МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЛУГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ВЛАДИМИРА ДАЛЯ»

Стахановский инженерно-педагогический институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования «Луганский государственный университет имени Владимира Даля»
Кафедра информационных систем

КОНСПЕКТ ЛЕКЦИЙ

по дисциплине

«ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»

для студентов направления подготовки
Профессиональное обучение (по отраслям),
магистерская программа «Информационные технологии и системы»

Луганск 2023

УДК 69.032/007, 612.313

Рекомендовано к изданию Учебно-методическим советом
ГОУ ВО ЛНР «ЛГУ им. В. ДАЛЯ»
(протокол № от 2023 г.)

Конспект лекций по дисциплине **«Интеллектуальные информационные системы»** для студентов направления подготовки **Профессиональное обучение (по отраслям)**, магистерская программа «Информационные технологии и системы» / Сост.: Д. С. Тимошенко. – **Стаханов:** ФГБОУ ВО «ЛГУ им. В. Даля», 2023. – 86 с.

Конспект лекций содержит семь лекций по дисциплине «Интеллектуальные информационные системы», которые включают в себя теоретические сведения, примеры решения, задачи и варианты с данными для их выполнения. К каждой лабораторной работе приведены вопросы для самопроверки. В методических рекомендациях представлен список использованных источников.

Предназначены для студентов магистерской программы «Информационные технологии и системы».

Составители: ст. преп. Тимошенко Д.С.

Ответственный за выпуск: доц. Карчевский В.П.

Рецензент: доц. Карчевская Н.В.

Аннотация

Для получения теоретических знаний и приобретения необходимых умений, программой учебной дисциплины предусмотрено проведение лекционных занятий. Для проведения лекционных занятий составлен конспект лекций, который содержат теоретический материал и практические примеры для создания интеллектуальных систем.

Целью освоения дисциплины Интеллектуальные информационные системы является: дать будущим специалистам в области информационных систем, знания и навыки в сфере состояния и перспектив развития ИИС, которые требуются для понимания сути использования современных ИКТ в процессе количественного и качественного обоснования принимаемых решений на предприятиях самого различного направления и масштаба; изучить классификацию, отличия, структуру построения, функционирование ИИС; используемые для этого методы, принципы и ИКТ.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ В ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Рассматриваются понятия искусственного интеллекта (ИИ) и интеллектуальной системы. Сделан краткий исторический обзор работ в области ИИ, робототехники и промышленных роботов.

1.1. Предмет исследования искусственного интеллекта

Введение: Жизнь в XXI веке

Утром Вы спрашиваете персонального кноубота (knowledge robot), каковы обязательные мероприятия на сегодня, уточняете расписание. Затем садитесь в ALV (Autonomous Land Vehicle) и он отвозит Вас в офис по оптимальному маршруту за минимальное время. По дороге через Сеть Вы заказываете тур на Гавайи и делаете покупки. Авиабилеты, билеты на концерты и экскурсии, квитки на забронированные места в отеле, соответствующие Вашим требованиям и бюджету, а так же приобретенные товары доставляются по требуемому адресу через пару часов. Интеллектуальные агенты помогают выполнять любую работу: слежение за сложными технологическими процессами, разработка долговременных стратегий, принятие решений, подбор наилучших кандидатов на вакантные должности, организация различных мероприятий и многое другое.

Все эти системы реальны уже сегодня. Например, ALV разработаны в Карнеги Мэлон Университете (США), они могут самостоятельно управлять авто средствами в условиях городского трафика на дистанции до 50 км. Заказами через Сеть никого не удивишь, только качество сервиса еще не достаточно высокое. Интеллектуальные системы используются на многих предприятиях, но, к сожалению, все еще остаются слишком дорогими.

В 1950 году британский математик Алан Тьюринг опубликовал в журнале «Mind» свою работу «Вычислительная машина и интеллект», в которой описал тест для проверки программы на интеллектуальность. Он предложил поместить исследователя и программу в разные комнаты и до тех пор, пока исследователь не определит, кто за стеной - человек или программа, считать поведение программы разумным. Это было одно из первых определений интеллектуальности, то есть А. Тьюринг предложил называть интеллектуальным такое поведение программы, которое будет моделировать разумное поведение человека.

1.2. Определение ИИС

С тех пор появилось много *определений* ***интеллектуальных систем*** (*ИС*) и ***искусственного интеллекта*** (*ИИ*).

Сам термин *ИИ (AI - Artificial Intelligence)* был предложен в 1956 году на семинаре в Дартсмутском колледже (США). Приведем некоторые из этих определений. Д. Люгер в своей книге [2] определяет «***ИИ*** как область компьютерных наук, занимающуюся исследованием и автоматизацией разумного поведения».

В учебнике по *ИС* [3] дается такое определение: «***ИИ*** - это одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными задачи, общаясь с ЭВМ на ограниченном подмножестве естественного языка».

1.3. Искусственный интеллект и интеллектуальное поведение

Понятие *искусственный интеллект* (ИИ) различными авторами трактуется по-

разному, но большинство согласно, что оно базируется на двух главных идеях: изучение мыслительных процессов человека и представление этих процессов с помощью машин (компьютеров, роботов и т.д.). Выделяют три основных цели разработок в области ИИ:

1. сделать машины умнее (первоначальная цель);
2. понять, что такое интеллект (научная цель);
3. сделать машины более полезными (предпринимательская цель).

На рис. 1 представлены основные области приложения ИИ, перечислены разделы науки, на базе которых возникло научное направление ИИ.

Под *интеллектуальным поведением* понимается такое поведение машины, которое соответствует человеческому. Интеллектуальным может быть названо, например, следующее поведение:

- самообучение;
- понимание двусмысленных или противоречивых сообщений;
- быстрое и правильное реагирование на новую ситуацию;
- эффективное использование процедуры заключений (выводов) для решения проблем;
- анализ сложных ситуаций;
- предсказание.

Применение ИИ позволяет:

1) строить интеллектуальный (дружественный) интерфейс в информационных системах;
2) решать задачи, которые не могут быть решены обычными методами;
3) значительно увеличить скорость и качество решения задач;
4) решать задачи в условиях неполноты данных;
5) анализировать большие объемы информации;
6) понимать речь, ручное письмо и т.д.

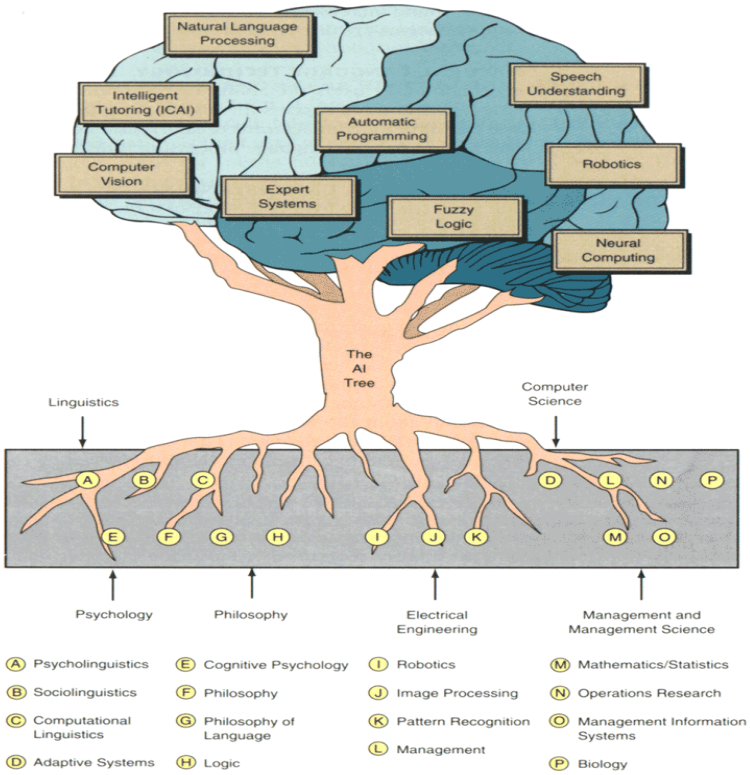


Рис. **Ошибка! Закладка не определена.**. Дисциплины и области применения ИИ

Обычные (или процедурные) программы основаны на алгоритмах. Алгоритм представляет собой жесткую последовательность некоторых математических формул, которая приводит к решению задачи. Интеллектуальные программы базируются на исчислениях, которые являются более общим понятием, чем алгоритмы. Процесс решения задачи может быть неизвестен заранее, программа оперирует с правилами вывода, имитирующими человеческие рассуждения.

1.4. Определения, используемые в дисциплине ИИС

Предметом информатики является обработка информации по известным законам.

Предметом *ИИ* является изучение интеллектуальной деятельности человека, подчиняющейся заранее неизвестным законам. *ИИ* это все то, что не может быть обработано с помощью алгоритмических методов.

Система - множество элементов, находящихся в отношениях друг с другом и образующих причинно-следственную связь.

Адаптивная система - это система, которая сохраняет работоспособность при непредвиденных изменениях свойств управляемого объекта, целей управления или окружающей среды путем смены алгоритма функционирования, программы поведения или поиска оптимальных, в некоторых случаях просто эффективных, решений и состояний.

Традиционно, по способу адаптации различают ***самонастраивающиеся, самообучающиеся и самоорганизующиеся системы*** [4].

Алгоритм - последовательность заданных действий, которые однозначно определены и выполнимы на современных ЭВМ за приемлемое время для решаемой задачи.

ИС - это *адаптивная система*, позволяющую строить программы целесообразной деятельности по решению поставленных перед ними задач на основании конкретной ситуации, складывающейся на данный момент в окружающей их среде [5].

Сделаем два важных дополнения к данному *определению*.

1. К сфере решаемых *ИС* задач относятся задачи, обладающие, как правило, следующими особенностями:
   - в них неизвестен алгоритм решения задач (такие задачи будем называть интеллектуальными задачами);
   - в них используется помимо традиционных данных в числовом формате информация в виде изображений, рисунков, знаков, букв, слов, звуков;
   - в них предполагается наличие выбора (не существует алгоритма - это значит, что нужно сделать выбор между многими вариантами в условиях неопределенности). Свобода действий является существенной составляющей интеллектуальных задач.
2. *Интеллектуальные робототехнические системы* (*ИРС*) содержат переменную, настраиваемую модель внешнего мира и реальной исполнительной системы с объектом управления. Цель и управляющие воздействия формируются в *ИРС* на основе *знаний* о внешней среде, объекте управления и на основе моделирования ситуаций в реальной системе.

О каких признаках интеллекта уместно говорить применительно к *интеллектуальным системам*? *ИС* должна уметь в наборе фактов распознать

существенные, *ИС* способны из имеющихся фактов и *знаний* сделать выводы не только с использованием дедукции, но и с помощью аналогии, индукции и т. д. Кроме того, *ИС* должны быть способны к самооценке - обладать рефлексией, то есть средствами для оценки результатов собственной работы. С помощью подсистем объяснения *ИС* может ответить на вопрос, почему получен тот или иной результат. Наконец, *ИС* должна уметь обобщать, улавливая сходство между имеющимися фактами.

Можно ли считать шахматную программу *интеллектуальной системой*? Если шахматная программа при повторной игре делает одну и ту же ошибку - то нельзя. Обучаемость, адаптивность, накопление опыта и *знаний* - важнейшие свойства интеллекта. Если шахматная программа реализована на компьютере с бесконечно-высоким быстродействием и обыгрывает человека за счет просчета всех возможных вариантов игры по жестким алгоритмам - то такую программу мы также не назовем интеллектуальной. Но если шахматная программа осуществляет выбор и принятие решений в условиях неопределенности на основе эффективных методов принятия решений и эвристик, корректируя свою игру от партии к партии в лучшую сторону, то такую программу можно считать достаточно интеллектуальной.

Всякий раз, как только возникают сомнения в интеллектуальности некоторой системы, договоримся вспоминать тест Алана Тьюринга на интеллектуальность. После этого сомнения и дальнейшие споры, как правило, прекращаются.

Следует определить также понятие ***знания*** - центрального понятия в *ИС*. Рассмотрим несколько определений.

1. ***Знания*** есть результат, полученный познанием окружающего мира и его объектов.
2. ***Знания*** - система суждений с принципиальной и единой организацией, основанная на объективной закономерности.
3. ***Знания*** - это формализованная информация, на которую ссылаются или которую используют в процессе логического вывода (рис. 1.1).
4. Под ***знаниями*** будем понимать совокупность фактов и правил. Понятие правила, представляющего фрагмент *знаний*, имеет вид: ***если <условие> то <действие>***

Например, если X истинно и Y истинно, то Z истинно с достоверностью P.

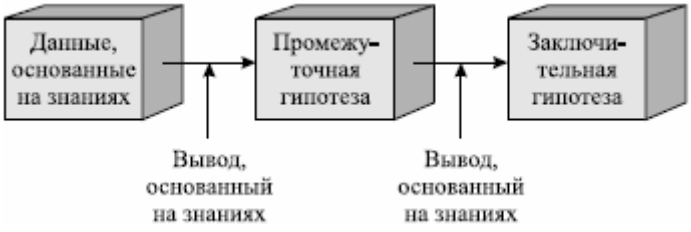


Рис. 1.1. Процесс логического вывода в ИС

Определения 1 и 2 являются достаточно общими философскими определениями. В ИС принято использовать определение 3 для определения *знаний*. Определение 4 есть частный случай определения 3.

Под статическими *знаниями* будем понимать *знания*, введенные в *ИС* на этапе проектирования. Под динамическими *знаниями* (опытом) будем понимать *знания*, полученные *ИС* в процессе функционирования или эксплуатации в реальном масштабе времени.

Знания можно разделить на факты и правила. Под фактами подразумеваются *знания* типа «А это А», они характерны для баз данных. Под правилами (продукциями) понимаются *знания* вида «ЕСЛИ-ТО». Кроме этих *знаний* существуют так называемые

метазнания (*знания* о *знаниях*). Создание продукционных систем для представления *знаний* позволило разделить *знания* и управление в компьютерной программе, обеспечить модульность продукционных правил, т. е. отсутствие синтаксического взаимодействия между правилами. При создании моделей представления *знаний* следует учитывать такие факторы, как однородность представления и простота понимания. Выполнить это требование в равной степени для простых и сложных задач довольно сложно.

Рассмотрим подробнее систему управления *ИРС*, структурная схема которой представлена на рис. 1.2. На этом рисунке стрелками обозначено направление движения информации, двунаправленными стрелками обозначено взаимодействие типа «запрос-ответ» и «действие-подтверждение», весьма распространенное в информационных системах. Входом системы является Блок ввода информации, предназначенный для ввода числовых данных, текста, речи, *распознавания изображений*. Информация на вход системы может поступать (в зависимости от решаемой задачи) от пользователя, внешней среды, объекта управления. Далее входная информация поступает в Блок логического вывода, либо сразу в базу данных (БД) - совокупность таблиц, хранящих, как правило, символьную и числовую информацию об объектах предметной области (в нашем курсе лекций - объектах *робототехники*).

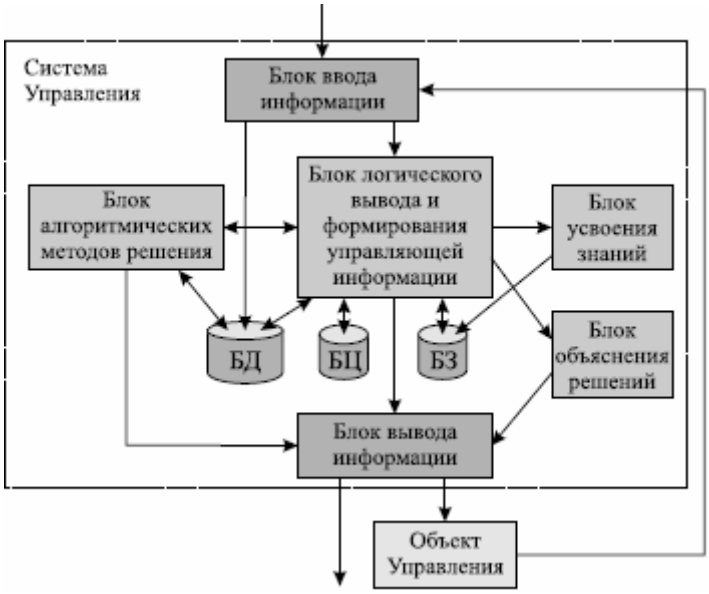


Рис. 1.2. Структурная схема интеллектуальной робототехнической системы

Блок логического вывода (БЛВ) и формирования управляющей информации обеспечивает нахождение решений для нечетко формализованных задач *ИС*, осуществляет планирование действий и формирование управляющей информации для пользователя или объекта управления на основе Базы Знаний (БЗ), БД, Базы Целей (БЦ) и Блока Алгоритмических Методов Решений (БАМР).

БЗ - совокупность *знаний*, например, система продукционных правил, о закономерностях предметной области.

БЦ - это множество локальных целей системы, представляющих собой совокупность *знаний*, активизированных в конкретный момент и в конкретной ситуации для достижения глобальной цели.

БАМР содержит программные модули решения задач предметной области по жестким алгоритмам.

Блок усвоения *знаний* (БУЗ) осуществляет анализ динамических *знаний* с целью их усвоения и сохранения в БЗ.

Блок объяснения решений (БОР) интерпретирует пользователю последовательность логического вывода, примененную для достижения текущего результата.

На выходе системы Блок вывода информации обеспечивает вывод данных, текста, речи, изображений и другие результаты логического вывода пользователю и/или Объекту Управления (ОУ).

Контур обратной связи позволяет реализовать свойства адаптивности и обучения *ИС*. На этапе проектирования эксперты и инженеры по *знаниям* наполняют базу *знаний* и базу целей, а программисты разрабатывают программы алгоритмических методов решений. База данных создается и пополняется, как правило, в процессе эксплуатации *ИС*.

Динамика работы *ИРС* может быть описана следующим образом. При поступлении информации на внешнем языке системы на вход БВИ производится ее интерпретация во внутреннее представление для работы с символьной моделью системы. БЛВ выбирает из БЗ множество правил, активизированных поступившей входной информацией, и помещает эти правила в БЦ как текущие цели системы. Далее БЛВ по заданной стратегии, например, стратегии максимальной достоверности, выбирает правило из БЦ и пытается доопределить переменные модели внешнего мира и исполнительной системы с объектом управления. На основе этого активизируются новые правила БЗ и начинается логический вывод в системе продукций (правил). Эта процедура заканчивается, как только решение будет найдено, либо когда будет исчерпана БЦ. Найденное решение из внутреннего представления интерпретируется Блоком Вывода информации во внешний язык подсистемы управления низшего уровня и объекта управления. Более подробно этот процесс рассматривается в разделе 3.

1.5. Исторический обзор работ в области ИИ

Среди важнейших классов задач, которые ставились перед *ИИ* с момента его зарождения как научного направления (с середины 50-х годов XX века), следует выделить следующие трудно формализуемые задачи, важные для задач *робототехники*: *доказательство теорем*, *управление роботами*, *распознавание изображений*, *машинный перевод* и *понимание текстов на естественном языке*, *игровые программы*, *машинное творчество* (*синтез музыки*, *стихотворений*, текстов).

Доказательство теорем.

Изучение приемов *доказательства теорем* сыграло важную роль в развитии *ИИ*. Формализация дедуктивного процесса с использованием логики предикатов помогает глубже понять некоторые компоненты рассуждений. Многие неформальные задачи, например, медицинская диагностика, допускают формализацию как задачу на *доказательство теорем*. Поиск *доказательства математической теоремы* требует не только произвести дедукцию, исходя из гипотез, но также создать интуитивные догадки и гипотезы о том, какие промежуточные утверждения следует доказать для вывода *доказательства* основной теоремы.

