков. – СПб. : Изд-во Политехн. ун-та, 2010. – 240 с. – Режим доступа: window.edu.ru/resource/677/76677.

ГЛОССАРИЙ

А.

Агенда *{от англ.* расписание) — это тип структур для реализации стратегий управления -списков событий (или списков заявок). Дословно, agenda — повестка дня, расписание. Используется также написание русскими буквами — агенда. На основе структуры агенда реализуется стратегия управления агенда-система. Агенда-система работает с таким понятием, как источник знания.

Адаптивность — способность интеллектуальной системы (нейросетевой или иной) настраиваться в соответствии с данными. Синонимом этого понятия является способность к обучению.

Аксон *{от гр.* ось) — выходное окончание биологического нейрона, простая длинная ветвь нейрона, которая передает выходной сигнал из клетки. В искусственных нейронных сетях аксон моделируется как выход системы.

Анализ Что-Если **(What-If)** — способность и возможность спрашивать у компьютера, каким будет эффект при изменении некоторых входных данных. Анализ Что-Если можно представить следующим образом: что случится, произойдет с решением, если входные переменные, допущения или значения параметров изменятся? При допущении приемлемого пользовательского интерфейса, ЛПР легко может задать компьютерной модели вопросы такого типа и получить быстрые ответы.

Антецедент представляет собой утверждение типа "x есть низкий", где "низкий" - это терм (лингвистическое значение), заданный нечетким множеством на универсальном множестве лингвистической переменной x. Квантификаторы "очень", "более-менее", "не", "почти" и т.п. могут использоваться для модификации термов антецедента.

Ассоциативная память *(от лат.* соединение) — тип памяти, который позволяет ассоциировать одну вещь с другой. Ассоциативная память ИНС «вспоминает» наиболее близкий хранящийся обучающий образец, когда ей представляется простой входной образец. Ассоциативная память — это также способность восстанавливать полные ситуации из частичной информации.

Атрибут *(от лат.* наделенное, присовокупленное) — существенное, необходимое свойство, признак объекта, предмета или явления.

.

Б.

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

В.

Верификация знаний *(от лат.* делать истинным) — основная функция «Подсистемы верификации и совершенствования знаний» в интеллектуальных и экспертных системах. Эксперты обладают способностями верифицировать и совершенствовать знания. То есть, они могут анализировать свои собственные знания и их использование, обучаться от них и улучшать их для будущих консультаций. Аналогично, такая эволюция необходима в интеллектуальных системах при обучении, так, чтобы программа могла анализировать рассуждения под углом их успеха или неудачи. Это может привести к улучшениям, и как результату, более точным БЗ и более эффективному рассуждению. 466

Вес — значение, присваиваемое каждой связи на входе нейрона. Аналог синапса в биологическом нейроне мозга. Веса управляют входным потоком для обработки элементов в нейронной сети.

Визуализация данных *(от лат.* зрительный) — графическая анимация (оживление) или видео представление данных и результатов анализа данных.

Витрина данных – упрощенный вариант хранилища данных, содержащий только тематически объединенных данные

Вывод — процесс получения заключений из данных свидетельств. Достижение решения путем рассуждений.

Вывод на предикатах — в логической системе процесс получения знаний, выводимых из формул на основе аксиом с помощью правил вывода.

.

Е.

Естественный язык - в лингвистике любой язык общения между людьми. Под естественностью некоторого языка понимается наличие синонимии и омонимии слов и словосочетаний, а также свободный порядок слов в предложении.

Естественно-языковой интерфейс — интерфейс пользователя, который использует естественный человеческий язык для взаимодействия.

Естественно-языковой процессор — пользовательский интерфейс, основывающийся на технологии ИИ, который позволяет пользователю взаимодействовать с компьютерной системой таким же образом, как он разговаривает с другим человеком

Д.

Данные — это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства; сведения, полученные путем из-

мерения, наблюдения, логических или арифметических операций, представленные в форме, пригодной для постоянного хранения, передачи и (автоматизированной) обработки.

Дедукция *(от лат.* выведение) — в логике такая форма мышления, когда новая мысль выводится чисто логическим путем (т.е. по законам логики) из предшествующих мыслей и аксиом.

Декларативное представление знаний *(от лат.* заявление) — представление знаний, в котором описания выполняемых процедур не содержатся в явном виде. Декларативные представления это, как правило, множество фактов и утверждений, не зависящих от того, где они используются. Предметная область в такой форме представляется в виде синтаксического описания ее состояния.

Дендрит *(от гр.* дерево) — ветвящийся отросток ассоциативного нейрона, воспринимающий импульс от других нейронов. Дендриты обычно обслуживают входную функцию клетки, хотя многие дендриты имеют также выходные функции. В искусственных нейронных сетях дендрит моделируется входом нейрона.

Дерево вывода — схематический просмотр процесса вывода, который показывает порядок, в котором правила были проверены.

Дерево решений — графическое представление последовательности взаимосвязанных решений, которое описывает данные древовидными структурами. Дерево решений состоит из узлов, ветвей и листьев. Дерево всегда начинается с корневого узла и растет вниз, разделяя данные на каждом уровне на новые уз-

лы. Деревья решений являются альтернативным представлением таблицы решений и особенно хороши в решении классификационных задач. Их главными преимуществами является визуализация данных; представление отношений и связей в задаче графически способность обращения со сложными ситуациями в компактной форме. Однако оно не может быть громоздким, если в процесс вовлечены много альтернатив или состояний природы.

Дефаззификацией (defuzzification) называется процедура преобразования нечеткого множества в четкое число.

Домен *(от лат.* владение) — относительно узкая проблемная область или множество значений атрибута соответствующего объекта или отношения.

З.

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; это хорошо структурированные данные, или данные о данных, или метаданные.

Знания поверхностные. Поверхностные знания — знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области.

Знания глубинные — абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

знания процедурные- знания, представленные в алгоритмах.

знаниями декларативные считаются предложения, записанные на языках представления знаний, приближенных к естественному и понятных неспециалистам.

Знания эмпирические - знания, которые могут добываться ИС путем наблюдения за окружающей средой.

И.

Иерархическое рассуждение *(от гр.* священная власть) — метод рассуждения, основанный на дереве поиска, в котором определенные альтернативы, объекты или события могут быть исключены на разных уровнях иерархии поиска.

Индукция *(от лат.* наведение) — в логике, форма мышления, посредством которой мысль наводится на какое-либо общее правило, общее положение, присущее всем единичным предметам какого-либо класса.

Инженерия знаний — 1) теория, методология и технология интеллектуальных и экспертных систем, которые охватывают методы добычи, анализа и выражения в правилах знаний экспертов; 2) процесс построения интеллектуальной системы. Основные этапы этого процесса: оценка проблемы; приобретение данных и знаний; развитие прототипа системы; развитие завершенной системы; оценка и исправление системы; интеграция и сопровождение системы.

Инструментальные средства инженерии знаний — системы программирования, которые упрощают работу по созданию и проектированию интеллектуальных систем, в особенности пакеты для часто встречающихся задач, и языки высокого уровня для эвристического программирования.

Интеллект *(от лат.* ум, рассудок) — мыслительные способности человека к обучению, пониманию, рассуждению, умозаключениям, решению задач и принятию адекватных решений. Человек пытается повысить «интеллектуальные» возможности компьютера, передавая ему все более сложные функции по поиску и обработке информации. Машина мыслит интеллектуально, если она может достичь человеческого уровня выполнения некоторых когнитивных заданий.

Интеллектуальная БД — СУБД, которая проявляет черты искусственного интеллекта для помощи пользователю или разработчику. Технологии ИИ, особенно ЭС и искусственные нейронные сети, могут сделать доступ и манипуляции в сложных БД проще. Одним из путей является усиление роли СУБД в обеспечении этого совместно со способностью выведения заключений, что в результате получило общее название «интеллектуальная БД».

Интеллектуальная система поддержки решений (Интеллектуальная СППР) — интеллектуальная СППР — это компьютерная система, состоящая из 5 основных взаимодействующих компонентов: языковой подсистемы (связи между пользователем и другими компонентами интеллектуальной СПР), информационной подсистемы (хранилище данных и средств их обработки), подсистемы знаний (хранилище знаний о проблемной области, таких как процедуры, эвристики и правила, и средства обработки знаний), подсистемы моделей (хранилище моделей, языков моделирования, средств управления моделирова-

нием) и подсистемы обработки и решения задач (связующее звено между другими подсистемами).

Интенсиональная семантическая сеть — описывает предметную область на обобщенном, концептуальном уровне.

Интенсиональные знания *(от лат.* усиление, концентрация) — описывают абстрактные объекты, события, отношения, например, ПОСТАВЩИК, ПОТРЕБИТЕЛЬ, ТРАНСПОРТ.

Информационная система поддержки решений (ИСПР) — связывает интеллектуальные ресурсы управленца со способностями и возможностями компьютера для улучшения качества решений. Эти системы предназначены для менеджеров, принимающих управленческие решения в условиях слабоструктурированных и слабо определенных задач. См. также

Искусственная нейронная сеть (ИНС) - Artificialneuralnetwork — (ANN) — парадигма обработки информации, вдохновленная структурой и функциями человеческого мозга. ИНС — это компьютерная система, создающая значимые модели из большого количества данных. Она состоит из определенного количества простых взаимосвязанных процессоров, называемых нейронами, которые являются аналогами биологических нейронов мозга. Нейроны соединены связями с определенными весами, которые передают сигналы от одного нейрона к другому. В то время как в биологической нейронной сети в процесс обучения вовлекается регулирование синапсов, ИНС обучают посредством повторяющихся регулировок весов. Эти веса хранят знания, необходимые для решения конкретной задачи. ИНС работает с не доопределенной информацией и могут

распознавать модели, не слишком ясные для людей, и они адаптируются при получении новой информации.