В 1954 году А. Ньюэлл задумал создать программу для игры в шахматы. Дж. Шоу и Г. Саймон объединились в работе по проекту Ньюэлла и в 1956 году они создали язык программирования IPL-I (предшественник LISPa) для работы с символьной информацией. Их первыми программами стала программа LT (Logic Theorist) для *доказательства теорем* и исчисления высказываний (1956 год), а также

программа NSS (Newell, Shaw, Simon) для игры в шахматы (1957 год). LT и NSS привели к созданию А. Ньюэллом, Дж. Шоу и Г. Саймоном программы GPS (General Problem Solver) в 1957-1972 годах Программа GPS моделировала используемые человеком общие стратегии решения задач и могла применяться для решения шахматных и логических задач, *доказательства теорем*, грамматического разбора предложений, математического интегрирования, головоломок типа «Ханойская башня» и т. д. Процесс работы GPS воспроизводит методы решения задач, применяемые человеком: выдвигаются подцели, приближающие к решению, применяется эвристический метод (один, другой и т. д.), пока не будет получено решение. Попытки прекращаются, если получить решение не удается. Программа GPS могла решать только относительно простые задачи. Ее универсальность достигалась за счет эффективности. Специализированные «решатели задач» - STUDENT (Bobrov, 1964) и др. лучше проявляли себя при поиске решения в своих предметных областях. GPS оказалась первой программой (написана на языке IPL-V), в которой предусматривалось планирование стратегии решения задач.

Для решения трудно формализуемых задач и, в частности, для работы со *знаниями* были созданы языки программирования для задач *ИИ*: LISP (1960 год, J. MacCatthy), Пролог (1975-79 годы, D. Warren, F. Pereira), ИнтерLISP, FRL, KRL, SMALLTALK, OPS5, PLANNER, QA4, MACSYMA, REDUCE, РЕФАЛ, CLIPS. К числу наиболее популярных традиционных языков программирования для создания *ИС* следует также отнести C++ и Паскаль.

Распознавание изображений.

Рождение *робототехники* выдвинуло задачи машинного зрения и *распознавания изображений* в число первоочередных.

В традиционном *распознавании образов* появился хорошо разработанный математический аппарат, и для не очень сложных объектов оказалось возможным строить практически работающие системы классификации по признакам, по аналогии и т. д. В качестве признаков могут рассматриваться любые характеристики распознаваемых объектов. Признаки должны быть инвариантны к ориентации, размеру и вариациям формы объектов. Алфавит признаков придумывается разработчиком системы. Качество *распознавания* во многом зависит от того, насколько удачно придуман алфавит признаков. *Распознавание* состоит в априорном получении вектора признаков для выделенного на изображении отдельного распознаваемого объекта, и лишь затем в определении того, какому из эталонов этот вектор соответствует.

П. Уинстон в начале 80-х годов обратил внимание на необходимость реализации целенаправленного процесса машинного восприятия. Цель должна управлять работой всех процедур, в том числе и процедур нижнего уровня, т. е. процедур предварительной обработки и выделения признаков. Должна иметься возможность на любой стадии процесса в зависимости от получаемого результата возвращаться к его началу для уточнения результатов работы процедур предшествующих уровней. У П. Уинстона, так же как и у других исследователей, до решения практических задач дело не дошло, хотя в 80-е годы вычислительные мощности больших машин позволяли начать решение подобных задач. Таким образом, ранние традиционные системы *распознавания*, основывающиеся на последовательной организации процесса распознавания и классификации объектов, эффективно решать задачи восприятия сложной зрительной информации не могли.

Экспертные системы.

Методы *ИИ* нашли применение при создании автоматических консультирующих систем. До 1968 года исследователи в области *ИИ* работали на основе общего подхода - упрощения комбинаторики, базирующегося на уменьшении перебора альтернатив исходя из здравого смысла, применения числовых функций оценивания и различных эвристик.

В начале 70-х годов произошел качественный скачок и пришло понимание, что необходимы глубокие *знания* в соответствующей области и выделение *знаний* из данных, получаемых от эксперта. Появляются экспертные системы (ЭС), или системы, основанные на *знаниях*.

ЭС DENDRAL (середина 60-х годов, Стэнфордский университет) расшифровывала данные масс-спектрографического анализа.

ЭС MYCIN (середина 70-х годов, Стэнфордский университет) ставила диагноз при инфекционных заболеваниях крови.

ЭС PROSPECTOR (1974-1983 годы, Стэнфордский университет) обнаруживала полезные ископаемые.

ЭС SOPHIE обучала диагностированию неисправностей в электрических цепях. ЭС XCON помогала конфигурировать оборудование для систем VAX фирмы DEC, ЭС PALLADIO помогала проектировать и тестировать СБИС-схемы.

ЭС JUDITH помогает специалистам по гражданским делам и вместе с юристом и с его слов усваивает фактические и юридические предпосылки дела, а затем предлагает рассмотреть различные варианты подходов к разрешению дела.

ЭС LRS оказывает помощь в подборе и анализе информации о судебных решениях и правовых актах в области кредитно-денежного законодательства, связанного с использованием векселей и чеков.

ЭС «Ущерб» на основе российского трудового законодательства обеспечивает юридический анализ ситуации привлечения рабочих и служащих к материальной ответственности при нанесении предприятию материального ущерба действием или бездействием.

Список созданных ЭС можно перечислять очень долго. Были разработаны и внедрены тысячи реально работающих экспертных систем. (См темы6 и 7).

Разработка инструментальных средств для создания ЭС ведется постоянно. Появляются экспертные системы оболочки, совершенствуются технологии создания ЭС. Язык Пролог (1975-79 годы) становится одним из основных инструментов создания ЭС. Язык CLIPS (C Language Integrated Production System) начал разрабатываться в космическом центре Джонсона NASA в 1984 году [6]. Язык CLIPS свободен от недостатков предыдущих инструментальных средств для создания ЭС, основанных на языке LISP. Появляется инструментарий EXSYS, ставший в начале 90-х годов одним из лидеров по созданию ЭС [7]. В начале XXI века появляется теория интеллектуальных агентов и экспертных систем на их основе [8]. Web-ориентированный инструментарий JESS (Java Expert System Shell), использующий язык представления *знаний* CLIPS, приобрел достаточную известность в настоящее время [9]. Среди отечественных инструментальных средств следует отметить веб-ориентированную версию комплекса АТ-ТЕХНОЛОГИЯ, разработанного на кафедре Кибернетики МИФИ. В этом комплексе вся прикладная логика как комплекса в целом, так и разработанных в нем веб-интегрированных ЭС, сосредоточена на стороне

сервера [10].

Практика внедрения ЭС показала, что нет чудодейственных рецептов - нужна кропотливая работа по вводу в ЭВМ опыта и *знаний* специалистов всех областей науки.

Машинный перевод и понимание текстов на естественном языке.

Началом работ по *машинному переводу* следует считать 1954 год, когда в США с помощью ЭВМ было переведено шестьдесят фраз. Этот известный «Джорджтаунский эксперимент» произвел неизгладимое впечатление на специалистов. Тогда казалось, что достаточно создать большие хранилища словарей для перевода с одного языка на другой, разработать правила перевода - и проблема будет решена. Когда выяснилось, что проблема не так проста, был создан язык-посредник, облегчающий сопоставление фраз на разных языках. Во второй половине семидесятых годов этот язык-посредник превратился в семантическую модель представления смысла переводимых текстов. Таким образом, достижения в области *машинного перевода* оказались очень полезными для *ИИ*, так как они показали, что анализ естественно-языковых конструкций невозможен без создания семантической модели, в которой интерпретируется смысл слов. Создание семантической модели привело к пониманию необходимости создания языка для внутреннего представления *знаний*. И, наконец, важным достижением явилось понимание того, что анализ текстов на естественном языке (ЕЯ) состоит из четырех основных этапов: морфологический анализ, синтаксический анализ, семантический анализ, прагматический анализ [11].

Задача разбора предложения на естественном языке даже с ограниченным словарем в сотню слов является типичной задачей *ИИ*. Для достаточно формальных предметных областей и конкретных наборов предметов, обладающих однозначными, недвусмысленными характеристиками и названиями, такая система была создана Т. Виноградом в 1971 году. Его знаменитая программа SHRDLU [12] работает в абстрактном мире, состоящем из разноцветных кубиков и пирамид. В этом мире функционирует «однорукий» робот Робби, способный общаться с человеком на естественном (английском) языке и совершать манипуляции с объектами этого мира. Этот робот также способен понимать достаточно сложные выражения и самостоятельно планировать свою деятельность и объяснять мотивы тех или иных действий.

Вот небольшой пример - фрагмент реального диалога с этой системой:

- Найди блок, который выше, чем тот, что в твоей руке, и положи его в коробку.
- ОК.
- Поставь, пожалуйста, один из красных блоков на другой, а сверху либо зеленый блок, либо пирамиду.
- ОК.
- Служит ли самая короткая вещь, которая опирается на опору самой высокой пирамиды, опорой для чего-то еще?
- Да, для зеленой пирамиды.
- Когда ты брался за зеленую пирамиду?
- В то время, когда я строил башенку из красного куба, большого красного блока и большого зеленого куба.
- Почему?
- Чтобы от нее избавиться.
- Почему?

- Чтобы освободить красный куб.

Многократное повторение вопроса «Почему?» в итоге приведет к ответу: «Потому что вы меня об этом попросили». Самым интересным здесь является то, что программа не отвечает на заранее заданные типовые фразы. SHRDLU, можно сказать, «понимает», о чем ее спрашивают, и отвечает достаточно разумно. Сам алгоритм Винограда настолько элегантен, что занимает всего несколько сотен строк кода на языке LISP, любимом языке разработчиков *ИИ*, занимающихся анализом ЕЯ. Этот пример с роботом Робби весьма показателен и мы будем обращаться к нему в разных лекциях.

Надо отметить, что даже для английского языка, который служит основой для всех современных языков программирования в силу своей лаконичности и достаточно формальной семантики, до сего дня не удалось создать более-менее эффективную программную систему, способную адекватно понимать СМЫСЛ фраз из достаточно больших областей *знаний*, например, нашего обыденного мира.

В разборе и *понимании естественного русского языка* массу проблем создает сложная падежная система, склонения, времена, отсутствие формального порядка следования членов предложения. Тем не менее российскими учеными созданы эффективные системы разбора фраз ограниченного естественного языка (ОЕЯ) [13], [14], [15].

Игровые программы.

К числу первых *игровых программ* можно отнести программу Артура Самуэля по игре в чекерс (американские шашки), написанную в 1947 году, причем в ней использовался ряд основополагающих идей *ИИ*, таких, как перебор вариантов и самообучение.

Научить компьютер играть в шахматы - одна из интереснейших задач в сфере *игровых программ*, использующих методы *ИИ*. Она была поставлена уже на заре вычислительной техники, в конце 50-х годов. В шахматах существуют определенные уровни мастерства, степени качества игры, которые могут дать четкие критерии интеллектуального роста машины. Поэтому компьютерными шахматами активно занимались ученые умы во всем мире. Но шахматы - игра, соревнование, и чтобы продемонстрировать свои логические способности, компьютеру необходим непосредственный противник. В 1974 году впервые прошел чемпионат мира среди шахматных программ в рамках очередного конгресса IFIP (International Federation of Information Processing) в Стокгольме. Победителем этого состязания стала советская шахматная программа «Каисса» (Каисса - богиня, покровительница шахмат). Эта программа была создана в Москве, в Институте проблем управления Академии наук в команде разработчиков программы-чемпиона, лидерами которой были Владимир Арлазаров, Михаил Донской и Георгий Адельсон-Вельский. «Каисса» показала всему миру способности русских специалистов в области эвристического программирования.

Машинное творчество.

В 1957 году американские исследователи М. Мэтьюз и Н. Гутман посетили концерт одного малоизвестного пианиста. Концерт им обоим не понравился, и, придя домой, М. Мэтьюз тут же стал писать программу, играющую музыку. Идея Мэтьюза, развиваясь, породила целый класс музыкальных языков программирования, которые вначале назывались MUSIC с номером версии. Язык C-Sound произошел как раз из этих программ. А отделение Стэндфордского института исследований, где работал тогда М. Мэтьюз, выросло в музыкальный исследовательский центр под названием

CCRMA.

В 1959 году советский математик Рудольф Зарипов начал «сочинять» одноголосные музыкальные пьесы на машине «Урал» [16]. Они назывались «Уральские напевы» и носили характер эксперимента. При их сочинении использовались случайные процессы для различных элементов музыкальной фактуры (форма, ритм, звуковысотность и т. д.). С тех пор появилось очень много программ для алгоритмической композиции. Для различных музыкальных задач было создано специальное программное обеспечение: системы многоканального сведения; системы обработки звука; системы синтеза звука; системы интерактивной композиции; программы алгоритмической композиции и др.

В 1975-1976 годах были проведены эксперименты по сравнению машинной и «человеческой» музыки. Для эксперимента были выбраны мелодии песен известных советских композиторов, опубликованные в сборниках избранных песен, и мелодии, сочиненные на вычислительной машине «Урал-2» по программе Р. Зарипова. Результаты экспериментов таковы: машинные сочинения жюри признало в большинстве случаев наиболее интересными и, «без сомнения, написанными человеком». Таким образом, деятельность машины удовлетворяла критерию Тьюринга - слушатели-эксперты не узнали ее.

Д. А. Поспелов в своем интервью «Литературной газете» [№1, 1976] слегка иронизирует над методом Р. Зарипова, вспоминая, что примерно такой же способ «творчества» предложил еще Остап Бендер в «Золотом теленке», продав журналисту Ухудшанскому свое «Незаменимое пособие для сочинения юбилейных статей, табельных фельетонов, а также парадных стихотворений, од и тропарей», избавляющее от «необходимости ждать, покуда вас окатит потный вал вдохновенья». Из раздела первого (словарь) берутся нужные существительные, прилагательные, глаголы, смешиваются по образцам раздела второго (творческая часть) и получается «шедевр». Такой метод можно запрограммировать и можно написать повести, рассказы, стихи. Но вряд ли это можно назвать творчеством. Практически очевидно, что таким образом не будет создано гениальное в общечеловеческом смысле произведение.

Не будем требовать от *интеллектуальных систем* гениальности. *ИС* уже сейчас способны делать много полезной и разумной работы, которая требует какой-то доли интеллекта.

Среди направлений работ в области *ИИ* следует также выделить **НЕЙРОКИБЕРНЕТИКУ**, или иначе говоря, подход к разработке машин, демонстрирующих «разумное» поведение, на основе архитектур, напоминающих устройство мозга и называемых **нейронными сетями** (НС). В 1942 году, когда Н. Винер определил концепции кибернетики, В. Мак-Каллок и В. Питс опубликовали первый фундаментальный труд по НС, где говорилось о том, что любое хорошо заданное отношение вход-выход может быть представлено в виде формальной НС [17]. Одна из ключевых особенностей нейронных сетей состоит в том, что они способны обучаться на основе опыта, полученного в обучающей среде. В 1957 году Ф. Розенблат изобрел устройство для распознавания на основе НС - персептрон, который успешно различал буквы алфавита, хотя и отличался высокой чувствительностью к их написанию [18].

Читателю, возможно, интересно узнать, что у рядовых муравьев и пчел примерно 80 нейронов на особь (у царицы - 200-300 нейронов), у тараканов - 300

нейронов и эти существа показывают отличные адаптационные свойства в процессе эволюции. У человека число нейронов более 10^{10}.

Пик интереса к НС приходится на 60-е и 70-е годы, но в последние десять лет наблюдается резко возросший объем исследований и разработок НС. Это стало возможным в связи с появлением нового аппаратного обеспечения, повысившего производительность вычислений в НС (нейропроцессоры, транспьютеры и т. п.). НС хорошо подходят для *распознавания образов* и решения задач классификации, оптимизации и прогнозирования. Поэтому основными областями применения НС являются:

1. промышленное производство и *робототехника*;
2. военная промышленность и аэронавтика;
3. банки и страховые компании;
4. службы безопасности;
5. биомедицинская промышленность;
6. телевидение и связь; и другие области.

Заканчивая *исторический обзор* работ в области *ИИ*, следует вернуться в 1981 год. В это время японские специалисты, объединившие свои усилия под эгидой научно-исследовательского центра по обработке информации JIPDEC, опубликовали программу НИОКР с целью создания к 1991 году прототипа ЭВМ нового поколения. Эта программа, получившая на Западе название «японский вызов», была представлена как попытка построить интеллектуальный компьютер, к которому можно было бы обращаться на естественном языке и вести беседу.

Серьезность, с которой основные конкуренты Японии откликнулись на брошенный им вызов, объясняется тем, что прежде переход от одного поколения к другому характеризовался изменением элементной базы, ростом производительности и расширением сервисных возможностей для пользователей, владеющих в той или иной мере профессиональными навыками программирования. Переход к ЭВМ пятого поколения означал резкий рост «интеллектуальных» способностей компьютера и возможность диалога между компьютером и непрофессиональным пользователем на естественном языке, в том числе в речевой форме или путем обмена графической информацией - с помощью чертежей, схем, графиков, рисунков. В состав ЭВМ пятого поколения также должна войти система решения задач и логического мышления, обеспечивающая способность машины к самообучению, ассоциативной обработке информации и получению логических выводов. Уровень «дружелюбия» ЭВМ по отношению к пользователю повысится настолько, что специалист из любой предметной области, не имеющий навыков работы с компьютером, сможет пользоваться ЭВМ при помощи естественных для человека средств общения - речи, рукописного текста, изображений и образов.

В литературе того времени [19] достаточно подробно описываются все эти вопросы. Здесь отметим только основные компоненты программного обеспечения (ПО), планируемые для систем пятого поколения:

- базовая программная система, включающая систему управления базой *знаний* (СУБЗ), систему приобретения и представления *знаний*, систему решения задач и получения выводов, систему обучения и объяснения решений;
- базовая прикладная система, включающая *интеллектуальную систему* автоматизированного проектирования (САПР) сверхбольших интегральных схем (СБИС) и архитектур ЭВМ, *интеллектуальную систему*

программирования, систему *машинного перевода* и понимания ЕЯ, систему *распознавания образов* и обработки изображений (не менее 100 000 единиц информации в виде изображений), систему распознавания речи (не менее 10 000 слов), базы *знаний* (БЗ) о предметных областях, а также утилитные системы для ввода программ и данных, обеспечивающие диагностику и обслуживание.

Теперь с позиции нашего времени можно сказать, что фирма Microsoft постаралась частично ответить на «японский вызов» в своих версиях операционной системы Windows для персональных компьютеров серии IBM PC AT/486 и выше. Уровень «дружелюбия» ЭВМ пятого поколения по отношению к пользователю действительно значительно повысился по сравнению с другими поколениями ЭВМ. В эти же годы стремительное развитие Internet стало мощным шагом по пути создания распределенных баз *знаний*.

1.6. Кратко о развитии робототехники

Развитие *робототехники* относится к глубокой древности человеческой деятельности. Еще во времена Гомера люди мечтали создать механических помощников человека, выполняющих его трудовую деятельность. Гомер пишет в своем известном произведении «Илиада»

...Навстречу ему золотые служанки вмиг подбежали,
Подобные девам живым, у которых
Разум в груди заключен, и голос, и сила,
Которых самым различным трудам обучили
Бессмертные боги...