Интеллектуальная система - система или устройство с программным обеспечением, имеющие возможность воспроизводить функции рассуждений человека.

Интеллектуальное программирование. Раздел искусственного интеллекта, занимающийся сочинением музыки, обучающих программ, работой с текстами, компьютерными играми.

Интеллектуальные информационные системы - это интеллектуальная система, имеющая мощную базу данных

Интеллектуальный редактор базы знаний — программа, представляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок и других сервисных средств, облегчающих работу с базой.

Интерпретация данных. Это одна из традиционных задач для экспертных систем. Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных.

Интерфейс пользователя — комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.

Искусственный интеллект (ИИ) — имитация некоторых видов интеллектуальной человеческой деятельности в электронных системах. Целью искусственного интеллекта как науки является создание компьютерных устройств и технологий, способных выполнять действия, которые требуют человеческого интеллекта.

Искусственная нейронная сеть (Artificial neural network)– это система, состоящая из многих простых вычислительных элементов (нейронов), работающих параллельно, функция которых определяется структурой сети, силой взаимных связей, а вычисления производятся в самих элементах или узлах.

З.

Знания — 1) теоретическое понимание предмета или явления и проверенный практикой опыт познания окружающего мира, отражение действительности в мышлении человека; 2) совокупность сведений у индивидуума, общества или у интеллектуальной системы о мире (конкретной предметной области, совокупности объектов или объекте), включающих в себя факты, убеждения, информацию о свойствах объектов, закономерностях процессов и явлений, эвристические правила использования этой информации для принятия решений. Существенным отличием знаний от данных является их интерпретируемость

К.

Кибернетика - наука об управлении, связи и переработке информации. Основным объектом исследования кибернетики являются абстрактные кибернетические системы: от компьютеров до человеческого мозга и человеческого общества.

Класс *(лат.* разряд) — термин, используемый в объектно-ориентированном программировании для обозначения группы предметов со сходными характеристиками.

Кластеризация (Clustering) – это один из методов анализа данных, позволяющих классифицировать многомерные наблюдения, каждое из которых описывается набором переменных X1,X2…Xn. Целью кластеризации является образование групп схожих между собой объектов.

Когнитология, когнитивная наука — Cognitivescience*(от лат.* познание) — междисциплинарная наука, изучающая процессы восприятия, познания, понимания, представления, мышления и обучения, и моделирующая принципы организации и работы естественных и искусственных интеллектуальных систем. Когнитология основывается на искусственном интеллекте, психологии, лингвистике, философии, нейрофизиологии и образовании.

Когнитолог. Инженер по знаниям — специалист в области искусственного интеллекта, выступающий в роли промежуточного буфера между экспертом и базой знаний. Синонимы: инженер-интерпретатор, аналитик.

Консеквент *(от лат.* следствие) — заключение или действие в правой части правила. Эта часть правила содержит описание действий, которые должны быть совершены над БД в случае выполнения соответствующих условий. В простейших продукционных системах они только определяют, какие элементы следует добавить (или иногда удалить) в БД.

Концепт *(от лат.* понимание) — описательная схема для класса вещей или конкретный пример такой схемы, в которой некоторые из ее общих свойств конкретизированы так, чтобы охарактеризовать конкретный подкласс или элемент, который является примером описания класса.

Л.

Лингвистическая переменная *(от лат.* язык) — переменная, которая может иметь значения в виде элементов языка, таких как слова и фразы. В нечеткой логике термины лингвистическая переменная и нечеткая переменная являются синонимами.

Логика высказываний, пропозициональная *(от лат.* слово, мысль; *лат.* суждение, высказывание) — формальная логическая система рассуждений, в которой заключения выводятся из ряда утверждений в соответствии со строгим множеством правил.

Логика предикатов *(от лат.* слово, мысль; *лат.* сказанное) — логическая система рассуждений, используемая в программах ИИ для обозначения отношений между группами данных. Логика предикатов является расширением логики высказываний, так как основным объектом здесь является переменное высказывание (предикат), истинность и ложность которого зависят от значений его переменных.

М.

Мета *(от гр.* после, за) — приставка, означающая рефлексивное применение соответствующего понятия.

Метод резолюций *(от лат.* решение) — метод рассуждения и вывода в логических системах, используемый в системах искусственного интеллекта. Метод предложен Робинсоном.