Первыми помощниками человека были механизмы, позволяющие увеличить его силу и скорость перемещения. Даже первые счетные машины строились на механическом принципе. Однако впервые слово «робот» было введено Карелом Чапеком в 1920 г. в фантастической пьесе «РУР» («Рассумские универсальные роботы»). Областью применения роботов стали области деятельности человека, опасные для его жизнедеятельности. Как правило, это были дистанционно управляемые манипуляторы для работы в атомных реакторах, в подводных аппаратах и космических кораблях. В 1947 году в Арагонской национальной лаборатории были впервые разработаны механические руки для работы с радиоактивными материалами [20]. Уже в 1948 году данные роботы были оснащены системой отражения усилия, чтобы оператор имел возможность ощущать усилие, развиваемое исполнительным органом [21]. Первые *луноходы* и *марсоходы* были оснащены манипуляторами для сбора грунта. Управление данными манипуляторами осуществлялось с земли по командам оператора. В 1963 году уже была исследована проблема распознавания многогранных объектов, а в 1968 году уже были созданы программные устройства, позволяющие с применением телевизионной камеры находить предметы, которые должен был взять робот своим захватным устройством [22].

Таким образом, теоретические основы современной *робототехники* были заложены еще в 60-е годы, но их реализация сдерживалась отсутствием соответствующих технологий, материалов, ресурсов вычислительных систем. В это же время писатель-фантаст Айзек Азимов придумывает слово «роботикс» (*робототехника*) и впервые формулирует три закона *робототехники*:

1. Робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинен вред.
2. Робот должен подчинятся командам человека, если эти команды не противоречат первому закону.

3. Робот должен заботиться о своей безопасности, пока это не противоречит первому и второму закону.

Эти три закона Айзека Азимова до сегодняшнего дня остаются стандартами при проектировании и разработке роботов.

Робототехника XX века характеризуется выдающимися практическими достижениями.

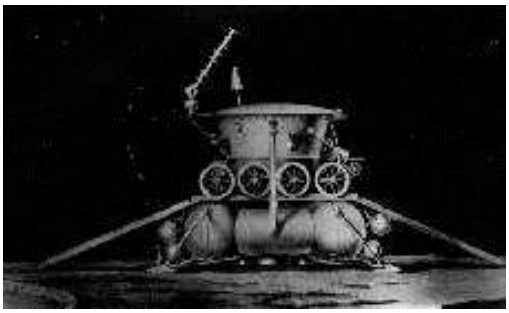


Рис. 1.3. Луноход1

1. **Советские луноходы покорили Луну.** 17 ноября 1970 года *Луноход*-1 (аппарат 8ЕЛ, вес 756 кг, длина с открытой крышкой солнечной батареи 4,42 м, ширина 2,15 м, высота 1,92 м) съехал с посадочной ступени на лунный грунт в Море Дождей (рис. 1.3). Он стал пятым подвижным образованием на Луне после Армстронга, Олдрина, Конрада и Бина. *Луноход*-1 активно функционировал 301 сутки 06 час 37 мин, прошел расстояние 10 540 м, обследовал площадь в 80 000 м^2, с помощью телесистем передал свыше 20 000 снимков поверхности и более 200 панорам, более чем в 500 точках поверхности определил физико-механические свойства поверхностного слоя лунного грунта, а в 25 точках провел его химический анализ. *Луноход*-2 в составе станции Е-8 № 204 (Eyna-21) был запущен 8 января 1973 года. Последнее сообщение ТАСС о движении аппарата было датировано 9 мая. Говорилось, что *луноход* начал движение от разлома Прямой на восток к мысу Дальний. Судя по всему, в этот день было пройдено лишь 800 м. Там *луноход* и остался. Погубил его кратер. *Луноход*-2 смог превысить отпущенные ему ресурсом три месяца.

2. **Два витка вокруг Земли и автоматическая посадка беспилотного орбитального корабля «Буран»**, выведенного в конце 1988 года на околоземную орбиту с помощью самой мощной в мире ракеты-носителя «Энергия» - это «заключительный аккорд» российской космонавтики на финише советской эпохи. Больше всего восторгов вызвало приземление «Бурана» в конце полета на посадочной полосе, выполненное с ювелирной точностью.

3. **Промышленные роботы.** Широкое внедрение роботов в производственной сфере началось в семидесятые годы прошлого столетия. В сфере производства применялись *промышленные роботы*, управляемые автоматически от систем числового программного управления. Выполнение транспортных операций при штамповке, точечная и дуговая сварка выполнялись с помощью роботов с позиционной и контурной системами управления. Уже на операциях дуговой сварки нашли применение датчики слежения за свариваемым стыком. Применение элементов адаптации позволило расширить возможности *промышленных роботов*. Особое место занимают *промышленные роботы* на сборочных операциях, особенно, при сборке элементов электронной промышленности. Оптические датчики контроля позволили выполнять сортировку изделий по этикеткам либо особым меткам. С помощью

силовой обратной связи Г. Иноу удалось создать систему управления *промышленного робота*, способного вставлять вал в отверстие по информации о развиваемом усилии при касании [23].

В настоящее время существует множество работающих *промышленных роботов*. Фирмы ABB, STAUBLI, REIS, MOTOMAN, ADEPT и другие производят *промышленных роботов* для манипулирования, сварки, покраски, упаковки, шлифовки, полировки и т. д. с большим спектром применения и по точности, и по характеру выполняемых операций.

В области *робототехники* также происходит смена поколений. В книге И. М. Макарова и Ю. И. Топчиева [24] выделяются *4 поколения промышленных роботов*:

1. Роботы с циклическим управлением без обратной связи, выполняющие неоднократно одинаковые операции.
2. Роботы с обратной связью, выполняющие разные операции.
3. Обучаемые роботы. Обучение таких роботов движению по разным траекториям и различным захватам осуществляет оператор.
4. Интеллектуальные роботы. Такие роботы могут находить нужные детали, оценивать обстановку и принимать наилучшие решения.

4. **Достижения среди роботов в общепринятом понимании**, подразумевающем: «Машина с антропоморфным (человекоподобным) поведением», которая частично или полностью выполняет функции человека при взаимодействии с окружающим миром. Из них отметим следующие.

В 1977 году фирмой Quasar Industries создан робот, умеющий подметать пол, стричь траву на лужайках и готовить простую пищу. Корпорация Object Recognition Systems объявила в 1982 году о создании системы зрения для роботов, которая позволяет им вынимать детали, произвольно расположенные в ящиках или других емкостях [25]. В 1982 году фирма Mitsubishi объявила о роботе, который был настолько ловок, что прикуривал сигарету и снимал телефонную трубку. Самым замечательным в 1982 году был признан американский робот Cubot, собирающий с помощью своих механических пальцев, камеры-глаза и компьютера-мозга кубик Рубика менее чем за четыре минуты.

Появление первых роботов дало мощный толчок к развитию таких направлений, как машинное зрение и *распознавание изображений*, построение методов моделирования **состояний мира**, построение **планов** для последовательности действий и управление выполнением этих планов, управление работой роботов в трехмерном пространстве. Интеллект роботов постоянно повышается с созданием более совершенных человеко-машинных интерфейсов. Существенно расширяется диапазон их применения.

Японская корпорация Sony объявила в 2000 году о создании нового поколения роботов-собак, которые понимают на слух около 50 команд и даже могут фотографировать то, что видят своими глазами-камерами. Новый робот получил то же ласковое имя «Айбо», что и первое поколение умных электронных собачек, появившихся на рынке годом раньше. К умению прыгать, бегать, вилять хвостиком, катать мячик и демонстрировать различные чувства - от страха до щенячьей радости, четвероногий робот нового поколения добавил способность реагировать на кличку, которую присваивает ему хозяин, подавать лапу, садиться и бежать вперед. По особому указанию он фотографирует глазами-камерами и полученную картинку потом можно посмотреть на экране компьютера. Новый «Айбо», больше похожий на львенка,

чем на щенка, стоит 150 тысяч иен (около 1,4 тыс. долл.).

В апреле 2003 года в Японии, в городе Иокогамае, прошла четвертая по счету выставка роботов «Robodex» (рис. 1.4). Как заведено, выставляются на ней так называемые персональные железяки: роботы-домохозяйки, роботы-клоуны и роботы-охранники. Аббревиатура в названии мероприятия расшифровывается ни много ни мало как «робот твоей мечты» (Robot Dream Exposition). Гвоздем выставки стал робот SDR-4X фирмы Sony. Создатели стараются сохранить за ним репутацию массовика-затейника: в новую модель заложены 10 песен, 1000 телодвижений и 200 интерактивных диалогов. Неясным остается вопрос: кто будет платить за него баснословную цену «машины класса люкс».

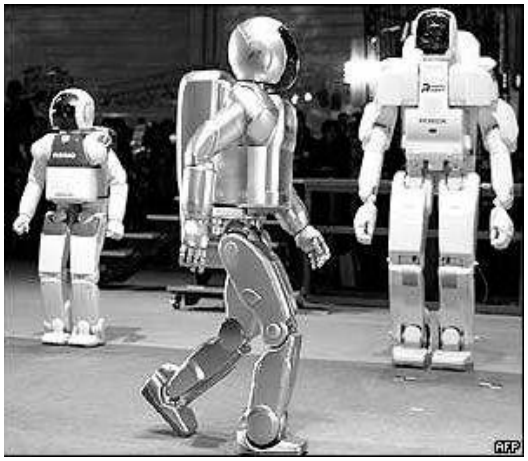


Рис. 1.4. На выставке «Robodex»

В Японии проводятся ежегодные чемпионаты мира по футболу среди роботов - RoboCup. *Соревнования* проводятся в нескольких лигах. В лиге малых роботов (small size) играют машины размером 15 × 18 сантиметров, которые управляются внешней компьютерной системой. В играх в лиге средних роботов (middle size) участвуют более мощные автономные роботы размером 50 × 50 сантиметров, оснащенные собственным мощным бортовым компьютером и системой технического зрения. С недавних пор введена еще одна лига, в ней играют робособаки, которых производит компания Sony. В 2002 году своеобразное *соревнование* проходило среди «андроидов». Правда, настоящего футбола в их исполнении увидеть не удалось: технология ходьбы проработана пока довольно слабо, так что «андроиды» соревновались в пробивании штрафных и умении ходить.

Международные *соревнования мобильных роботов*, в том числе по футболу, и Научно-технический Фестиваль молодежи «Мобильные роботы» имени профессора Е. А. Девянина проводятся в Москве на базе Института механики Московского государственного университета им. М. В. Ломоносова, начиная с 1998 года (http://www.robot.ru). Молодежная команда Московского Государственного Университета участвует в международных *соревнованиях робототехнических систем* с 1995 года. Выступления команды МГУ во Франции в рамках международного Фестиваля «Наук и технологий» были успешными: в 1996 и 1998 годах команда занимала первые места. В разработке роботов и подготовке молодежных команд с 1995 года участвовали Д. Е. Охоцимский, В. М. Буданов, Е. В. Гурфинкель, Е. А. Девянин, Д. Н. Жихарев, А. В. Ленский, с 1997 года - А. А. Голован и А. А. Гришин.

В 2004 году прошли гонки автомобилей без водителей Grand Challenge от Лос-Анджелеса до Лас-Вегаса - это одно из значительных событий в *робототехнике*. К участию в *соревновании* допускались только беспилотные роботы - на их борту не должно быть ни людей, ни животных. На участие в *соревнованиях* было заявлено

около сотни команд, 25 из них были допущены к квалификационному отбору и 15 из них этот отбор прошли. Организаторы *соревнований* остались довольны результатами, несмотря на то, что ни один робот не прошел трассу. Следующая попытка назначена на 2006 год.

За последние несколько лет Пентагон значительно увеличил финансирование проектов по созданию боевых роботов. Деньги выделяются как крупным оборонным корпорациям, так и небольшим исследовательским группам в американских университетах. Причиной такой активности военного ведомства США является негативная реакция американского общества на большое количество жертв среди солдат во время военных операций Пентагона за рубежом.

В апреле 2004 года американский производитель роботов iRobot Corporation получил первую похоронку - во время боевых действий в Ираке был разрушен робот-сапер PackBot. Представители компании iRobot, базирующейся в городе Берлингтон, штат Массачусетс, получили от Пентагона официальное сообщение о том, что робот PackBot был уничтожен противником во время боевых действий (робот взорвался на мине, от которой мог пострадать человек). В настоящее время в Ираке и Афганистане находятся от 50 до 100 роботов-саперов типа PackBot. Их используют для рекогносцировки, ликвидации минных полей, уничтожения боеприпасов противника. Эта модель приспособлена к действиям в условиях сложного ландшафта. Каждый их этих роботов весит около 21 кг и стоит почти 50 тыс. долл.

Американские *марсоходы* Spirit и Opportunity провели в 2004 году научную миссию по исследованию Красной планеты. Оба аппарата исследовали метеоритные кратеры, вели поиск интересных объектов для подробного изучения, обнаружили свидетельства наличия воды на Марсе.

Перечень удивительных достижений в области *робототехники* можно продолжать очень долго. Появляется большое количество научно-технической литературы по *робототехнике* для специалистов и студентов, как построить робот, начиная от механики, датчиков и заканчивая радиоуправлением и программированием. Вот лишь немногие из этих книг [26], [27], [28], [29], [30]. Все это подтверждает уверенность в том, что самые интересные достижения в этой области еще впереди.

1.7. Области коммерческого использования искусственного интеллекта

Интеллектуальная информационная система (ИИС) – это информационная система (ИС), которая основана на концепции использования базы знаний (БЗ) для генерации алгоритмов решения экономических задач различных классов в зависимости от конкретных информационных потребностей пользователей. *БЗ* – хранилище единиц знаний, описывающих атрибуты и действия, связанные с объектами проблемной области, а так же возможные при этом неопределенности.

К ИИС относятся: экспертные системы (ЭС), естественно-языковые, понимания речи, управления роботами, распознавания образов, нечеткие, нейронные сети, интеллектуальные агенты (инструкторы). Системы, комбинирующие две и более из перечисленных систем называются *гибридными*.

ЭС. *ЭС* – это интеллектуальная информационная система (ИИС), предназначенная для решения слабо формализуемых задач на основе накапливаемого в БЗ опыта работы экспертов в проблемной области. *Эксперт* – специалист, знания которого помещаются в БЗ. ЭС имитирует процесс рассуждений эксперта при решении сложной задачи. ЭС представляют огромный интерес для современных предприятий, так как они способны увеличить производительность и восполнить

дефицит высококвалифицированных и дорогостоящих работников – экспертов. Наиболее широкое распространение ЭС получили именно в области экономики. В основном ЭС используются при принятии решений. ЭС должна решать поставленные задачи на уровне не ниже уровня людей-экспертов в данной предметной области. Идея построения ЭС проста: формализовать знания людей-экспертов, вложить их и механизмы принятия решений в компьютерную систему и обращаться к ЭС как к консультанту. ЭС не только предложит решение, но и, если необходимо, объяснит каким образом оно получено или модифицирует его.

Мощность любой ЭС зависит прежде всего от ее БЗ, это ядро системы. Работа со знаниями включает в себя следующие этапы:

1. накопление и извлечение знаний (от экспертов, книг, отчетов и др. источников);
2. формализация знаний или построение модели знаний о данной предметной области;
3. вывод на знаниях (получение решений в результате применения определенного набора правил к знаниям из БЗ);
4. представление решения в виде, понятном пользователю – не эксперту.

ЭС и Интернет/интранет.

Интернет/интранет технологии поддерживают ЭС и наоборот. Телекоммуникационные технологии обеспечивают доступ к возможностям ЭС огромного числа пользователей, это способствует окупаемости затрат на разработку ЭС. К сожалению, только немногие ЭС доступны по сети. ЭС могут быть связаны по сети не только с пользователями, но и с другими системами, включая базы данных, системы принятия решений, управления роботами. Сетевые технологии открывают новые возможности в разработке ЭС группами людей, разделенных территориально, а так же в реализации ЭС.

1.8. ИИС других типов

ИИС могут быть интегрированы с ЭС, образуя гибридные системы, или использовать уникальные технологии.

Естественно-языковые (ЕЯ) и системы распознавания речи. Решение многих задач может упроститься, если мы сможем общаться с компьютером на нашем естественном языке. Для понимания нашей речи компьютер должен обладать достаточными лингвистическими, общими, а так же знаниями о пользователях и их целях. Области применения подобных систем: интеллектуальный интерфейс (в основном для баз данных), грамматический и смысловой анализ текста и составление рефератов, писем, перевод с одного естественного языка на другой, перевод с одного языка программирования на другой, распознавание и синтезация речи компьютером. Чем выше качество таких систем, тем выше их стоимость (например качество синтезированного голоса может быть очень высоким). Некоторые банки ввели системы обслуживания клиентов с синтезируемой речью, которые выдают информацию на запросы о балансе, наличности и т.д.

Нейронные сети. Знания не всегда удается представить в виде формализованной модели. Другим подходом к построению ИИС является имитация строения человеческого мозга, который состоит из множества (порядка 150 миллиардов) клеток – нейронов, объединенных в нейросети по несколько тысяч нейронов.

Представление знаний и обработка информации в нейрокомпьютерах базируется

на массовом распараллеливании процессов. Нейрокомпьютер состоит из искусственных нейронов (processing element), на вход каждого из них поступает входная информация от других нейронов или из окружающей среды. Входные данные описывают атрибуты некоторой проблемы (например, информация о клиенте, попросившем кредит). Каждый входной сигнал умножается на вес, который соответствует «силе» входа. Значения весов формируются на основе прошлого опыта и продолжают изменяться в зависимости от нового. Знания формируются и хранятся как множество значений весов и система обучается новым знаниям путем установления значений весов. Умноженные входные сигналы поступают на суммирующий блок в нейроне и создается выход по специальному алгоритму. Выходной сигнал(ы) может стать входным для нейрона, расположенного на следующем уровне (см. рис. 3). Каждая нейросеть формируется из трех и более уровней: входного, скрытого(ых) и выходного.

Каждый вход соответствует значению единственного атрибута (например, возраст, наличие дома и т.д.). Каждый выход (даже соответствующий качественной информации) должен быть сведен к бинарному (1 или 0). Выходной сигнал есть решение задачи (например, предоставить кредит или нет).

Хотя нейрокомпьютеры основаны на параллельных процессах, дешевле использовать обычные процессоры и имитировать распараллеливание. Хотя уже сегодня некоторые суперкомпьютеры реализуют реальные параллельные вычисления. Нейросети эффективны в ситуациях, когда нужно проанализировать большое количество данных для оценивания ситуации. Например, при принятии решения о выдаче кредита нужно просмотреть случаи из прошлого опыта с ответами да/нет.

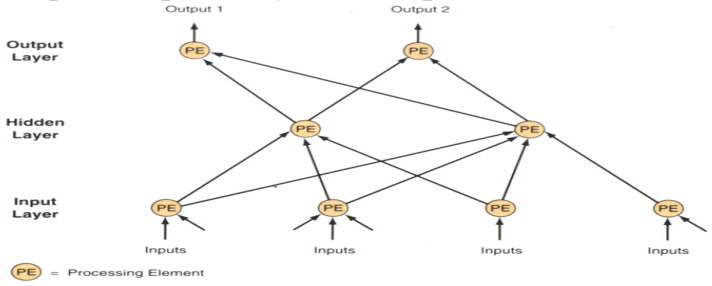


Рис. Нейронная сеть со скрытым уровнем

Области применения нейронных сетей в коммерции:

.обнаружение нарушений при уплате налогов;

.анализ рынка ценных бумаг, предсказание курсов валют;

.выдача кредитов;

.предсказание последствий того или иного решения;

.предсказание результатов продвижения на рынке новых товаров;

.управление аэролиниями: заполнение мест и составление расписания;

.оценивание кандидатов на должность;

.оптимальное распределение ресурсов;

.установление подлинности подписи и др.