Механизм вывода — это та часть интеллектуальной системы, которая фактически исполняет функции рассуждения. Используя необходимые знания из БЗ и соответствующие данные, интеллектуальная система с использованием этого механизма может делать логический вывод, получая нужное знание, зачастую не хранящееся в явном виде в БЗ. В экспертных системах, основанных на правилах, его также называют управляющая структура или интерпретатор правил. Механизм вывода имеет два главных элемента: интерпретатор, который выполняет выбранные позиции агенды, используя соответствующие правила БЗ; планировщик, который поддерживает управление агендой. Механизм вывода реализует общую встраиваемую схему поиска решений.

Многослойный персептрон — наиболее общая топология ИНС, в которой персептроны связаны вместе для формирования слоев. Многослойный персептрон имеет входной слой, по крайней мере, один скрытый слой и выходной слой. Наиболее популярным методом обучения многослойного персептрона является обратное распространение.

Мобильный интеллектуальный агент *(от лат.* подвижный; умный; действующий) — интеллектуальный программный агент, который перемещается в Интернет от одного сайта к другому с целью поиска и доставки требуемой информации.

Модель *(от лат.* мера, образец) — материальный или идеальный аналог оригинала, создаваемый для хранения и расширения знания о нем. Это абстрактное представление, которое поясняет компоненты или связи явления.

«Мягкая» информация — нечеткая, неофициальная, субъективная, неясная, подразумеваемая и неопределенная информация.

«Мягкие» вычисления — Softcomputing— когнитивная методология, объединяющая такие современные направления искусственного интеллекта, как искусственные нейронные сети и нейрокомьютинг, нечеткую логику, индуктивную логику, эволюционные вычисления, генетическое программирование, теорию адаптации, интеллектуальный анализ данных, вычисления при помощи слов и обработку восприятий.

Мультиагентная система — система с многочисленными сотрудничающими интеллектуальными программными агентами

Н.

Нейробионическое направление. Это направление в развитии систем искусственного интеллекта, связанное с моделированием интеллектуальных способностей человеческого мозга.

Нейрокибернетика - научное направление, изучающее основные закономерности организации и функционирования нейронов и нейронных образований. Основным методом нейрокибернетики является математическое моделирование, при этом данные физиологического эксперимента используются в качестве исходного материала для создания моделей.

Нейрокомпьютеры - это системы, в которых алгоритм решения задачи представлен логической сетью элементов частного вида - нейронов с полным отказом от булевских элементов типа И, ИЛИ, НЕ. Как следствие этого введены специфические связи между элементами, которые являются предметом отдельного рассмотрения.

Нейрокомпьютинг - скоро развивающаяся область вычислительных технологий, стимулированная исследованиями мозга. Вычислительные операции выполняются огромным числом сравнимо обычных, нередко адаптивных процессорных частей. Нейрокомпьютинг, по собственному происхождению, совершенно приспособлен для сравнения образов, определения образов и синтеза систем управления.

Нейрон (биологический) – клетка мозга, способная генерировать электрический импульс, в случае, когда суммарный потенциал превысит критическую величину. Соединяясь друг с другом, нейроны образуют сеть, по которой путешествуют электрические импульсы, именно нейронные сети мозга обрабатывают информацию. При этом «обучение» сети и запоминание информации базируется на настройке значений весов связей между нейронами.

Нейронные сети – класс моделей, основанных на биологической аналогии с мозгом человека и предназначенных после прохождения этапа так называемого обучения на имеющихся данных для решения разнообразных задач анализа данных.

Нейронная сеть – это процессор с массивным распараллеливанием операций,

обладающий естественной способностью сохранять экспериментальные знания и делать их доступными для последующего использования. Он похож на мозг в двух отношениях: сеть приобретает знания в результате процесса обучения и для хранения информации используются величины интенсивности межнейронных соединений, которые называются синаптическими весами.

Нейронные вычисления — вычислительный подход к моделированию человеческого мозга, который основывается на соединении большого количества простых процессоров для генерации сложного поведения. Нейронные вычисления могут быть выполнены специализированным техническим обеспечением или программным обеспечением, называемым искусственные нейронные сети, которые имитируют структуру и функции человеческого мозга на компьютере. См. **Нейрокомпьютинг.**

Неопределенность является неотъемлемой частью процессов принятия решений. Их принято разделять на три класса: - неопределенность, связанная с неполнотой наших знаний о проблеме, по которой принимается решение; - Неопределенность, которая возникает в связи с непредсказуемостью реакции окружающей среды на наши действия; неопределенность – неточно понимаются цели непосредственно самим ЛПР.