Нечеткие системы. Нечеткая логика позволяет делать выводы в условиях неопределенности подобно тому, как это делает человек. Рассуждения не всегда являются «черно-белыми», только «да или нет», часто ответ «может быть» оказывается наиболее приемлемым. Применение нечеткой логики для принятия решений повышает производительность системы до 3000%. В настоящее время систем, основанных только на нечеткой логике, единицы, в основном ИИС интегрируют в себе различные технологии.

1.9. Интеллектуальные агенты

Интеллектуальные агенты (ИА), или инструкторы, представляют собой новую технологию, которая становится наиболее важной среди других информационных технологий в XXI веке. ИА являются решением многих проблем, например, качественный автоматический поиск информации в Интернет, облегчение поиска товаров в Сети, консультирование, выполнение рутинных, трудоемких функций работников. Основные характеристики ИА:

- автономность, способность действовать самостоятельно для достижения поставленной цели, способность возобновлять активность;
- активный отклик, опережающие корректные ответные действия;
- самостоятельность, работа ИА без постоянного мониторинга со стороны «мастера»;
- модульность, возможность использования в качестве подсистемы в других ИС;
- интерактивность, способность взаимодействовать с человеком или другими ИС (особенно важно в мульти – ИА);
- использование логического вывода, применение формализованных моделей знаний для принятия решений;
- дружественность, легкость в понимании и использовании ИА;
- способность к обучению и самообучению.

1.10. Примеры ИИС

Экспертиза загрязнений[1]. ЭС применяется в Kraft pulps mills (Канада) для анализа и диагностики загрязнений от смол. Если экспертиза не проводится вовремя, убытки могут составит до $80 миллионов. Поэтому система крайне необходима и используется на 40 канадских фабриках 20 различных компаний. Ежегодная прибыль от применения ЭС оценивается в $22 миллиона.

Система диагностики[1]. В компании General Electric (GE) более 40 лет Дэвид Смит диагностирует неисправности дизельных локомотивов и выдает рекомендации по их ремонту. Компания приняла решение о разработке ЭС, так как скоро Дэвид Смит должен уйти на пенсию. Традиционным решением было бы нанять молодых специалистов, способных перенять опыт Дэвида. Но GE желала решить проблему кардинально и не иметь зависимости от специалистов такой квалификации. Разработчики ЭС три года работали с Дэвидом Смитом и преобразовали его знания в вид, понятный компьютеру. Сейчас новая диагностическая технология позволяет быстро искать неисправности и объяснять их причину, поясняя на множестве схем и чертежей. ЭС установлена на всех железных дорогах GE и дает ощутимое повышение производительности труда.

Робот в госпитале. В большом госпитале ежедневно больным назначается около 12 000 медицинских препаратов. Если распределение лекарств по лоткам больных производить вручную, то число ошибок составляет 1%. Любая из этих ошибок может привести к смертельному исходу. Специальный робот (Automated Healthcare) доставляет медикаменты со склада и распределяет по специальным ячейкам для больных согласно назначению врача. Затем они доставляются на этаж обслуживающего персонала. Робот ведет учет и контроль выданных препаратов.

Контрольные вопросы

1. Что входит в понятие «онтологические исследования» в данной Проблемной области?
2. Что представляет собой концептуальная модель знаний?
3. Какими особенностями должна обладать концептуальная модель для представления полученных знаний деревом решений? Системой продукционных правил?
4. Какая связь между деревом решений и системой продукционных правил?
5. Как построить решатель в виде дерева решений? С чего начать?
6. Расскажите о работе решателя, представленного таблицей переходов.
7. Области применения нейронных сетей в коммерции

2. СИСТЕМЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Рассмотрены системы и модели представления знаний, такие как фреймы, исчисления предикатов, системы продукций, семантические сети, нечеткие множества.

Традиционно, *системы представления знаний (СПЗ)* для ИС используют следующие основные виды *моделей*: *фреймы*, *исчисления предикатов*, *системы продукций*, *семантические сети*, нечеткие множества. Рассмотрим эти *модели* подробно.

2.1. Фреймы

Фреймы предложены в 1975 году Марвином Минским [31].

Фрейм (рамка в переводе с англ.) - это единица представления знаний, запомненная в прошлом, детали которой могут быть изменены согласно текущей ситуации.

Фрейм представляет собой структуру данных, с помощью которых можно, например, описать обстановку в комнате или место встречи для проведения совещания. М.Минский предлагал эту *модель* для описания пространственных сцен. Однако с помощью *фреймов* можно описать ситуацию, сценарий, роль, структуру и т.д.

Фрейм отражает основные свойства объекта или явления. Структура *фрейма* записывается в виде списка свойств, называемых во *фрейме* слотами. Рассмотрим запись *фрейма* на языке FRL (Frame Representation Language) [32] - языке, похожем на LISP, но только внешне из-за наличия скобок.

Например, *фрейм* СТОЛ может быть записан в виде 3 слотов: слот НАЗНАЧЕНИЕ (purpose), слот ТИП (type) и слот ЦВЕТ (colour) следующим образом:

```
(frame СТОЛ
    (purpose (value(размещение предметов для
     деятельности рук)))
    (type (value(письменный)))
    (colour (value (коричневый))))
```

Во *фрейме* СТОЛ представлены только ДЕКЛАРАТИВНЫЕ средства для описания объекта, и такой *фрейм* носит название *фрейм*-образец. Однако существуют также *фреймы*-экземпляры, которые создаются для отображения фактических

ситуаций на основе поступающих данных и ПРОЦЕДУРАЛЬНЫХ средств (демонов), например, следующих:

IF-DEFAULT - по умолчанию
IF-NEEDED - если необходимо
IF-ADDED - если добавлено
IF-REMOVED - если удалено

Слот IS-A или AKO (A Kind Of) определяет иерархию *фреймов* в сети *фреймов*. Такая связь обеспечивает наследование свойств. Слот isa указывает на *фрейм* более высокого уровня, откуда неявно наследуются свойства аналогичных слотов.

Рассмотрим фрагмент описания из "мира блоков" (рис. 2.1) в виде *фреймов*.

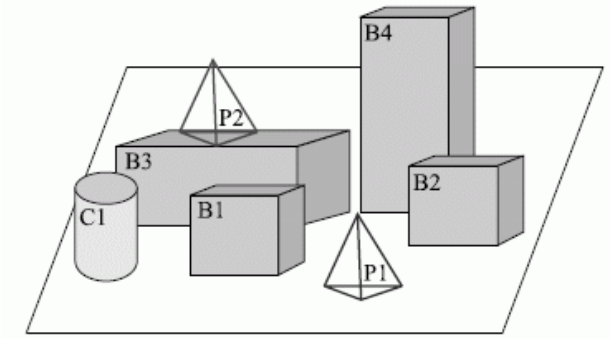


Рис. 2.1. "Мир блоков"

```
(frame (name (Cube))
   (isa (Block World))
   (length (NULL))
   (width (IF-DEFAULT (use length)))
   (height (IF-DEFAULT (use length))))
 (frame (name (B1))
   (isa (Cube))
   (color (red))
   (length (80)))
 (frame (name (B2))
   (isa (Cube))
   (color (green))
   (length (65))
   (who_put (value (NULL))
     (IF_NEEDED (askuser))))
```

Слот isa указывает на то, что объекты B1 и B2 являются подтипом объекта Cube и наследуют его свойства, а именно, length = width = height. Демон IF_NEEDED запускается автоматически, если понадобится узнать, кто поставил B2 на стол. Полученный ответ (Робби) будет подставлен в значение слота who_put. Аналогично работают демоны IF-ADDED и IF-REMOVED.

Допустим, однорукому роботу Робби дается приказ "Возьми желтый предмет, который поддерживает пирамиду". На языке представления знаний (ЯПЗ) вопрос записывается так:

```
(object ? X
   (color (yellow))
   (hold ? Y
     (type (pyramid))))
```

Программа сопоставления с образцом находит в базе знаний описание объектов:

```
(frame (name (B3))
   (type (block))
   (color (yellow))
   (size (20 20 20))
```

(coordinate (20 50 0))
(hold (P2)))

и

(frame (name (P2))
(type (pyramid))
...)

Ответ получен X = B3, Y = P2 и Робби выдается команда take(object=B3).

Таков общий механизм представления знаний в виде *фреймов*. Реализация этого механизма потребует решения других, более сложных проблем, например, автоматического ввода знаний для трехмерных объектов, работы с трехмерными быстродвижущимися объектами (своеобразный тест на реакцию) и т.д. Эти проблемы ждут своего эффективного решения.

2.2. Исчисления предикатов

Традиционная булева алгебра и исчисление высказываний [33] не всегда подходят для выражения логических рассуждений, проводимых людьми, более удобен для этого язык логики предикатов. Под ***исчислением предикатов*** понимается формальный язык для представления отношений в некоторой предметной области. *Исчисление предикатов* подробно обсуждается в ряде книг по теории ИИ [2], [7], [33]. Основное преимущество *исчисления предикатов* - хорошо понятный мощный механизм математического вывода, который может быть непосредственно запрограммирован. Дальнейшее изложение ведется с учетом того, что читатель знаком с основами булевой алгебры.

Предикатом называют предложение, принимающее только два значения: "истина" или "ложь". Для обозначения предикатов применяются логические связки между высказываниями: ¬ - не, ∨- или, ∧- и, ⊃- если, а также квантор ∃существования и квантор всеобщности ∀

∃x(...) - существует такой x, что ...

∀x(...) - для любого x

Таким образом, логика предикатов оперирует логическими связками между высказываниями, например, она решает вопросы: можно ли на основе высказывания A получить высказывание B и т.д.

Рассмотрим некоторые примеры. Высказывание "у каждого человека есть отец" можно записать:

∀x ∃y (человек(x) ⊃ отец(y,x))

Выражение "Джон владеет красной машиной" записывается, например, так:

∃x (владеет(Джон, x) ⊃ машина(x) ∧ красный(x))

Рассмотрим вывод, дающий заключение на основе двух предпосылок:

Предпосылка 1: Все люди смертны

∀x (человек(x) ⊃ смертен(x))

∀x (p(x) ⊃ q(x))

Предпосылка 2: Сократ - человек

p(a)

Заключение: Сократ - смертен

Смертен(Сократ)

q(a)

Если обозначить через f функцию одного аргумента, то логическая формула для этого высказывания будет иметь вид:

$\forall x\ (f(x) \supset q(x))$

Алфавит логики предикатов состоит из элементов (символов):

x, y, z, u, v, w - переменные;

a, b, c, d, e - константы;

f, g, h - функциональные символы;

p, q, r, s, t - предикатные символы;

$\neg$, $\vee$, $\wedge$, $\supset$, $\forall$, $\exists$- логические символы.

Запишем на языке *исчисления предикатов* некоторое выражение:

$\exists y\ \forall x$ (человек(x) $\supset$ отец(y,x))

Что означает записанное выражение? Ответ очевиден: "у всех людей общий отец".

Приведем пример простого доказательства на языке *исчисления предикатов*.

Даны следующие факты:

1. "Иван является отцом Михаила" - отец(a,b)
2. "Петр является отцом Василия" - отец(c,d)
3. "Иван и Петр являются братьями" -

$\exists w$ (брат(a,c) $\supset$ отец(w,a) $\wedge$ отец(w,c))

Даны следующие определения:

4. "Брат отца является дядей" -

$\exists y$ (дядя(x,u) $\supset$ отец(y,u) $\wedge$ брат(y,x))

5. "Сын дяди является двоюродным братом" -

$\exists x$ (дв.брат(z,u) $\supset$ дядя(x,u) $\wedge$ отец(x,z))

Требуется доказать, что "Михаил и Василий являются двоюродными братьями":

6. $\exists x\ \exists y$ (дв.брат(b,d) $\supset$отец(y,b) $\wedge$брат (y,x) $\wedge$отец(x,d))

Делаем подстановки y = Иван, b = Михаил и x = Петр, d = Василий, видим, что предикаты 1, 2, 3 дают правильное предложение 6.

Рассмотренный нами язык называется *исчислением предикатов* первого порядка и позволяет связывать знаком квантора переменные, соответствующие объектам из предметной области, но не предикаты или функции.

Исчисление предикатов второго порядка позволяет связывать знаком квантора не только переменные, соответствующие объектам из предметной области, но и предикаты или функции. Примером *исчисления предикатов* второго порядка может служить выражение "Единственное качество Джона - это честность", которое записывается так:

$\exists P$ (P(Джон) $\wedge$ качество(P) $\supset$ P = честность)

На этом мы закончим знакомство с этой *моделью* и вернемся к ней в следующей лекции при рассмотрении правил вывода, принципа резолюции и методов поиска на основе *исчисления предикатов*.

2.3. Системы продукций

Под продукцией будем понимать выражение:

Если $<X_1, X_2 \ldots X_n>$ то

$<\{Y_1, D_1\}, \ldots \{Y_m, D_m\}>$,

где: X_i, Y_i - логические выражения, D_i - фактор достоверности (0,1) или фактор уверенности (0,100).

Системы продукций - это набор правил, используемый как база знаний,

поэтому его еще называют базой правил. В Стэндфордской теории фактор уверенности CF (certainty factor) принимает значения от +1 (максимум доверия к гипотезе) до -1 (минимум доверия).

А.Ньюэлл и Г.Саймон отмечали в GPS, что продукции соответствуют навыкам решения задач человеком в долгосрочной памяти человека. Подобно навыкам в долгосрочной памяти эти продукции не изменяются при работе системы. Они вызываются по "образцу" для решения данной специфической проблемы. Рабочая память *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.

Рассмотрим пример использования *продукционных систем* для решения шахматной задачи хода конем в упрощенном варианте на доске размером 3 x 3 [2]. Требуется найти такую последовательность ходов конем, при которой он ставится на каждую клетку только один раз (рис. 2.2).

Записанные на рисунке предикаты move(x,y) составляют базу знаний (базу фактов) для задачи хода конем. Продукционные правила - это факты перемещений move, первый параметр которых определяет условие, а второй параметр определяет действие (сделать ход в поле, в которое конь может перейти). Продукционное множество правил для такой задачи приведено ниже.

P1: If (конь в поле 1) then (ход конем в поле 8)
P2: If (конь в поле 1) then (ход конем в поле 6)
P3: If (конь в поле 2) then (ход конем в поле 9)
P4: If (конь в поле 2) then (ход конем в поле 7)
P5: If (конь в поле 3) then (ход конем в поле 4)
P6: If (конь в поле 3) then (ход конем в поле 8)
P7: If (конь в поле 4) then (ход конем в поле 9)
P8: If (конь в поле 4) then (ход конем в поле 3)
P9: If (конь в поле 6) then (ход конем в поле 1)
P10: If (конь в поле 6) then (ход конем в поле 7)
P11: If (конь в поле 7) then (ход конем в поле 2)
P12: If (конь в поле 7) then (ход конем в поле 6)
P13: If (конь в поле 8) then (ход конем в поле 3)
P14: If (конь в поле 8) then (ход конем в поле 1)
P15: If (конь в поле 9) then (ход конем в поле 2)
P16: If (конь в поле 9) then (ход конем в поле 4)

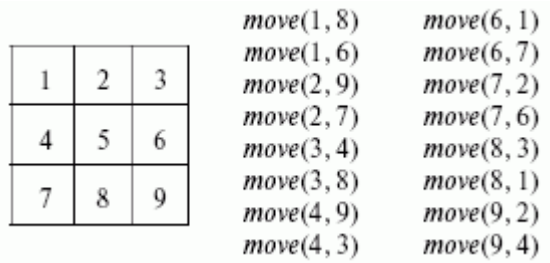


Рис. 2.2. Шахматная доска 3x3 для задачи хода конем с допустимыми ходами

Допустим, необходимо из исходного состояния (поле 1) перейти в целевое состояние (поле 2). Итерации *продукционной системы* для этого случая игры показаны в таблице 2.1.

Таблица 2.1. Итерации для задачи хода конем

№ итерации Текущее поле Целевое поле Конфликтное множество Активация правила

1 1 2 1, 2 1
2 8 2 13, 14 13
3 3 2 5, 6 5
4 4 2 7, 8 7
5 9 2 15, 16 15
6 2 2
Выход

Продукционные системы могут порождать бесконечные циклы при поиске решения. В *продукционной системе* эти циклы особенно трудно определить, потому что правила могут активизироваться в любом порядке. Например, если в 4-й итерации выбирается правило 8, мы попадаем в поле 3 и зацикливаемся. Самая простая стратегия разрешения конфликтов сводится к тому, чтобы выбирать первое соответствующее перемещение, которое ведет в еще не посещаемое состояние. Следует также отметить, что конфликтное множество это простейшая база целей. В следующей лекции мы рассмотрим различные стратегии поиска в *продукционных системах* и пути разрешения конфликтов. В заключение данного раздела лекции перечислим 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.

2.4. Семантические сети

Семантика в бытовом понимании означает смысл слова, художественного произведения, действия и т.д. ***Семантическая сеть (СС)*** - это граф, дуги которого есть отношения между вершинами (значениями). *Семантические сети* появились как *модель СПЗ* при решении задач разбора и понимания смысла естественного языка. *Модели* в виде *СС* активно развиваются в работах зарубежных и отечественных ученых, вбирая в себя важнейшие свойства других типов *моделей* [34], [35], [36], [37].

Пример *семантической сети* для предложения типа "Поставщик осуществил поставку изделий по заказу клиента до 1 июня 2004 года в количестве 1000 штук" приведен на рис. 2.3.

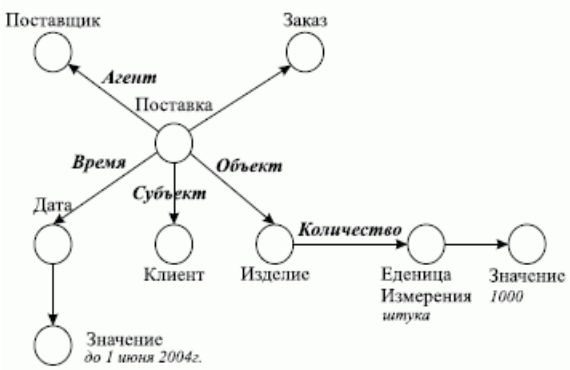


Рис. 2.3. Пример семантической сети

На этом примере видно, что между объектами Поставщик и Поставка определено отношение "агент", между объектами Изделие и Поставка определено отношение "объект" и т.д.

Число отношений, используемых в конкретных *семантических сетях*, может быть самое разное. К.Филмор, один из первых поборников идеи семантических падежей при разборе предложений, проводил свои рассуждения, пользуясь дюжиной отношений [34]. Неполный список возможных отношений, используемых в *семантических сетях* для разбора предложений, выглядит следующим образом [5].

Агент - это то, что (тот, кто) вызывает действие. Агент часто является подлежащим в предложении, например, "**Робби** ударил мяч".

Объект - это то, на что (на кого) направлено действие. В предложении объект часто выполняет роль прямого дополнения, например, "Робби взял желтую **пирамиду** ".

Инструмент - то средство, которое используется агентом для выполнения действия, например, "Робби открыл дверь **с помощью ключа** ".

Соагент служит как подчиненный партнер главному агенту, например, "Робби собрал кубики **с помощью Суззи**".

Пункт отправления и пункт назначения - это отправная и конечная позиции при перемещении агента или объекта: "Робби перешел **из комнаты в библиотеку** ".

Траектория - перемещение от пункта отправления к пункту назначения: "Они прошли **через дверь по ступенькам на лестницу** ".

Средство доставки - то в чем или на чем происходит перемещение: "Он всегда едет домой на **метро** ".

Местоположение - то место, где произошло (происходит, будет происходить) действие, например, "Он работал **за столом** ".

Потребитель - то лицо, для которого выполняется действие: "Робби собрал кубики **для Суззи**".

Сырье - это, как правило, материал, из которого что-то сделано или состоит. Обычно сырье вводится предлогом из, например, "Робби собрал Суззи **из интегральных схем** ".

Время - указывает на момент совершения действия: "Он закончил свою работу **поздно вечером** ".

Наиболее типичный способ вывода в *семантических сетях (СС)* - это способ сопоставления частей сетевой структуры. Это видно на следующем простом примере, представленном нарис. 2.4.

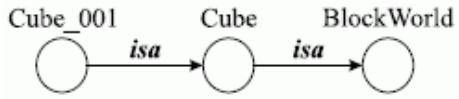


Рис. 2.4. Процедура сопоставления в СС

Куб Cube принадлежит миру BlockWorld.

Куб Cube_001 есть разновидность куба Cube.

Легко сделать вывод:

Куб Cube_001 есть часть мира BlockWorld.

Еще один пример поиска в *СС*. Представим вопрос "какой объект находится на желтом блоке?" в виде подсети, изображенной на рис. 2.5. Произведем сопоставление вопроса с сетью, представленной на рис. 2.6. В результате сопоставления получается ответ - "Пирамида".

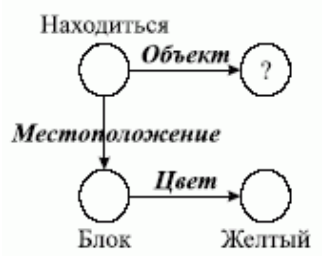


Рис. 2.5. Вопрос в виде СС

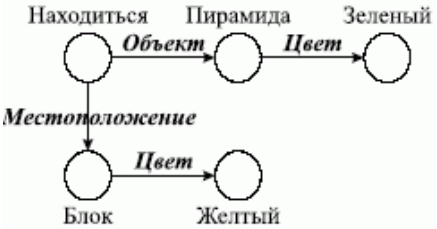


Рис. 2.6. Процедура сопоставления в СС

2.5. Нечеткая логика

При формализации знаний достаточно часто встречаются качественные знания, например, высокая температура при гриппе, слабое свечение нити накаливания, молодой дипломат и т.д. Для формального представления таких качественных знаний американский математик, профессор информатики в Университете в Беркли (Калифорния) Лофти А.Заде (Иран) предложил в 1965 году формальный аппарат нечеткой (fuzzy) логики [38].

Нечеткое подмножество N множества M определяется как множество упорядоченных пар $N = \{\mu_N(x)/x\}$, где $\mu_N(x)$ - характеристическая функция принадлежности (или просто функция принадлежности), принимающая значения в интервале [0, 1] и указывающая степень (или уровень) принадлежности элемента x подмножеству N. Таким образом, нечеткое множество N можно записать как

$$N = \sum_{i=1}^{n} (\mu(X_i) / X_i),$$

где X_i - i-е значение базовой шкалы, а знак "+" не является обозначением операции сложения, а имеет смысл объединения.

Определим лингвистическую переменную (ЛП) как переменную, значение которой определяется набором словесных характеристик некоторого свойства. Например, ЛП "возраст" может иметь значения

ЛП = МлВ, ДВ, ОВ, ЮВ, МВ, ЗВ, ПВ, СВ ,

обозначающие возраст младенческий, детский, отроческий, юношеский, молодой, зрелый, преклонный и старый, соответственно. Множество М - это шкала прожитых человеком лет [0..120]. Функция принадлежности определяет, насколько мы уверены, что данное количество прожитых лет можно отнести к данному значению ЛП. Допустим, что неким экспертом к молодому возрасту отнесены люди в возрасте 20 лет со степенью уверенности 0,8, в возрасте 25 лет со степенью уверенности 0,95, в возрасте 30 лет со степенью уверенности 0,95 и в возрасте 35 лет со степенью уверенности 0,7. Итак:

$\mu(X_1)=0,8$; $\mu(X_2)=0,95$; $\mu(X_3)=0,95$; $\mu(X_4)=0,7$;

Значение ЛП=МВ можно записать:

$$МВ = \mu(X_1) / X_1 + \mu(X_2) / X_2 + \mu(X_3) / X_3 + \mu(X_4) / X_4 =$$

$$= 0,8 / X_1 + 0,95 / X_2 + 0,95 / X_3 + 0,7 / X_4 .$$

Таким образом, нечеткие множества позволяют учитывать субъективные мнения отдельных экспертов. Для большей наглядности покажем множество МВ графически при помощи функции принадлежности (рис. 2.7).

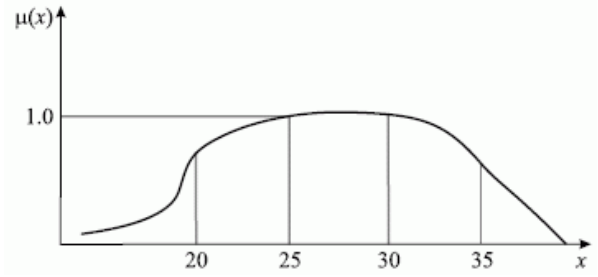


Рис. 2.7. График функции принадлежности

Для операций с нечеткими множествами существуют различные операции, например, операция "нечеткое ИЛИ" (иначе) задается в логике Заде [39], [40]:

$\mu(x)=\max(\mu_1(x), \mu_2(x))$

и при вероятностном подходе так:

$\mu(x)=\mu_1(x)+\mu_2(x)-\mu_1(x) \cdot \mu_2(x)$.

Существуют и другие операции над нечеткими числами, такие как расширенные бинарные арифметические операции (сложение, умножение и пр.) для нечетких чисел, определяемые через соответствующие операции для четких чисел с использованием принципа обобщения и т.д.

Как мы увидим в дальнейшем, нечеткие множества (другое название - мягкие вычисления) очень часто применяются в экспертных системах. Нечеткая логика применяется как удобный инструмент для управления технологическими и индустриальными процессами, для интеллектуального домашнего хозяйства и электроники развлечения, в системах обнаружения ошибок и других экспертных системах. Разработаны специальные средства нечеткого вывода, например, инструментальное средство Fuzzy CLIPS. Нечеткая логика была изобретена в Соединенных Штатах, и сейчас быстрый рост этой технологии начался в Японии, Европе и теперь снова достиг США.

Развитием этого направления является реализации в *системах представления знаний* НЕ-факторов: неполнота, неточность, недоопределенность, неоднозначность, некорректность и др. [41].

Завершая лекцию по *СПЗ*, следует отметить следующее. *Системы представления знаний* и технологии работы со знаниями продолжают развиваться. Читатель может самостоятельно познакомиться с новым языком описания

декларативных знаний (ЯОДЗ) и технологией функционально-ориентированного проектирования (ФОП-технологией) для решения информационно-сложных задач в работах [42], [43].

Кроме традиционных языков (LISP, PROLOG, SMALLTALK, РЕФАЛ) и инструментальных средств (LOOPS, KEE, ART) для представления знаний в настоящее время появляются новые веб-ориентированные версии ИС [44]. Весьма популярными стали средства на базе JAVA: системы Exsys Corvid, JESS. Язык HTML явился основой для представления знаний в среде Интернет [3]. С такими современными средствами, как система G2 и система CLIPS, читатель сможет познакомиться в лекциях 6 и 7.

Контрольные вопросы.

1. Как можно организовать интерфейс пользователя?
2. Что необходимо учесть при построении интерфейса?
3. Каким требованиям должен удовлетворять интерфейс пользователя?
4. Как организовать работу Экспертной системы, чтобы учесть ответы пользователя на вопросы ЭС?
5. Что значит «удобный» интерфейс пользователя?
6. В чем смысл нечеткой логики?
7. Что такое семантические сети?

3. МЕТОДЫ ПОИСКА РЕШЕНИЙ

Рассматриваются методы поиска решений в пространстве состояний, процедура BACKTRACK, алгоритмы эвристического поиска, алгоритм минимакса, алгоритм наискорейшего спуска, алгоритм оценочных функций, алгоритм штрафных функций, альфа-бета - процедура, поиск решений на основе исчисления предикатов, метод резолюции, поиск решений в продукционных системах.

Традиционными *методами поиска решений* в ИС считаются: методы *поиска в пространстве состояний* на основе различных *эвристических алгоритмов*, *методы поиска на основе предикатов* (*метод резолюции* и др.), поиск решений в *продукционных системах*, поиск решений в семантических сетях и т. д. Рассмотрим эти методы подробно.

3.1. Методы поиска решений в пространстве

Методы поиска решений в пространстве состояний начнем рассматривать с простой задачи о миссионерах и людоедах. Три миссионера и три людоеда находятся на левом берегу реки и им нужно переправиться на правый берег, однако у них имеется только одна лодка, в которую могут сесть лишь 2 человека. Поэтому необходимо определить план, соблюдая который и курсируя несколько раз туда и обратно, можно переправить всех шестерых. Однако если на любом берегу реки число миссионеров будет меньше, чем число людоедов, то миссионеры будут съедены. Решения принимают миссионеры, людоеды их выполняют.

Основой метода являются следующие этапы.

1. Определяется конечное число состояний, одно из состояний принимается за начальное и одно или несколько состояний определяются как искомое (конечное, или терминальное). Обозначим состояние S тройкой S=(x,y,z), где x и y - число миссионеров и людоедов на левом берегу, z= {L,R} - положение лодки на левом (L) или правом (R) берегах. Итак, начальное состояние S_0=(3,3, L) и конечное (терминальное) состояние S_k=(0,0, R).
2. Заданы правила перехода между группами состояний. Введем понятие действия M:[u, v]w, где u - число миссионеров в лодке, v - число людоедов в лодке,w - направление движения лодки (R или L).
3. Для каждого состояния заданы определенные условия допустимости (оценки) состояний: $x \geq y; 3-x \geq 3-y$; $u+v \leq 2$.
4. После этого из текущего (исходного) состояния строятся переходы в новые состояния, показанные нарис. 3.1. Два новых состояния следует сразу же вычеркнуть, так как они ведут к нарушению условий допустимости (миссионеры будут съедены).
5. При каждом переходе в новое состояние производится оценка на допустимость состояний и если при использовании правила перехода для текущего состояния получается недопустимое состояние, то производится возврат к тому предыдущему состоянию, из которого было достигнуто это текущее состояние. Эта процедура получила название бэктрекинг (bac tracing или ***BACKTRACK***).

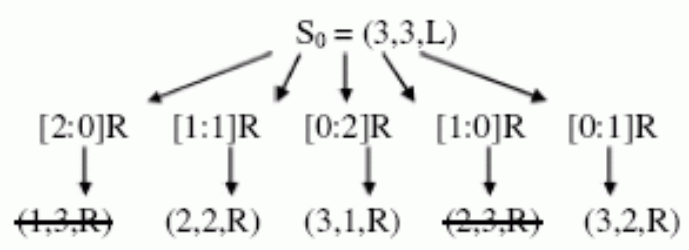


Рис. 3.1. Переходы из исходного состояния

Теперь мы можем проанализировать полностью алгоритм простейшего *поиска решений в проблемном пространстве*, описанный группами состояний и переходами между состояниями на рис. 3.2. Решение задачи выделено на рис. 3.2 жирными стрелками. Такой *метод поиска* $S_0 \Rightarrow S_k$ называется прямым *методом поиска*. Поиск $S_k \Rightarrow S_0$ называют обратным поиском. Поиск в двух направлениях одновременно называют двунаправленным поиском.

Как уже упоминалось, фундаментальным понятием в *методах поиска* в ИС является идея рекурсии и процедура *BACKTRACK*. В качестве примера многоуровневого возвращения рассмотрим задачу размещения на доске 8 x 8 восьми ферзей так, чтобы они не смогли "съесть" друг друга.

Допустим, мы находимся на шаге размещения ферзя в 6 ряду и видим, что это невозможно. Процедура *BACKTRACK* пытается переместить ферзя в 5 строке и в 6 строке опять неудача. Только возврат к 4 строке и нахождение в ней нового варианта размещения приведет к решению задачи. Читатель сам может завершить решение этой задачи на основе процедуры *BACKTRACK*.

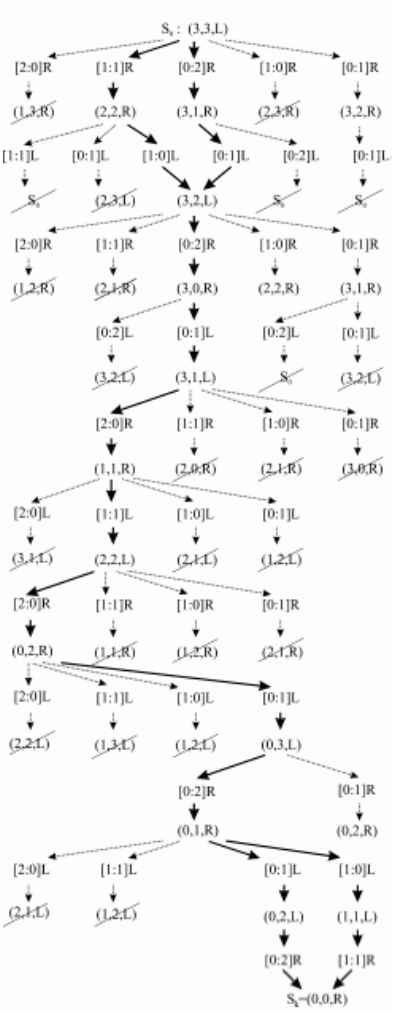
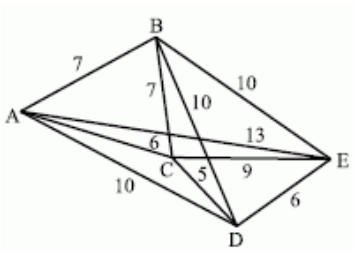


Рис. 3.2. Метод поиска в пространстве состояний

3.2. Алгоритмы эвристического поиска

В рассмотренных примерах поиска решений число состояний невелико, поэтому перебор всех возможных состояний не вызвал затруднений. Однако при значительном числе состояний время поиска возрастает экспоненциально, и в этом случае могут помочь *алгоритмы эвристического поиска*, которые обладают высокой вероятностью правильного выбора решения. Рассмотрим некоторые из этих алгоритмов.



Алгоритм наискорейшего спуска по дереву решений

Пример построения более узкого дерева рассмотрим на примере задачи о коммивояжере. Торговец должен побывать в каждом из 5 городов, обозначенных на карте (рис. 3.3).

Рис. 3.3.

Задача состоит в том, чтобы, начиная с города A, найти минимальный путь, проходящий через все остальные города только один раз и приводящий обратно в A. Идея метода исключительно проста - из каждого города идем в ближайший, где мы еще не были. Решение задачи показано на рис. 3.4.

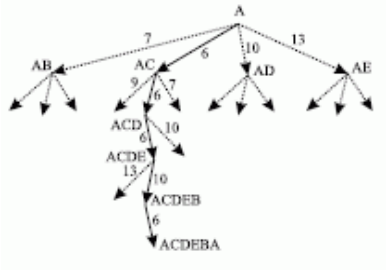


Рис. 3.4.

Такой алгоритм поиска решения получил название *алгоритма наискорейшего спуска* (в некоторых случаях - наискорейшего подъема).

Алгоритм оценочных (штрафных) функций

Умело подобранные оценочные функции (в некоторых источниках - штрафные функции) могут значительно сократить полный перебор и привести к решению достаточно быстро в сложных задачах. В нашей задаче о людоедах и миссионерах в качестве самой простой целевой функции при выборе очередного состояния можно взять число людоедов и миссионеров, находящихся "не на месте" по сравнению с их расположением в описании целевого состояния. Например, значение этой функции f=x+y для исходного состоянияf_0=6, а значение для целевого состояния f_1=0.

Эвристические процедуры поиска на графе стремятся к тому, чтобы минимизировать некоторую комбинацию стоимости пути к цели и стоимости поиска. Для задачи о людоедах введем оценочную функцию:

f(n) = d(n) + w(n)

где d(n) - глубина вершины n на дереве поиска и w(n) - число находящихся не на нужном месте миссионеров и людоедов. Эвристика заключается в выборе минимального значения f(n). Определяющим в *эвристических процедурах* является выбор оценочной функции.

Рассмотрим вопрос о сравнительных характеристиках оценочных целевых функций на примере функций для игры в "8" ("пятнашки"). Игра в "8" заключается в нахождении минимального числа перестановок при переходе из исходного состояния в конечное (терминальное, целевое).

Рассмотрим две оценочные функции:

h_1(n) & = Q(n)

h_2(n) & = P(n) + 3S(n),

где Q(n) - число фишек не на месте; P(n) - сумма расстояний каждой фишки от

места в ее целевой вершине; S(n) - учет последовательности нецентральных фишек (штраф +2 если за фишкой стоит не та, которая должна быть в правильной последовательности; штраф +1 за фишку в центре; штраф 0 в остальных случаях).

Сравнение этих оценочных функций приведено в таблица 3.1.

Таблица 3.1. Сравнение оценочных функций

Оценочная функция h	**Стоимость (длина) пути L**	**Число вершин, открытых при нахождении пути N**	**Трудоемкость вычислений, необходимых для подсчета h S**	**Примечания**
h_1 S_0 S_1	5 >18	13 100-8!(=40320)	8	Поиск в ширину
h_2 S_0 S_1	5 18	11 43	8*2+8+1+1	Поиск в глубину

На основе сравнения этих двух оценочных функций можно сделать выводы.

- Основу алгоритма поиска составляет выбор пути с минимальной оценочной функцией.
- Поиск в ширину, который дает функция h_1, гарантирует, что какой-либо путь к цели будет найден. При поиске в ширину вершины раскрываются в том же порядке, в котором они порождаются.
- Поиск в глубину управляется эвристической компонентой 3S(n) в функции h_2 и при удачном выборе оценочной функции позволяет найти решение по кратчайшему пути (по минимальному числу раскрываемых вершин). Поиск в глубину тем и характеризуется, что в нем первой раскрывается та вершина, которая была построена самой последней.
- Эффективность поиска возрастает, если при небольших глубинах он направляется в основном в глубь эвристической компонентой, а при возрастании глубины он больше похож на поиск вширь, чтобы гарантировать, что какой-либо путь к цели будет найден. Эффективность поиска можно определить как E=K/L*N*S, где K и S (трудоемкость) - зависят от оценочной функции, L - длина пути,N - число вершин, открытых при нахождении пути. Если договориться, что для оптимального пути E=1, то $K=L^0*N^0*S^0$.

Алгоритм минимакса

В 1945 году Оскар Моргенштерн и Джон фон Нейман предложили ***метод минимакса***, нашедший широкое применение в теории игр. Предположим, что противник использует оценочную функцию (ОФ), совпадающую с нашей ОФ. Выбор хода с нашей стороны определяется максимальным значением ОФ для текущей позиции. Противник стремится сделать ход, который минимизирует ОФ. Поэтому этот метод и получил название минимакса. На рис. 3.5 приведен пример анализа дерева ходов с помощью *метода минимакса* (выбранный путь решения отмечен жирной линией).