Нейротрансмиттеры *(от гр.* нерв + *от лат.* пересылать) — химические вещества в биологической нейронной сети, высвобождаемые при достижении импульса между аксоном и дендритом синаптического окончания. Нейротрансмиттеры диффундируют через синапс, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать импульсы.

Нечеткий вывод — процесс рассуждения, основанный на нечеткой логике.

Нечеткая логика — способы рассуждения, которые могут справлять с неопределенностью и неполной информацией. Эта логическая система разработана для представления условий, которые не могут быть просто описаны при помощи бинарных терминов «истина» и «ложь». Была предложена Л. Заде в 1965 г. В отличие от Булевой логики, нечеткая логика является многозначной и обращается с понятием частичной истинности (значение истинности между «полностью истинно» и «полностью ложно»). Относится к теории нечетких множеств.

Нечеткая переменная — величина, которая может принимать лингвистические значения. Например, нечеткая переменная «скорость» может принимать такие значения, как «малая», «небольшая», «средняя», «высокая».

Нечеткая система — множество нечетких правил, преобразующих входные данные в выходные. В простейшем случае эксперт устанавливает эти правила словами или символами. В сложных нейро-нечетких системах нейросетевая система обучается правилам по данным или по наблюдениям за действиями людей-экспертов. На каждый пример входных данных в некоторой степени откликаются все правила в массивной ассоциативной памяти. Чем ближе сходство входного примера с частью «Если» нечеткого правила, тем больше отклик части «Тогда».

Нечеткое правило — условное высказывание вида «Если X есть A, Тогда Y есть B», где A и B — нечеткие множества. Правило есть связь между нечеткими множествами. Каждое правило определяет нечеткую область (декарто-

во произведение A x B) в «пространстве состояний» системы. Чем обширнее нечеткие множества A и B, тем обширнее и более неопределенная нечеткая область. Нечеткие правила нечетких систем являются блоками для построения знаний.

Нечеткая логика (Fuzzy logic). Умозаключение с использованием нечетких множеств или множеств нечетких правил. Это направление восходит к первым работам по нечетким множествам, выполенным Лофти Заде (Lofti Zaden) в 1960-1970 гг.

Нечеткое множество $A \subseteq X$ представляет собой набор пар $A = \{\langle x, \mu_A(x)\rangle | x \in U\}$, где $x \in X$ и μ_A — функция принадлежности, т.е. $\mu_A : U \to [0,1]$, которая представляет собой некоторую субъективную меру соответствия элемента нечеткому множеству и может принимать значения от нуля, который обозначает абсолютную не принадлежность, до единицы, которая, наоборот, говорит об абсолютной принадлежности элемента x нечеткому множеству A.

Нечетким логическим выводом называется получение заключения в виде нечеткого множества, соответствующего текущим значениях входов, с использованием нечеткой базы знаний и нечетких операций.

Нечеткой базой знаний называется совокупность нечетких правил "Если - то", определяющих взаимосвязь между входами и выходами исследуемого объекта. Обобщенный формат нечетких правил такой: Если посылка правила, то заключение правила.

Нечеткое правило - условное высказывание вида «если X есть A, то Y есть B», где A и B нечеткие множества

Нечетким числом называется выпуклое нормальное нечеткое множество с кусочно-непрерывной функцией принадлежности, заданное на множестве действительных чисел.

Нечеткая система - множество нечетких правил, преобразующих входные данные в выходные. В простейшем случае эксперт устанавливает эти правила, в более сложном, - например, нейросеть.

О.

Оболочка — Shell— развитое программное обеспечение для разработки интеллектуальных и экспертных систем, которое содержит базовую структуру системы, без деталей, относящихся к специальной задаче. Сложная экспертная система, лишенная своих специфических знаний (например, EMYCIN — EmptyMYCIN).

Обучающий алгоритм — обучающая процедура, используемая в искусственных нейронных сетях.

Оптимальное решение *{от лат.* наилучший) — наилучшее возможное решение моделированной задачи. Т.е. это решение, при котором степень достижения цели, связанная с ним, является наивысшей. Например, общая доходность максимизирована. Оптимальное решение может быть получено с использованием специального алгоритма.

Обучение. Под обучением понимается использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом «ученике» и его характерных ошибках, затем в работе они способны диагностировать слабости в познаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов ученика с целью передачи знаний.

Обучение нейронной сети (Training) - целенаправленный процесс изменения межслойных синаптических связей, итеративно повторяемый до тех пор, пока сеть не приобретет необходимые свойства.

Обучение с учителем или обучение контролируемое или обучение управляемое (Supervised learning, Associative learning). Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Предъявляется выходной вектор, вычисляется выход сети и сравнивается с соответствующим целевым вектором, разность(ошибка) с помощью обратной связи подается в сеть и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Обучение без учителя или самообучение или обучение неконтролируемое или обучение неуправляемое (Unsupervised learning, Self-organization) Алгоритм

обучения без учителя подстраивает веса сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения выделяет статистические свойства обучающего множества и группирует сходные векторы в классы.