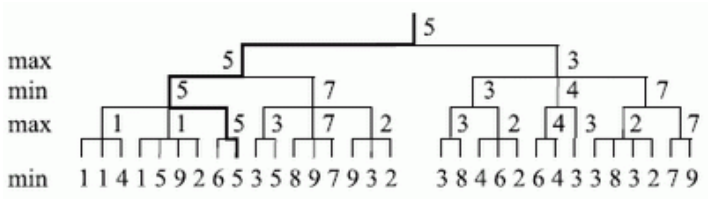


Рис. 3.5. Дерево ходов

Развивая *метод минимакса*, назначим вероятности для выполняемых действий в

задаче о миссионерах и людоедах:

P([2 : 0]R) = 0; 8; P([1 : 1]R) = 0; 5;

P([0 : 2]R) = 0; 9;

P([1 : 0]R) = 0; 3; P([0 : 1]R) = 0; 3:

Интуитивно понятно, что посылать одного людоеда или одного миссионера менее эффективно, чем двух человек, особенно на начальных этапах. На каждом уровне мы будем выбирать состояние по критерию P_i. Даже такой простой подход позволит нам избежать части тупиковых состояний в процессе поиска и сократить время по сравнению с полным перебором. Кстати, этот подход достаточно распространен в экспертных продукционных системах.

Альфа-бета-процедура

Теоретически, это эквивалентная минимаксу процедура, с помощью которой всегда получается такой же результат, но заметно быстрее, так как целые части дерева исключаются без проведения анализа. В основе этой процедуры лежит идея Дж. Маккарти об использовании двух переменных, обозначенных α и β (1961 год).

Основная идея метода состоит в сравнении наилучших оценок, полученных для полностью изученных ветвей, с наилучшими предполагаемыми оценками для оставшихся. Можно показать, что при определенных условиях некоторые вычисления являются лишними. Рассмотрим идею отсечения на примере рис. 3.6. Предположим, позиция A полностью проанализирована и найдено значение ее оценки α. Допустим, что один ход из позиции Y приводит к позиции Z, оценка которой по *методу минимакса* равна z. Предположим, что $z \le \alpha$. После анализа узла Z, когда справедливо соотношение $y \le z \le \alpha \le s$, ветви дерева, выходящие из узла Y, могут быть отброшены (альфа-отсечение).

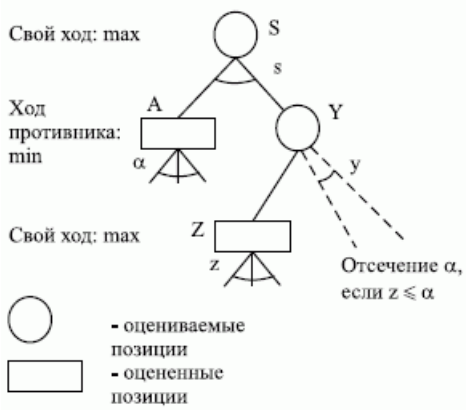


Рис. 3.6. - отсечение

Если мы захотим опуститься до узла Z, лежащего на уровне произвольной глубины, принадлежащей той же стороне, что и уровень S, то необходимо учитывать минимальное значение оценки β, получаемой на ходах противника.

Отсечение типа β можно выполнить всякий раз, когда оценка позиции, возникающая после хода противника, превышает значение β. Алгоритм поиска строится так, что оценки своих ходов и ходов противника сравниваются при анализе дерева с величинами α и β соответственно. В начале вычислений этим величинам присваиваются значения $+\infty$ и $-\infty$, а затем, по мере продвижения к корню дерева, находится оценка начальной позиции и наилучший ход для одного из противников.

Правила вычисления α и β в процессе поиска рекомендуются следующие:

1. у MAX вершины значение α равно наибольшему в данный момент значению среди окончательных возвращенных значений для ее дочерних вершин;

2. у MIN вершины значение β равно наименьшему в данный момент значению среди окончательных возвращенных значений для ее дочерних вершин.

Правила прекращения поиска:

3. можно не проводить поиска на поддереве, растущем из всякой MIN вершины, у которой значение β не превышает значения αвсех ее родительских MAX вершин;

4. можно не проводить поиска на поддереве, растущем из всякой MAX вершины, у которой значение αне меньше значения β всех ее родительских MIN вершин.

На рис. 3.7 показаны α-β отсечения для конкретного примера. Таким образом, α-β-алгоритм дает тот же результат, что и *метод минимакса*, но выполняется быстрее.

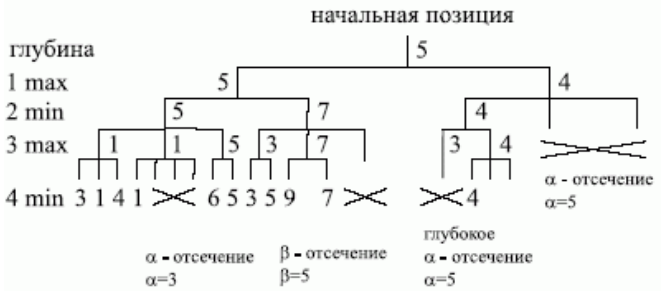


Рис. 3.7. a-b отсечение для конкретного примера

Использование *алгоритмов эвристического поиска* для поиска на графе И, ИЛИ выигрышной *стратегии* в более сложных задачах и играх (шашки, шахматы) не реален. По некоторым оценкам игровое дерево игры в шашки содержит 10^{40} вершин, в шахматах 10^{120} вершин. Если при игре в шашки для одной вершины требуется 1/3 наносекунды, то всего игрового времени потребуется 10^{21} веков. В таких случаях вводятся искусственные условия остановки, основанные на таких факторах, как наибольшая допустимая глубина вершин в дереве поиска или ограничения на время и объем памяти.

Многие из рассмотренных выше идей были использованы А. Ньюэллом, Дж. Шоу и Г. Саймоном в их программе GPS. Процесс работы GPS в общем воспроизводит *методы решения задач*, применяемые человеком: выдвигаются подцели, приближающие к решению; применяется *эвристический метод* (один, другой и т. д.), пока не будет получено решение. Попытки прекращаются, если получить решение не удается.

Программа STRIPS (STanford Research Institut Problem Solver) вырабатывает соответствующий *порядок действий робота* в зависимости от поставленной цели. Программа способна обучаться на опыте решения предыдущих задач. Большая часть игровых программ также обучается в процессе работы. Например, знаменитая шашечная программа Самюэля, выигравшая в 1974 году у чемпиона мира, "заучивала наизусть" выигранные партии и обобщала их для извлечения пользы из прошлого опыта. Программа HACHER Зуссмана, управляющая поведением робота, обучалась также и на ошибках.

3.3. Методы поиска решений на основе исчисления предикатов

Семантика исчисления предикатов обеспечивает основу для формализации логического вывода. Возможность логически выводить новые правильные выражения из набора истинных утверждений очень важна. Логически выведенные утверждения

корректны, и они совместимы со всеми предыдущими интерпретациями первоначального набора выражений. Обсудим вышесказанное неформально и затем введем необходимую формализацию.

В исчислении высказываний основным объектом является переменное высказывание (предикат), истинность или ложность которого зависит от значений входящих в него переменных. Так, истинность предиката "x был физиком" зависит от значения переменной x. Если x - П. Капица, то предикат истинен, если x - М. Лермонтов, то он ложен. На языке исчисления предикатов утверждение $\forall x(P(x) \supset Q(x))$ читается так: "для любого x если P(x), то имеет место и Q(x)". Иногда его записывают и так: $\forall x\ (P(x) \rightarrow Q(x))$. Выделенное подмножество тождественно истинных формул (или правильно построенных формул - ППФ), истинность которых не зависит от истинности входящих в них высказываний, называется аксиомами.

В исчислении предикатов имеется множество правил вывода. В качестве примера приведем классическое правило отделения, или modus ponens :

$(A, A \rightarrow B) / B$

которое читается так "если истинна формула A и истинно, что из A следует B, то истинна и формула B". Формулы, находящиеся над чертой, называются посылками вывода, а под чертой - заключением. Это правило вывода формализует основной закон дедуктивных систем: из истинных посылок всегда следуют истинные заключения. Аксиомы и правила вывода исчисления предикатов первого порядка задают основу формальной дедуктивной системы, в которой происходит формализация схемы рассуждений в логическом программировании. Можно упомянуть и другие правила вывода.

Modus tollendo tollens : Если из A следует B и B ложно, то и A ложно.

Modus ponendo tollens : Если A и B не могут одновременно быть истинными и A истинно, то B ложно.

Modus tollendo ponens : Если либо A, либо B является истинным и A не истинно, то B истинно.

Решаемая задача представляется в виде утверждений (аксиом) f_1, F_2... F_n исчисления предикатов первого порядка. Цель задачи B также записывается в виде утверждения, справедливость которого следует установить или опровергнуть на основании аксиом и правил вывода формальной системы. Тогда решение задачи (достижение цели задачи) сводится к выяснению логического следования (выводимости) целевой формулы B из заданного множества формул (аксиом) f_1, F_2... F_n. Такое выяснение равносильно доказательству общезначимости (тождественно-истинности) формулы

$f_1 \& F_2 \& ... \& F_n \rightarrow B$

или невыполнимости (тождественно-ложности) формулы

$f_1 \& F_2 \& ... \& F_n \& \neg B$

Из практических соображений удобнее использовать доказательство от противного, то есть доказывать невыполнимость формулы. На доказательстве от противного основано и ведущее правило вывода, используемое в логическом программировании, - ***принцип резолюции***. Робинсон открыл более сильное правило вывода, чем modus ponens, которое он назвал *принципом резолюции* (или правилом резолюции). При использовании *принципа резолюции* формулы исчисления предикатов с помощью несложных преобразований приводятся к так называемой дизъюнктивной форме, то есть представляются в виде набора дизъюнктов. При этом

под дизъюнктом понимается дизъюнкция литералов, каждый из которых является либо предикатом, либо отрицанием предиката.

Приведем пример дизъюнкта:

$\forall x\ (P(x, c_1) \supset Q(x, c_2))$.

Пусть P - предикат уважать, c_1 - Ключевский, Q - предикат знать,c_2 - история. Теперь данный дизъюнкт отражает факт "каждый, кто знает историю, уважает Ключевского".

Приведем еще один пример дизъюнкта:

$\forall x\ (P(x, c_1) \& P(x, c_2))$.

Пусть P - предикат знать, c_1 - физика, c_2 - история. Данный дизъюнкт отражает запрос "кто знает физику и историю одновременно".

Таким образом, условия решаемых задач (факты) и целевые утверждения задач (запросы) можно выразить в дизъюнктивной форме логики предикатов первого порядка. В дизъюнктах кванторы всеобщности $\forall$, $\exists$, обычно опускаются, а связки $\supset$, $\neg$, $\wedge$заменяются на $\rightarrow$импликацию.

Вернемся к *принципу резолюции*. Главная идея этого правила вывода заключается в проверке того, содержит ли множество дизъюнктов R пустой (ложный) дизъюнкт. Обычно резолюция применяется с прямым или обратным методом рассуждения. Прямой метод из посылок A, A $\rightarrow$B выводит заключение B (правило modus ponens). Основной недостаток прямого метода состоит в его не направленности: повторное применение метода приводит к резкому росту промежуточных заключений, не связанных с целевым заключением. Обратный вывод является направленным: из желаемого заключения B и тех же посылок он выводит новое подцелевое заключение A. Каждый шаг вывода в этом случае связан всегда с первоначально поставленной целью. Существенный недостаток *метода резолюции* заключается в формировании на каждом шаге вывода множества резольвент - новых дизъюнктов, большинство из которых оказывается лишними. В связи с этим разработаны различные модификации *принципа резолюции*, использующие более эффективные *стратегии поиска* и различного рода ограничения на вид исходных дизъюнктов. В этом смысле наиболее удачной и популярной является система ПРОЛОГ, которая использует специальные виды дизъюнктов, называемых дизъюнктами Хорна.

Процесс доказательства *методом резолюции* (от обратного) состоит из следующих этапов:

1. Предложения или аксиомы приводятся к дизъюнктивной нормальной форме.
2. К набору аксиом добавляется отрицание доказываемого утверждения в дизъюнктивной форме.
3. Выполняется совместное разрешение этих дизъюнктов, в результате чего получаются новые основанные на них дизъюнктивные выражения (резольвенты).
4. Генерируется пустое выражение, означающее противоречие.
5. Подстановки, использованные для получения пустого выражения, свидетельствуют о том, что отрицание отрицания истинно.

Рассмотрим примеры применения *методов поиска решений на основе исчисления предикатов*. Пример "интересная жизнь" заимствован из [2]. Итак, заданы утверждения 1-4 в левом столбце таблица 3.2 Требуется ответить на вопрос: "Существует ли человек, живущий интересной жизнью?" В виде предикатов эти

утверждения записаны во втором столбце таблицы. Предполагается, что ∀X(smart(X)=¬stupid(X)) и ∀Y(wealthy(Y)=¬poor(Y)). В третьем столбце таблицы записаны дизъюнкты.

Таблица 3.2. Интересная жизнь

Утверждения и заключение	Предикаты	Предложения(дизъюнкты)
1. Все небедные и умные люди счастливы	∃X(¬poor(X) ∧ smart(X) → happy(X))	poor(X) ∧ ¬smart(X) & happy(X)
2. Человек, читающий книги, - неглуп	∀Y (read(Y) → smart(Y))	¬read(Y) & smart(Y)
3. Джон умеет читать и является состоятельным человеком	read(John) ∧ ¬poor(John)	3a read(John) 3b ¬poor(John)
4. Счастливые люди живут интересной жизнью	∀Z (happy(Z)→ exciting(Z))	¬happy(Z)&exciting(Z)
5. Заключение: Существует ли человек, живущий интересной жизнью?	∃ W(exciting(W))	exciting(W)
6. Отрицание заключения	¬∃ W(exciting(W))	¬exciting(W)

Отрицание заключения имеет вид (строка 6): ¬∃W(exciting(W))

Одно из возможных доказательств (их более одного) дает следующую последовательность резольвент:

1. ¬happy(Z)резольвента 6 и 4
2. poor(X) ∧¬smart(X)резольвента 7 и 1
3. poor(Y) ∧¬read(Y)резольвента 8 и 2
4. ¬read(John)резольвента 9 и 3b
5. NILрезольвента 10 и 3a

Символ NIL означает, что база данных выражений содержит противоречие и поэтому наше предположение, что не существует человек, живущий интересной жизнью, неверно.

В *методе резолюции* порядок комбинации дизъюнктивных выражений не устанавливался. Значит, для больших задач будет наблюдаться экспоненциальный рост числа возможных комбинаций. Поэтому в процедурах резолюции большое значение имеют также эвристики поиска и различные *стратегии*. Одна из самых простых и понятных *стратегий* - *стратегия* предпочтения единичного выражения, которая гарантирует, что резольвента будет меньше, чем наибольшее родительское выражение. Ведь в итоге мы должны получить выражение, не содержащее литералов вообще.

Среди других *стратегий* (поиск в ширину (breadth-first), *стратегия* "множества поддержки", *стратегия* линейной входной формы) *стратегия* "множества поддержки" показывает отличные результаты при поиске в больших пространствах дизъюнктивных выражений [2]. Суть *стратегии* такова. Для некоторого набора исходных дизъюнктивных выражений S можно указать подмножество T, называемое множеством поддержки. Для реализации этой *стратегии* необходимо, чтобы одна из резольвент в каждом опровержении имела предка из множества поддержки. Можно

доказать, что если S - невыполнимый набор дизъюнктов, а S-T - выполнимый, то *стратегия* множества поддержки является полной в смысле опровержения. С другими *стратегиями для поиска методом резолюции* в больших пространствах дизъюнктивных выражений читатель может познакомиться в специальной литературе [2], [45], [46].

Исследования, связанные с доказательством теорем и разработкой алгоритмов опровержения резолюции, привели к развитию языка логического программирования PROLOG (Programming in Logic). PROLOG основан на теории предикатов первого порядка. Логическая программа - это набор спецификаций в рамках формальной логики. Несмотря на то, что в настоящее время удельный вес языков LISP и PROLOG снизился и при решении задач ИИ все больше используются C, C++ и Java, однако многие задачи и разработка новых *методов решения задач* ИИ продолжают опираться на языки LISP и PROLOG. Рассмотрим одну из таких задач - задачу *планирования последовательности действий* и ее решение на основе теории предикатов.

3.4. Задачи планирования последовательности действий

Многие результаты в области ИИ достигнуты при решении " задач для робота ". Одной из таких простых в постановке и интуитивно понятных задач является задача *планирования последовательности действий*, или задача построения планов.

В наших рассуждениях будут использованы примеры традиционной робототехники (современная робототехника во многом основывается на реактивном управлении, а не на планировании). Пункты плана определяют атомарные действия для робота. Однако при описании плана нет необходимости опускаться до микроуровня и говорить о датчиках, шаговых двигателях и т. п. Рассмотрим ряд предикатов, необходимых для работы планировщика из мира блоков. Имеется некоторый робот, являющийся подвижной рукой, способной брать и перемещать кубики. Рука робота может выполнять следующие задания (U, V, W, X, Y, Z - переменные).

goto(X,Y,Z)перейти в местоположение X,Y,Z
pickup(W)взять блок W и держать его
putdown(W)опустить блок W в некоторой точке
stack(U,V)поместить блок U на верхнюю грань блока V
unstack(U,V)убрать блок U с верхней грани блока V

Состояния мира описываются следующим множеством предикатов и отношений между ними.

on(X,Y)блок X находится на верхней грани блока Y
clear(X)верхняя грань блока X пуста
gripping(X)захват робота удерживает блок X
gripping()захват робота пуст
ontable(W)блок W находится на столе

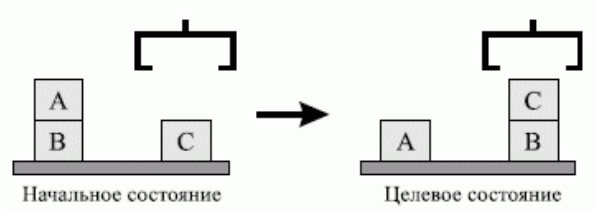


Рис. 3.8. Начальное и целевое состояния задачи из мира кубиков

Предметная область из мира кубиков представлена на рис. 3.8 в виде начального и целевого состояния для решения задачи *планирования*. Требуется построить *последовательность действий робота*, ведущую (при ее реализации) к достижению целевого состояния.

Состояния мира кубиков представим в виде предикатов. Начальное состояние

можно описать следующим образом:

start = [handempty, ontable(b),
ontable(c), on(a,b), clear(c),
clear(a)]

где: handempty означает, что рука робота Робби пуста.

Целевое состояние записывается так:

goal = [handempty, ontable(a),
ontable(b), on(c,b), clear(a),
clear(c)]

Теперь запишем правила, воздействующие на состояния и приводящие к новым состояниям.

(∀X) (pickup(X) ⟶ (gripping(X) ←
(gripping() ∧ clear(X) ∧ ontable(X))))

(∀X) (putdown(X) ⟶ ((gripping() ∧
ontable(X) ∧ clear(X)) ← gripping(X)))

(∀X) (∀Y) (stack(X,Y) ⟶
((on(X,Y) ∧ gripping() ∧ clear(X)) ←
(clear(Y) ∧ gripping(X))))

(∀X) (∀Y) (unstack(X,Y) ⟶
((clear(Y) ∧ gripping(X)) ←
(on(X,Y) ∧ clear(X) ∧ gripping())))

Прежде чем использовать эти правила, необходимо упомянуть о проблеме границ. При выполнении некоторого действия могут изменяться другие предикаты и для этого могут использоваться аксиомы границ - правила, определяющие инвариантные предикаты. Одно из решений этой проблемы предложено в системе STRIPS.

В начале 1970-х годов в Стэнфордском исследовательском институте (Stanford Research Institute Planning System) была создана система STRIPS для управления роботом. В STRIPS четыре оператора pickup, putdown, stack,unstack описываются тройками элементов. Первый элемент тройки - множество предусловий (П), которым удовлетворяет мир до применения оператора. Второй элемент тройки - список дополнений (Д), которые являются результатом применения оператора. Третий элемент тройки - список вычеркиваний (В), состоящий из выражений, которые удаляются из описания состояния после применения оператора.

Ведя рассуждения для рассматриваемого примера от начального состояния, мы приходим к *поиску в пространстве состояний*. Требуемая последовательность действий (план достижения цели) будет следующей:

unstack(A,B), putdown(A), pickup(C), stack(C,B)

Для больших графов (сотни состояний) поиск следует проводить с использованием оценочных функций. Более подробно о работах по *планированию*, в том числе современные публикации по адаптивному *планированию*, можно прочитать в литературе [7], [47], [48], [49], [50].

В качестве заключения по данному разделу лекции следует сказать, что планирование достижения цели можно рассматривать как *поиск в пространстве состояний*. Для нахождения пути из начального состояния к целевому (*плана последовательности действий робота*) могут применяться методы *поиска в пространстве состояний* с использованием исчисления предикатов.

3.5. Поиск решений в системах продукций

Поиск решений в системах продукций наталкивается на проблемы выбора правил из *конфликтного множества*, как это указывалось в предыдущей лекции.

Различные варианты решения этой проблемы рассмотрим на примере ЭСО CLIPS, на которой нам предстоит в 7 лекции разработать исследовательский прототип ЭС. Правила в ЭС, кроме фактора уверенности эксперта, имеют приоритет выполнения (salience). ***Конфликтное множество (КМ)*** - это список всех правил, имеющих удовлетворенные условия при некотором, текущем состоянии списка фактов и объектов и которые еще не были выполнены. Как отмечалось ранее, *конфликтное множество* это простейшая база целей. Когда активизируется новое правило с определенным приоритетом, оно помещается в список правил *КМ* ниже всех правил с большим приоритетом и выше всех правил с меньшим приоритетом. Правила с высшим приоритетом выполняются в первую очередь. Среди правил с одинаковым приоритетом используется определенная *стратегия*.

CLIPS поддерживает семь *стратегий* разрешения конфликтов.

*Стратегия*глубины (depth strategy) является *стратегией* по умолчанию (default strategy) в CLIPS. Только что активизированное правило помещается поверх всех правил с таким же приоритетом. Это позволяет реализовать поиск в глубину.

*Стратегия*ширины (breadth strategy) - только что активизированное правило помещается ниже всех правил с таким же приоритетом. Это, в свою очередь, реализует поиск в ширину.

LEX *стратегия* - активация правила, выполненная более новыми образцами (фактами), располагается перед активацией, осуществленной более поздними образцами. Например, как это указано в табл.3.3.

MEA *стратегия* - сортировка образцов не производится, а осуществляется только упорядочение правил по первым образцам, как это показано в столбце 3 таблица 3.3.

Таблица 3.3. Результаты применения LEX и MEA стратегий

Исходный набор правил	Правила, отсортированные LEX	Правила, отсортированные MEA
rule-6: f-1,f-4	rule-6: f-4,f-1	rule-2: f-3,f-1
rule-5: f-1,f-2,f-3	rule-5: f-3,f-2,f-1	rule-3: f-2,f-1
rule-1: f-1,f-2,f-3	rule-1: f-3,f-2,f-1	rule-6: f-1,f-4
rule-2: f-3,f-1	rule-2: f-3,f-1	rule-5: f-1,f-2,f-3
rule-4: f-1,f-2	rule-4: f-2,f-1	rule-1: f-1,f-2,f-3
rule-3: f-2,f-1	rule-3: f-2,f-1	rule-4: f-1,f-2

*Стратегия*упрощения (simplicity strategy) - среди всех правил с одинаковым приоритетом только что активизированное правило располагается выше всех правил с равной или большей определенностью (specificity). Определенность правила задается количеством сопоставлений в левой части правил плюс количество вызовов функций. Логические функции не увеличивают определенность правила.

*Стратегия*усложнения (complexity strategy) - среди всех правил с одинаковым приоритетом только что активизированное правило располагается выше всех правил с равной или большей определенностью.

Случайная *стратегия* (random strategy) - каждой активации назначается случайное число, которое используется для определения местоположения среди активаций с определенным приоритетом.

Подход на основе *стратегий поиска решений в продукционных ЭС* известен

достаточно давно. Весьма популярная в начале 90-х годов ЭСО GURU (ИНТЕР-ЭКСПЕРТ) также использовала подобные механизмы управления *стратегиями поиска*. Возможность смены *стратегии* в ходе решения задачи программным образом и накопление опыта, какие *стратегии* дают лучшие результаты для определенных классов задач, позволяет получить эффективные механизмы поиска решений в СПЗ на основе продукций.

Завершая данную лекцию, следует отметить, что существуют различные *методы поиска решений* в семантических сетях, например, метод обхода семантической сети - мультипарсинг. Данный метод оригинален тем, что позволяет параллельно "вести" по графу несколько маркеров и, тем самым, распараллеливать процесс поиска информации в семантической сети, что увеличивает скорость поиска. Эти методы используются, как правило, при представлении текста в виде объектно-ориентированной семантической сети и в данной лекции не рассматриваются.

Поиск в сетях фреймов, основанный на прецедентах вывод (Case-based Reasoning - CBR), правдоподобные рассуждения (plausible reasoning), *методы поиска* на основе нечеткой логики и другие *методы поиска решений* ИИ в данной лекции также не рассматриваются из-за ограничений на объем данного учебного пособия. Читателю рекомендуется обратиться к соответствующей литературе [49], [50], [51], [52], [53].

Контрольные вопросы к

1. Что такое экспертная система?
2. Основные компоненты ЭС?
3. Рассказать об этапах разработки ЭС.
4. Рассказать о компонентах блока объяснений экспертной системы.
5. Зачем нужно тестирование и опытная эксплуатация ЭС?
6. В чём может заключаться доработка ЭС?
7. Поиск решений в системах продукций.

4. РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ

Рассматриваются характеристики задач распознавания образов и их типы, основы теории анализа и распознавания изображений (признаковый метод), распознавание по методу аналогий. Среди множества интересных задач по распознаванию рассмотрены принципы и подход к распознаванию в задачах машинного чтения печатных и рукописных текстов.

Современные роботы, снабженные телевизионными камерами, способны достаточно хорошо видеть, чтобы работать с реальным миром. Они могут делать заключения о том, какого типа объекты присутствуют, в каких они находятся отношениях между собой, какие группы образуют, какой текст содержат и т. д. Однако сложные задачи распознавания, например, распознавание похожих трехмерных быстродвижущихся объектов или неразборчивого рукописного текста требуют совершенствования методов и средств для своего решения. В этой лекции мы рассмотрим основы некоторых традиционных методов распознавания. Наше рассмотрение мы начнем с наиболее часто применяемого *признакового метода*

распознавания [5], [54].

4.1. Общая характеристика задач распознавания образов и их типы.

Под образом понимается структурированное описание изучаемого объекта или явления, представленное вектором *признаков*, каждый элемент которого представляет числовое значение одного из *признаков*, характеризующих соответствующий объект. Общая структура *системы распознавания* и этапы в процессе ее разработки показаны на рис. 4.1.

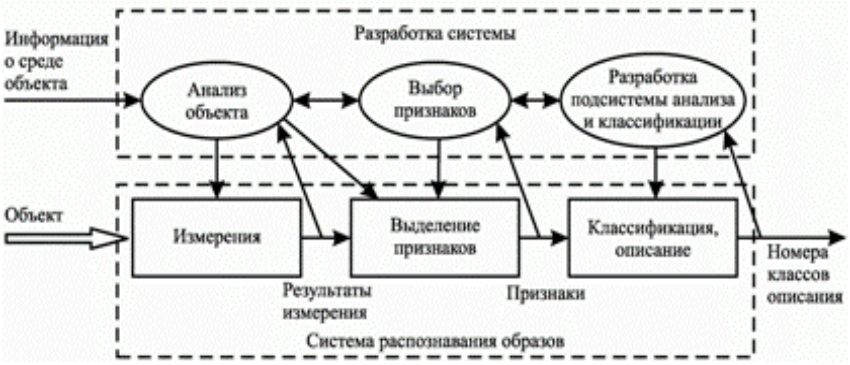


Рис. 4.1. Структура системы распознавания

Суть задачи распознавания - установить, обладают ли изучаемые объекты фиксированным конечным набором *признаков*, позволяющим отнести их к определенному классу.

Задачи *распознавания* имеют следующие *характерные черты*.

1. Это *информационные задачи*, состоящие из двух этапов: а) приведение исходных данных к виду, удобному для *распознавания*; б) собственно *распознавание* (указание принадлежности объекта определенному классу).
2. В этих задачах можно *вводить понятие аналогии или подобия объектов и формулировать понятие близости объектов* в качестве основания для зачисления объектов в один и тот же класс или разные классы.
3. В этих задачах можно *оперировать набором прецедентов-примеров*, классификация которых известна и которые в виде формализованных описаний могут быть предъявлены алгоритму *распознавания* для настройки на задачу в процессе обучения.
4. Для этих задач *трудно строить формальные теории и применять классические математические методы* (часто недоступна информация для точной математической модели или выигрыш от использования модели и математических методов не соизмерим с затратами).
5. В этих задачах *возможна "плохая" информация* (информация с пропусками, разнородная, косвенная, нечеткая, неоднозначная, вероятностная).

Целесообразно выделить следующие типы задач *распознавания*.

1. Задача *распознавания* - отнесение предъявленного объекта по его описанию к одному из заданных классов (обучение с учителем).
2. Задача автоматической классификации - разбиение множества объектов (ситуаций) по их описаниям на систему непересекающихся классов (таксономия, кластерный анализ, обучение без учителя).
3. Задача выбора информативного набора *признаков* при *распознавании*.
4. Задача приведения исходных данных к виду, удобному для *распознавания*.

5. Динамическое *распознавание* и динамическая классификация - задачи 1 и 2 для динамических объектов.

6. Задача прогнозирования - это задачи 5, в которых решение должно относиться к некоторому моменту в будущем.

4.2. Основы теории анализа и распознавания изображений.

Пусть дано множество M объектов ; на этом множестве существует разбиение на конечное число подмножеств (классов) Ω, i = {1,m},M = $\cup\Omega_i$ (i = 1..m) . Объекты ω задаются значениями некоторых *признаков* x_j, j= {1,N} . Описание объекта$I(\omega)=(x_1(\omega), ..., x_N(\omega))$ называют стандартным, если $x_j(\omega)$ принимает значение из множества допустимых значений.

Пусть задана *таблица обучения* (таблица 4.1). Задача *распознавания* состоит в том, чтобы для заданного объекта ω и набора классов Ω_1, ..., Ω_m по обучающей информации в *таблице обучения* $I_0(\Omega_1...\Omega_m)$ о классах и описанию I(ω) вычислить предикаты:

$P_i(\omega \in \Omega_i)=\{1(\omega \in \Omega_i), 0(\omega \in \Omega_i), (\omega \in \Omega_i)\}$,

где i= {1,m} , Δ - неизвестно.

Рассмотрим алгоритмы *распознавания*, основанные на вычислении оценок. В их основе лежит принцип прецедентности (в аналогичных ситуациях следует действовать аналогично).

Пусть задан полный набор *признаков* x_1, ..., x_N. Выделим систему подмножеств множества *признаков* S_1, ..., S_k. Удалим произвольный набор *признаков* из строк ω_1, ω_2, ..., ω_{rm} и обозначим полученные строки через $S\omega_1$, $S\omega_2$, ..., $S\omega_{rm}$, Sω' .

Правило близости, позволяющее оценить похожесть строк Sω' и $S\omega_r$ состоит в следующем. Пусть "усеченные" строки содержат q первых символов, то есть $S\omega_r=(a_1, ..., a_q)$ и $S\omega'=(b_1, ..., b_q)$. Заданы *пороги* $\varepsilon_1...\varepsilon_q$, δ. Строки $S\omega_r$ и Sω' считаются похожими, если выполняется не менее чем δнеравенств вида

$|a_j-b_j| \le \varepsilon_j$, j=1,2,..., q.

Величины $\varepsilon_1...\varepsilon_q$, δвходят в качестве параметров в модель класса алгоритмов на основе оценок.

Пусть $\Gamma_i(\omega')$ - оценка объекта ω' по классу Ω_i.

Описания объектов {ω'}, предъявленные для *распознавания*, переводятся в числовую матрицу оценок. Решение о том, к какому классу отнести объект, выносится на основе вычисления степени сходства *распознавания* объекта (строки) со строками, принадлежность которых к заданным классам известна.

Проиллюстрируем описанный алгоритм *распознавания* на примере. Задано 10 классов объектов (рис. 4.2а). Требуется определить *признаки таблицы обучения*, *пороги* и построить оценки близости для классов объектов, показанных на рис. 4.2б. Предлагаются следующие *признаки таблицы обучения*:

x_1- количество вертикальных линий минимального размера;

x_2- количество горизонтальных линий;

x_3- количество наклонных линий;

x_4- количество горизонтальных линий снизу объекта.

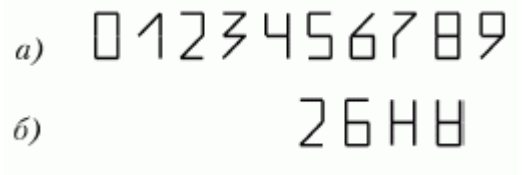


Рис. 4.2. Пример задачи по распознаванию

На рис. 4.3 приведена *таблица обучения* и *пороги*

$\varepsilon_1=1$, $\varepsilon_2=1$, $\varepsilon_3=1$, $\varepsilon_4=1$, $\delta=1$.

Из этой таблицы видно, что неразличимость символов 6 и 9 привела к необходимости ввода еще одного *признака* x_4.

	X_1	X_2	X_3	X_4	
	4	2	0		0
	2	0	1		1
	1	2	1		2
	0	2	2		3
	3	1	0		4
	2	3	0		5
✓	2	2	1	1	6
	1	1	1		7
	4	3	0		8
✓	2	2	1	0	9
	$\varepsilon_1=1$	$\varepsilon_2=1$	$\varepsilon_3=1$	$\varepsilon_4=1$	$\delta=1$

Рис. 4.3. Таблица обучения для задачи по распознаванию

Теперь может быть построена таблица *распознавания* для объектов на рис. 4.2б.

Объект	Объект 1	Объект 2	Объект 3	Объект 4
$\mathbf{x_1}$	**1**	**3**	**4**	**4**
$\mathbf{x_2}$	**2**	**3**	**1**	**2**
$\mathbf{x_3}$	**1**	**0**	**0**	**0**
$\mathbf{x_4}$		**1**		**1**
Результат *распознавания*	Цифра 2	Цифра 8 или 5		

Читателю предлагается самостоятельно ответить на вопрос: что будет, если увеличить *пороги* ε_1, ε_2, ε_3, ε_4, δ? Как изменится качество *распознавания* в данной задаче?

Заключая данный раздел лекции, отметим важную мысль, высказанную А. Шамисом в работе [55]: качество *распознавания* во многом зависит от того, насколько удачно создан алфавит *признаков*, придуманный разработчиками *системы*. Поэтому *признаки* должны быть инвариантны к ориентации, размеру и вариациям формы объектов.

4.2. Распознавание по методу аналогий.

Этот метод очень хорошо знаком студентам (знание решения аналогичной задачи помогает в решении текущей задачи).

Рассмотрим *этот метод* на примере задачи П. Уинстона [5] по поиску геометрических аналогий, представленном на рис. 4.4. Среди фигур второго ряда требуется выбрать $X \in \{1, 2, 3, 4, 5\}$ такое, что A так соотносится с B, как C соотносится с X, и такое, которое лучше всего при этом подходит. Для решения задачи необходимо понять, в чем разница между фигурами A и B (наличие/отсутствие жирной точки), и после этого ясно, что лучше всего для C подходит X=3 .

Решение таких задач предполагает описание изображения и преобразования (отношения между фигурами на изображениях), а также описание изменения

отдельных фигур, составление правил и оценка изменений.

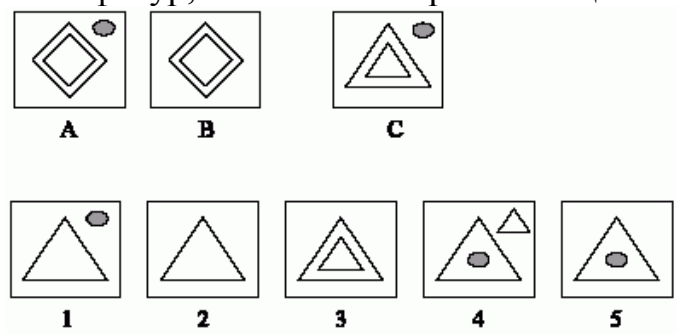


Рис. 4.4. Задача поиска геометрических аналогий

В качестве примера запишем три правила, показывающие, каким образом одно изображение (исходное) становится результирующим (рис. 4.5).

Правило 1 (исходное изображение):k выше m,k выше n,n внутри m

Правило 2 (результир. изображение):n слева m

Правило 3 (масшабирование, повороты):

kисчезло

mизменение масштаба 1:1, вращение 0^0

nизменение масштаба 1:2, вращение 0^0

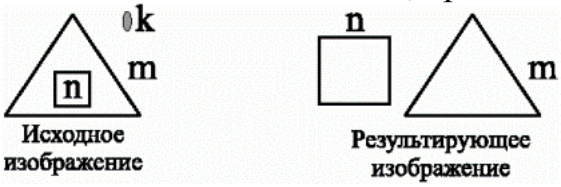


Рис. 4.5. Правила преобразования

Отметим важные моменты при таких преобразованиях. В исходном и результирующем изображениях допускаются отношения ВЫШЕ, ВНУТРИ, СЛЕВА, В результате преобразования изображение может стать МЕНЬШЕ, БОЛЬШЕ, испытать ПОВОРОТ или ВРАЩЕНИЕ, ОТРАЖЕНИЕ, УДАЛЕНИЕ, ДОБАВЛЕНИЕ. Написание правил лучше всего начинать с проведения диагональных линий через центры фигур. Лишние отношения (СПРАВА ОТ и СЛЕВА ОТ, ВЫШЕ и НИЖЕ, ИЗНУТРИ и СНАРУЖИ,) использовать не рекомендуется.

Теперь задачи распознавания мы можем решать достаточно просто, записав для отношений правила 1, 2, 3 и проведя сопоставление, например так, как это сделано для следующей задачи: найти X такое, что A ⇒B , как C ⇒X (рис. 4.6).

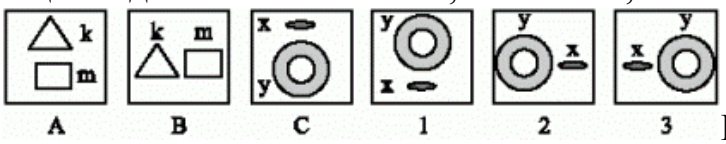


Рис. 4.6. Пример распознавания по аналогии

Правило 1 Правило 2 Правило 3 Результат

A ⇒B kвыше m kслева m k, m масштаб 1:1 поворот 0^0

C ⇒1 xвыше y yвыше x x, y масштаб 1:1 поворот 0^0

C ⇒2 xвыше y yслева x x, y масштаб 1:1 поворот 0^0

C ⇒3 xвыше y xслева y x, y масштаб 1:1 поворот 0^0

Сопоставление успешно

Дополнительно следует отметить, что разные виды преобразований могут иметь различные веса, например, исчезновению фигуры целесообразно назначить больший вес, чем преобразованию масштаба; а вращение фигуры может иметь меньший вес, чем отражение. С этими особенностями можно будет познакомиться в упражнениях к

данной лекции.