Оптимизация – нахождение решения, удовлетворяющего системе ограничений и максимизирующим или минимизирующим целевую функцию.

Обратная цепочка рассуждений — метод вывода, в котором система начинает с того, что хочет доказать некоторое утверждение и пытается установить факты, необходимые для доказательства этого утверждения; либо система пытается установить условия, при которых возможна некоторая рассматриваемая ситуация.

П.

Планирование. Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности

Персептрон *{от лат.* восприятие, представление) — модель процесса восприятия, осуществляемого при помощи сети нейронов. Это простейшая форма нейронной сети, предложенная Ф. Розенблаттом. Операция персептрона основана на нейронной модели Маккалока-Питтса. Он состоит из простого нейрона с регулируемыми синапсическими весами и пороговым элементом. Персептрон обучается заданию путем выполнения регулировок весов с целью уменьшения различий между фактическими и желаемыми выходами. Начальные веса при-

сваиваются случайным образом, а затем модернизируются для получения выхода, совместимого с обучающим примером.

Поиск — организованный процесс проверки пространства возможных решений, гарантирующий эффективное нахождение приемлемого решения.

Поиск решений методами перебора на дереве решений. К методам перебора при поиске решений относятся: поиск в глубину, поиск в ширину, поиск на основе стоимости дуг, поиск с возвратом (бэктрекинг).

Порог — константа, которая используется для переменной в качестве уровня сравнения. Если значение переменной превышает порог, выполняется некоторое действие (например, нейрон возбуждается), а когда это значение ниже порога, никаких действий не происходит.

Предикат *{от лат.* сказанное) — логическая функция; в логике предикатов функция от любого числа аргументов, принимающая значения истинности 1 и 0.

Представление знаний — структурирование предметных знаний в интеллектуальной системе с целью облегчения поиска решения задачи.

Прогноз *(от гр.* предвидение, предсказание) — вероятное научно обусловленное утверждение о будущем развивающейся системы; суждение о состоянии какого-либо объекта, процесса или явления к определенному моменту времени в будущем. Он не является директивой, но дает набор альтернатив, область ра-

циональных решений. Любой прогноз по ряду причин имеет вероятностный характер.

Продукционная модель — набор, правил вида «условия-действие», где условиями являются утверждения о содержимом некой базы данных, а действия представляют собой процедуры, которые могут изменять содержимое БД.

Процедурное представление знаний — представление знаний, в котором семантика непосредственно заложена в описание элементов базы знаний. Процедурные знания содержат в явном виде описание некоторых процедур, обрабатывающих определенный участок базы знаний, за счет чего повышается эффективность поиска решений. При этом текущее состояние представляется в виде набора специализированных процедур.

Предметную область можно характеризовать описанием области в терминах пользователя, а задачи - их типом. С точки зрения разработчика выделяются статические и динамические предметные области. Предметная область называется статической, если описывающие ее исходные данные не изменяются во времени. При этом производные данные (выводимые из исходных) могут появляться заново и изменяться (не изменяя при этом исходных данных). Если исходные данные, описывающие предметную область, изменяются за время решения задачи, то предметную область называют динамической.

Принятие решения – это особый вид человеческой деятельности, направленный на выбор лучшей из имеющихся альтернатив. Главной задачей, которую приходится разрешать при принятии решения, является выбор альтернативы, наилучшей для достижения некоторой цели, или ранжирование множества воз-

можных альтернатив по степени их влияния на достижение этой цели.

Прогнозирование. Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками.

Подсистема объяснений — программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?» Ответ на вопрос «как» — это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, то есть всех шагов цепи умозаключений. Ответ на вопрос «почему» — ссылка на умозаключение, непосредственно предшествовавшее полученному решению, то есть отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.

Поле знаний – поле, в котором содержатся основные понятия, используемые при описании предметной области, и свойства всех отношений, используемых для установления связей между понятиями. Поле знаний связано с концептуальной моделью проблемной области, в которой еще не учтены ограничения, которые неизбежно возникают при формальном представлении знаний в базе знаний.

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 экспертной системы.

Приобретением знаний называется выявление знаний из источников и преобразование их в нужную форму, а также перенос в базу знаний ИС. Источниками знаний могут быть книги, архивные документы, содержимое других баз знаний и т. п., т. е. некоторые объективизированные знания, переведенные в форму, которая делает их доступными для потребителя.

Процесс принятия решений – получение и выбор наиболее оптимальной альтернативы с учетом просчета всех последствий. При выборе альтернатив необходимо выбирать ту, которая наиболее полно отвечает поставленной цели, но при этом приходится учитывать большое количество противоречивых требований и, следовательно, оценивать выбранный вариант решения по многим критериям.

Р.

Распознавание образов (Pattern recognition) - разделение образов в неком пространстве на классы. Образ традиционно представляется в виде вектора измеренных величин.

Распознавание образов — технология определения соответствия внешнего образца с образцом, хранящимся в памяти компьютера. Это процесс классификации данных на предварительно определенные категории. Используется в ме-

ханизмах вывода, обработке изображений, нейрокомпьютинге и в распознавании речи.

Распознавание речи — перевод человеческого голоса в индивидуальные слова и предложения, понимаемые компьютером.

Регрессионный анализ *(от лат.* движение назад; *гр.* разложение, разбор) — метод анализа трендов (тенденций), основывающийся на статистической прогнозирующей модели.

Репликация *(от лат.* обращать назад, отражать) — копирование изменений в таблицах базы данных и их пересылка на удаленные сервера без учета особенностей локальных объектов,

Репозитарий — база данных, где хранятся метаданные.

Решатель— программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: дедуктивная машина, машина вывода, блок логического вывода.

С.

Самоорганизующаяся карта Кохонена — специальный класс ИНС с соревновательным обучением, предложенный Т.Кохоненом в начале 1980-х гг. Карта Кохонена состоит из простого слоя вычислительных нейронов с двумя типами связей: прямые связи от нейронов на входном слое к нейронам на выходном слое, и горизонтальные связи между нейронами на выходном слое. Горизонтальные связи используются для создания соревнования между нейрона-

ми. Нейроны обучаются путем перемещения своих весов от неактивных связей к активным. Только выигравший нейрон и его соседи допускаются к обучению.

Семантика *(от гр.* обозначающий) — относящаяся к смыслу, значению или значимости символьного выражения, а не к его форме (т.е. не к его синтаксису). Смысловое значение языка. Отношения между словами и предложениями.

Сеть Хопфилда — нейронная сеть с обратной связью с простым слоем. В сети Хопфилда выход каждого нейрона связан с входами всех других нейронов (но не со своим входом).

Синапс биологический *(от гр.* соединение, связь) — химически проявляющаяся связь между двумя нейронами в биологической нейронной сети, реализуемая таким образом, что состояние одной клетки влияет на состояние другой. Синапсы обычно встречаются между аксоном и дендритом.

Синапс математический (вес, синаптический вес) - межнейронное соединение, однонаправленная входная связь нейрона, соединенная с выходом другого нейрона.

Синтаксис *(от гр.* составление) — 1) относящийся к форме или структуре символьного выражения, а не к его смыслу или значению; 2) способ, которым слова объединяются и располагаются для формирования фраз и предложений.

Синтез *(от гр.* соединение, составление) — метод научного исследования какого-либо предмета, процесса или явления, состоящий в познании его как единого целого, в единстве и взаимной связи его частей. Синтез связан в процессе познания с анализом.

Система поддержки принятия решений (СППР) — Decisionsupportsystem (DSS) — компьютерная информационная система на управленческом уровне организации, которая объединяет данные, экспертные знания и сложные аналитические модели, чтобы поддержать процесс решения неструктурированных задач с широким вовлечением в этот процесс пользователя. Поддерживает принятие решений менеджерами и аналитиками.

Системы искусственного интеллекта. Системы искусственного интеллекта, созданные для решения практических задач. Экспертные системы, системы поддержки принятия решений, мультиагентные системы, интеллектуальные системы естественно-языкового интерфейса.

Система управления базой знаний (СУБЗ) — программное обеспечение, которое призвано обеспечивать создание, сопровождение и применение баз знаний в интеллектуальных системах. Для построения СУБЗ, как одного из важнейших инструментальных средств новой технологии, необходима интеграция методов представления знаний в интеллектуальных БД и ИИ. Поскольку СУБЗ представляет собой инструментальное средство, с ней работают в первую очередь программисты — разработчики интеллектуальных систем и администраторы баз знаний (инженеры знаний) — специалисты, отвечающие за проектирование и сопровождение баз знаний в актуальном состоянии, т.е. в таком состоянии, которое правильно отражает внешнюю среду.

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

Слой — основная архитектурная компонента ИНС, состоящая из множества процессорных элементов с одинаковыми функциональными возможностями и занимающая в сети положение, соответствующее определенной стадии обработки.

Слот *(от англ.* отверстие, щель) — признак или описание компоненты некоторого объекта во фрейме. Слоты могут соответствовать внутренне присущим признакам, таким как имя, определение или происхождение, или же представлять такие выведенные атрибуты, как значение, важность, или другие подобные объекты. Другими словами, слот — это атрибут, связанный с узлом в системе, основанной на фреймах. Заполнение слота приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.

Соединение — связь от одного нейрона к другому для передачи сигналов. Также упоминается как синапс, который часто ассоциируется с весом. Вес определяет силу передаваемого сигнала.

Стратегии управления обеспечивают разнообразное управление в рамках принятой для данной системы схемы механизма вывода.

Сходимость процесса обучения (Coincidence of the learning algorithm). Целью процедуры минимизации является отыскание глобального минимума, достиже-

ние его называется сходимостью процесса обучения.

Т.

Теория уверенности — теория для управления неопределенностью в интеллектуальных и экспертных системах, основанная на неточном рассуждении. Она использует коэффициенты уверенности для представления уровня доверия в гипотезе, что данное событие наблюдалось. Основные принципы этой теории были предложены Бухананом и Шортлиффом в 1975 г

Теория нечетких множеств — теория, разработанная Л. Заде в 1965 г. Аппарат теории нечетких множеств предназначен для решения широкого круга проблем, в которых важное место занимают субъективные, трудно формализуемые знания человека. Особый интерес эта вызывает в связи с исследованиями и разработками человеко-ориентированных социальных и управленческих систем, в частности, экспертных систем.

Терм–множеством (term set) называется множество всех возможных значений лингвистической переменной.

Термом (term) называется любой элемент терм–множества. В теории нечетких множеств терм формализуется нечетким множеством с помощью функции принадлежности.

Теория игр - математическая теория, предсказания результатов игр, в которых участники не имеют полной информации о намерениях друг друга.

Теория принятия решений — область исследования, изучающая закономерности выбора людьми путей решения разного рода задач и исследует способы поиска наиболее выгодных из возможных решений.

Трассировки рассуждений — механизм в языке инженерии знаний, который может предъявить правила или выполняемые программы, включая значения используемых правил. Трассировка рассуждений при объяснении предусматривает пересечение дерева целей для ответа на вопросы. Основываясь на дереве целей, система объяснений может объяснять, как было получено заключение.

Э.

Эволюционное моделирование - направление в математическом моделировании, объединяющее компьютерные методы моделирования эволюции, а также близкородственные по источнику заимствования идей другие направления в эвристическом программировании.

Эвристика – процесс поиска решений; прием решения задачи, основанный не на строгих математических моделях и алгоритмах, а на соображениях, восходящих к "здравому смыслу", отражает особенности того, как такие задачи решает человек, когда он не пользуется строго формальными приемами.

Х.

Хранилище данных – предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.

У.

Управление. Под управлением понимается функция организованной системы, поддерживающая определенный режим деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями.

Управление знаниями — активное управление специальными знаниями в организации, включая сбор, категоризацию и распространение знаний.

Ф.

Фаззификацией (fuzzification) называется процедура преобразования четких значений в степени уверенности посредством функций принадлежности.

Формализация. Процесс формализации знаний, полученных у эксперта, состоит из следующих шагов: выбор метода измерения нечеткости, получение исходных данных посредством опроса эксперта, реализация алгоритма построения функции принадлежности.

Фреймы *(от англ,* рама, остов) — метод представления знаний, основанный на минимальных структурах информации, необходимых для представления класса объектов, явлений или процессов. Фреймы связывают свойства с узлами, представляющими понятия и объекты. Свойства описываются атрибутами, называемыми слотами и их значениями. Фреймы используются для представления знаний в интеллектуальных системах и ЭС, основанных на фреймах.

Функция принадлежности — математическая функция, которая определяет нечеткое множество на пространстве рассуждений. Типичные формы функций

принадлежности, используемых в нечетких интеллектуальных и экспертных системах, это треугольники и трапеции.

Функция цели — математическая функция, которая связывает переменные с целью, измеряет степень достижения цели и должна быть оптимизирована.

Э.

Эвристика *(от гр.* нахожу) — эмпирическое правило или стратегия, которая может использоваться при решении сложных задач, упрощая или ограничивая поиск решений в предметной области, которая является сложной или недоступной ясному пониманию. Эвристики включают знания о том, как решать задачи эффективно, как планировать шаги при решении сложной задачи, как улучшать исполнение и т.д. Они развиваются из многолетнего опыта и являются неформальными субъективными знаниями о предметной области.

Экспертная система — система, которая использует человеческие знания и логику эксперта, встраиваемые в компьютер, для обеспечения высокоэффективного решения задач в некоторой узкой предметной области. Такие системы, как правило, представляют знания символически, исследуют и объясняют свои процессы рассуждения и предназначены для предметных областей, в которых людям для достижения мастерства необходимы годы специального обучения и практики.

Экспертные знания - знания, которые имеются у специалистов, но не зафиксированы во внешних по отношению к нему хранилищах. Экспертные знания являются субъективными.

Эпоха (Epoch) – один этап функционирования нейронной сети при обучении.