Методы распознавания по аналогии могут быть эффективнее, если используется обучение. Различают обучение с учителем, обучение по образцу (эталону) и др. виды обучения [2], [5]. Суть идеи такова. Программе распознавания предъявляется объект, например, арка. Программа создает внутреннюю модель:

(арка
(компонент1 (назначение (опора))
(тип (брусок)))
(компонент2 (назначение (опора))
(тип (брусок)))
(компонент3 (назначение (перекладина))
(тип (брусок))
(поддерживается (компонент1), (компонент2)))

После этого предъявляется другой объект и говорится, что это тоже арка. Программа вынуждена дополнить свою внутреннюю модель:

(арка
(компонент1 (назначение (опора))
(тип (брусок)))
(компонент2 (назначение (опора))
(тип (брусок)))
(компонент3 (назначение (перекладина))
(тип (брусок) или (клин))
(поддерживается (компонент1), (компонент2)))

После такого обучения *система распознавания* будет узнавать в качестве арки как первый, так и второй объект.

4.3. Актуальные задачи распознавания

Среди множества интересных задач по распознаванию (распознавание отпечатков пальцев, распознавание по радужной оболочке глаза, распознавание машиностроительных чертежей и т. д.) следует выделить **задачу определения реальных координат заготовки** и **определения шероховатости обрабатываемой поверхности** , рассмотренную в лекции 10. Другой актуальной задачей является *распознавание машинописных и рукописных текстов* в силу ее повседневной необходимости. Практическое значение задачи *машинного чтения печатных и рукописных текстов* определяется необходимостью представления, хранения и использования в электронном виде огромного количества накопленной и вновь создающейся текстовой информации. Кроме того, большое значение имеет оперативный ввод в информационные и управляющие системы информации с машиночитаемых бланков, содержащих как напечатанные, так и рукописные тексты. В связи с этим рассмотрим принципы и подход к *распознаванию в задаче машинного чтения печатных и рукописных текстов*, описанные в работе [55].Для решения данной задачи используются следующие основные принципы.

1. Принцип целостности - распознаваемый объект рассматривается как единое целое, состоящее из структурных частей, связанных между собой пространственными отношениями.
2. Принцип двунаправленности - создание модели ведется от изображения к модели и от модели к изображению.
3. Принцип предвидения заключается в формировании гипотезы о содержании изображения. Гипотеза возникает при взаимодействии процесса "сверху-вниз", разворачивающегося на основе модели среды, модели текущей ситуации и текущего результата восприятия, и процесса "снизу-вверх",

основанного на непосредственном грубом *признаковом* восприятии.

4. Принцип целенаправленности, включающий сегментацию изображения и совместную интерпретацию его частей.

5. Принцип "не навреди" - ничего не делать до распознавания и вне распознавания, то есть без "понимания".

6. Принцип максимального использования модели проблемной среды.

Указанные принципы реализованы в пакете программ "Графит" [56], в программах FineReader-рукопись и FormReader - для распознавания рукописных символов и, частично, в программе FineReader для *распознавания печатных текстов* [55]. Входящая в FormReader программа *чтения рукописных текстов* была выпущена в 1998 году одновременно с системой ABBYY FineReader 4.0. Эта программа может читать все рукописные строчные и заглавные символы, допускает ограниченные соприкосновения символов между собой и с графическими линиями и обеспечивает поддержку 10 языков. Основное применение программы - распознавание и ввод информации с машиночитаемых бланков.В системе ABBYY FormReader при *распознавании рукописных текстов* используются структурный, растровый, *признаковый*, дифференциальный и лингвистический уровни распознавания. Для более подробного освоения подходов к *распознаванию машинописных и рукописных текстов* в системе ABBYY FormReader читателю рекомендуется непосредственно ознакомиться с работой А. Шамиса [55], при этом знание основ машинной графики на уровне [57] подразумевается.С другими работами по распознаванию читатель может познакомиться в литературе [62], [63].Завершая этот раздел лекции, отметим особенности задачи зрительного восприятия роботов по сравнению с традиционными задачами *распознавания образов* и машинной обработки изображений [64]:

- необходимость построения комплексного описания среды на основе учета значительной априорной информации (модели проблемной среды) в отличиие от традиционной задачи выделения фиксированных *признаков* или измерения отдельных параметров;
- необходимость анализа трехмерных сцен не только в плане анализа трехмерных объектов по их плоским проекциям, но и в плане определения объемных пространственных отношений;
- необходимость анализа изображений, включающих одновременно несколько произвольно расположенных объектов (в общем случае произвольной формы) в отличие от традиционной задачи, когда для распознавания предъявляется, как правило, один объект;
- необходимость анализировать реальную динамическую среду, а не статические изображения;
- отсутствие постоянной фиксированной задачи и необходимость оперативно решать возникающие по ходу дела задачи;
- необходимость следить за изменениями в среде, которые могут порождать новые оперативные задачи;
- необходимость организации системного процесса взаимодействия в реальном времени нескольких подсистем робота ("глаз-мозг", "глаз-мозг-рука").

В заключение лекции следует отметить, что методов распознавания много, они опубликованы (см. список литературы к данной лекции). Успеха в создании серьезных программных продуктов по распознаванию и решению задач зрительного восприятия роботов добьются коллективы, упорно и кропотливо создающие и оттачивающие свои

инструментальные средства для реальных задач *распознавания изображений*.

Контрольные вопросы:

1. Описать актуальные задачи распознавания.
2. Распознавание по методу аналогий.
3. Общая характеристика задач распознавания образов, описать типы.
4. Что такое динамическое распознавание?

5. ОБЩЕНИЕ С ЭВМ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ. СИСТЕМЫ РЕЧЕВОГО ОБЩЕНИЯ

Рассматриваются проблемы понимания естественного языка и дается методология анализа текстов на естественном языке, состоящая из четырех этапов: морфологический анализ, синтаксический анализ, семантическая интерпретация и проблемный анализ. Рассматриваются общие вопросы создания системы речевого общения и построения акустического анализатора и синтезатора речевых сообщений.

Одной из популярных тем исследований ИИ, начиная с 50-х годов, является компьютерная лингвистика, и, в частности, машинный перевод. Появляются основополагающие работы Хомского [65], Вудса [66], Винограда [67], Шенка [68] за рубежом и работы Попова [69], Мальковского [70],Кузина [71] у нас в стране. Эти исследования показали, что проблемы компьютерной лингвистики не так просты и требуют дальнейшей проработки и развития.

5.1. Проблемы понимания естественного языка

Проблемы понимания естественного языка, будь то текст или речь, во многом зависят от знания предметной области. Понимание языка требует знаний о целях говорящего и о контексте. Необходимо также учитывать недосказанность или иносказательность. Например, даже в таком простом предложении «Ваня встретил Машу на поляне с цветами» нам не понятно, кто же был с цветами: Ваня, Маша или поляна? Еще один пример «Врач бегло говорила по-английски». Разбирая это предложение, необходимо в результате разбора зафиксировать, что врач была женщина. Крылатая фраза знаменитого русского лингвиста, академика Л.В.Щербы «Глокая куздра штеко будланула бокра и курдячит бокренка» говорит о том, что такая «непонятная» фраза построена по всем правилам русского языка, не вызывает проблем с грамматическим разбором такого предложения, но вызывает проблемы с пониманием. Попробуем сформулировать лишь некоторые *проблемы понимания естественного языка*.

1. Проблема СМЫСЛ-ТЕКСТ. Об этом только что говорилось и приведем еще один пример по этой проблеме. В предложении «Какой завод заказал оборудование для конвертерного цеха в Бельгии?» неясен смысл: был ли сделан заказ в Бельгии или цех находится в Бельгии.
2. Проблема планирования возникает при необходимости вести диалог, например, на тему «Куда Вы хотите лететь?». В этом случае нужно глубокое знание предметной области (номера рейсов, время прилета-отлета, цены и т.д.).
3. Проблема равнозначности. Будут ли равнозначны два предложения «У дома стоит слон» и «У дома стоит существо с хоботом и бивнями»? На первый взгляд нет сомнений в равнозначности этих предложений. А если в базе знаний существо с хоботом и бивнями определено двумя значениями: слон и мамонт, то такие сомнения, наверное, появятся.
4. Проблемы моделей участников общения. У участников общения должны быть сопоставимые модели представления знаний, необходимая глубина понимания, возможность логического вывода, возможность действия.
5. Проблема эллиптических конструкций, то есть опущенных элементов диалога. Например, в пословице «Береги платье снову, а честь -

смолоду» вторая часть предложения будет синтаксическим *эллипсисом* (опущен глагол береги).

6. Проблема временных противоречий. Например, в предложении «Я хотел завтра пойти в кино» глагол «хотел» в прошедшей форме сочетается с обстоятельством будущего времени «завтра», что противоречит общепринятой логике.

Закончим с перечислением проблем и поговорим об основных понятиях. В качестве языков для общения с программой могут быть: язык меню, язык приказов, анкетный язык. Это регламентированные языки, в них могут работать упрощенные схемы разбора, например, по ключевым словам, и эти языки мы не рассматриваем. В качестве *естественного языка* (ЕЯ) мы рассматриваем подмножество Ограниченного *Естественного Языка* (ОЕЯ) - это профессионально-ориентированное подмножество ЕЯ конечного пользователя. Для разбора ОЕЯ используются программные комплексы, называемые Лингвистическими Трансляторами (ЛТ). Возможная структурная схема ЛТ приведена на рис. 5.1.

Определим или напомним основные понятия. Слово - одна из основных единиц языка, служащая для именования предметов, лиц, процессов, свойств и т.д. Предложение - любое высказывание, являющееся сообщением о чем-либо. ***Словосочетание*** - простейшая единица речи, которая образуется на основе подчинительной связи (согласования, управления, примыкания) двух и более слов. *Словосочетание* в отличие от предложения не является, как правило, сообщением. ***Дискурс*** - связный текст. ***Лексема*** - слово во всей совокупности его лексических значений. ***Морфема*** - минимальная законченная часть слова. ***Аффикс*** - прикрепленная к корню часть слова (подразделяется на *префикс*, *суффикс*, *инфикс*). ***Омонимы*** - разные по значению, но одинаковые по написанию слова, *морфемы* и др. единицы языка («рысь» - бег, «рысь» - животное). ***Синонимы*** - разные по написанию слова, но одинаковые по значению («орать», «кричать» или «дорога», «путь»). ***Эллипсис*** - опущенные слова в предложении («Я еду кататься, а ты?»). ***Анафора*** - повторение объектов предложения («Город пышный, город бедный» - А.С.Пушкин).

Рис. 5.1. Структурная схема ЛТ

5.2. Анализ текстов на естественном языке

Как видно из рис. 5.1, разбор текстов на ОЕЯ состоит из четырех этапов.

Морфологический анализ

(*МА*) определяет грамматические признаки для каждой словоформы. Грамматические признаки наиболее важных частей речи приведены в табл. 5.1.

Таблица 5.1. Грамматические признаки

Часть речи	**Грамматические признаки**
Существительное	Род, число, падеж, склонение
Прилагательное	Род, число, падеж
Глагол	Время, лицо, число, спряжение, вид
Местоимение	Число, лицо

МА для предложения «На мельнице хранятся разные сорта пшеницы» дает следующие результаты разбора (цифрами обозначен порядок слов в предложении): ((на: предлог, 1) (мельница: существительное, жен. род, ед. число, предл. падеж, 2) (храниться: глагол, мн. число, наст. время, несовершенный вид, третье лицо, 3) (разный: прилагательное, мн. число, имен. падеж, 4) (сорт: существительное, муж. род, мн. число, имен. падеж, 5) (пшеница: существительное, жен. род, ед. число, родит. падеж, 6))

Таким образом, мы видим, что для ***МА*** необходим словарь основ слов и словоформ с их грамматическими признаками в зависимости от *аффиксов* и окончаний. *МА* состоит из выделения основы и флексий входной словоформы. По основе определяются основные характеристики данной *лексемы*, а по виду флексии определяются грамматические характеристики словоформы по словарю. Как правило, *МА* не вызывает больших трудностей на этом начальном этапе разбора, хотя и является достаточно трудоемким этапом из-за необходимости создания точных словарей.

Синтаксический анализ

(***СА***) определяет синтаксическую структуру входного предложения. Основные правила *синтаксического анализа*, в большинстве случаев , следующие.

Подлежащим в предложении может быть

1. существительное в именительном падеже;
2. местоимение в именительном падеже;
3. имя собственное в именительном падеже.

Сказуемое в предложении - это глагол, связанный с подлежащим и согласованный с ним в лице и числе. Подлежащее и сказуемое, как известно, это главные члены предложения.

Дополнение - это существительное, согласованное со сказуемым в падеже. Прямое дополнение - существительное в винительном падеже («Я вижу окно»). Косвенное дополнение - дополнение не в винительном падеже, часто с предлогом («Я ехала домой»).

Определение - это прилагательное, связанное с подлежащим или дополнением (связь в роде, числе и падеже - это сильная связь).

Обстоятельство - это, как правило, наречие (неизменяемая часть речи - «далеко», «редко») или существительное с предлогом, связанное со сказуемым только семантически.

СА для нашего предложения о пшенице даст следующие результаты: ((На мельнице : обстоятельство места, 1) (хранятся : сказуемое, 2) (разные : определение, 3) (сорта : подлежащее, 4) (пшеницы : дополнение, 5)).

Семантическая интерпретация

(*СИ*) определяет семантическое представление предложения. Результатом *СИ* должна быть модель в виде, например, семантической сети (см. лекцию 2) для отображения отношений между объектами предложения или лингвистического фрейма (ЛФ).

Исследование вопросов понимания английского *естественного языка* предложено в работах Хомского [65] (классическая книга по трансформационной грамматике), Вудса [66], Винограда [67] и других исследователей. Наибольшее распространение в 70-е годы получили расширенные сети переходов (РСП) Вудса и ATNL-грамматики. В английском языке проблема *СИ* упрощается за счет фиксированного порядка слов в предложении. Например, предложение на английском языке: «The dog has bitten John» переводится как «Собака укусила Джона» или «Джона укусила собака». В русском языке любой вариант перевода правильный и допустим. В английском языке возможен только единственный вариант построения такой фразы:

подлежащее (The dog) ⇒ сказуемое (has bitten) ⇒

⇒ дополнение (John).

Таким образом, использованию РСП и ATNL-грамматик для разбора русского ОЕЯ в чистом виде препятствуют нефиксированный порядок слов в предложениях русского языка, а также синтаксическая неоднозначность грамматических категорий в предложении. Эти ограничения на структуру фраз русского языка делают метод РСП малоэффективным.

Для *СИ* может быть использован метод семантических падежей К. Филмора [34], получивший развитие в работе [72]для разбора русского ОЕЯ. Рассмотрим этот метод подробнее. Предложения выражают чаще всего действия, которые будем отображать в виде предиката в модели на основе ЛФ. Под предикатом в данном случае понимается любой элемент или группа элементов, выполняющих функции сказуемого в предложении, а также атрибутивные формы глагола - причастие, деепричастие, инфинитив. Предикат имеет признаки (модальность, переходность, время, наклонение, возвратность, безличность и т.д.), которые являются необходимыми компонентами для правильной *семантической интерпретации* остальных членов предложения из внешней (грамматической) во внутреннюю (семантическую) структуру. Остальные члены предложения разбиваются на группы сильносвязанных слов, в которых выделяется главное слово (как правило, существительное). В группу его актантов включаются причастия, прилагательные, числительные, местоимения, неопределенно-количественные слова и т.д. Главные слова группы являются актантами предиката и выполняют различные семантические «роли», которые можно описать на основе семантических падежей К. Филмора [34]: агент, объект, цель и т.д., а также дополнительные падежи: адресат, добавочный предикат, инструмент, время, место, определитель, указатель, количество, пример, деталь и т.п.

Целью *СИ* является однозначное выражение смысла предложения в известных системе внутренних понятиях, отношениях и фактах, а также выделение понятий «новой» декларативной информации, приказа для повелительных предложений и вопросительного элемента для вопросительных предложений. *СИ* включает в себя следующие этапы.

1. Грамматическое и семантическое соотнесение очередного анализируемого элемента с уже разобранными элементами. Объединение элементов в группы сильносвязанных слов с проведением проверки «тестов ожидания» аналогично РСП. С помощью «тестов ожидания» можно проверить наличие фиксированных

синтаксических конструкций, информация о которых хранится во входном словаре. Бинарная таблица отношений содержит пары определяемого и зависимого лексических элементов с указанием их грамматико-семантических признаков и семантической роли зависимого слова.

2. Завершение оформления элементов в группы сильносвязанных слов с выделением главного слова, определением семантических ролей внутри группы и определением общих грамматических признаков группы (род, число, падеж и т.д.) на основе информации из словаря. Главное слово в группе выделяется с помощью фильтров модуля СУЩЕСТВИТЕЛЬНОЕ.

3. Определение предиката и его признаков по словарю и выделение в случае группы предикатов главного, связки, глагола «быть», предикативных элементов. Форма предиката (простая, составная глагольная, составная именная) выделяется с помощью фильтров модулей ПРЕДИКАТ. По грамматико-семантическим признакам предикаты разбиваются на несколько КЛАССОВ, указанных в словаре, которые необходимы для выбора формы предиката. КЛАССАМИ предикатов могут быть: ДЕЙСТВИЕ, ФАЗА, СОСТОЯНИЕ, ОБЛАДАНИЕ, РАСПОЛОЖЕНИЕ, ПЕРЕМЕЩЕНИЕ, МОДАЛЬНОСТЬ, КУПЛЯ, ПРОДАЖА, ОТВЛЕЧЕННОЕ ДЕЙСТВИЕ.

4. По окончании входной последовательности слов производится выбор ЛФ-шаблона по классу и типу предиката. В зависимости от типа (личные, безличные, страдательный залог) выбирается соответствующая модификация шаблона. Осуществляется заполнение ЛФ-шаблона с помощью таблиц бинарных отношений предикатов и существительных. В случае неопределенности происходит выделение дополнительных связей между группами существительного с помощью этих же бинарных таблиц. В случае неоднозначности связей используется ряд эвристических правил (принцип близости, принцип приоритетности предиката и т.д.) или обращение в базу знаний ИС.

Завершая описание этапа *СИ*, приведем результат *семантической интерпретации* нашего предложения «На мельнице хранятся разные сорта пшеницы» в виде семантической сети, показанной на рис. 5.2.

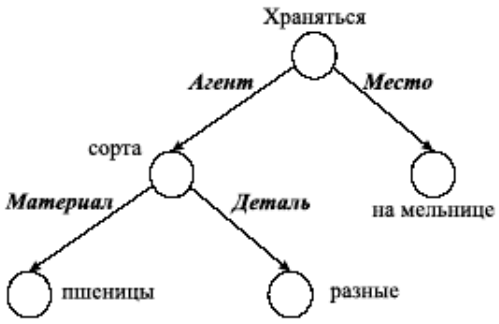


Рис. 5.2. Результат семантической интерпретации предложения

В виде лингвистического фрейма это выглядит следующим образом:

```
(предикат (хранятся)
   (агент (сорта)
      (материал (пшеницы))
      (деталь (разные)))
   (место (на мельнице)))
```

В этом примере в группе сильносвязанных слов («разные сорта пшеницы») выделены свои семантические отношения «материал» и «деталь» у главного слова

в группе («сорта»).

Проблемный анализ

На этом этапе осуществляется отображение входной строки, представленной в виде сети ЛФ, в сеть проблемных фреймов (ПФ), служащую для внутреннего представления знаний в ИС. Для каждой предметной области должна быть разработана собственная база знаний, включающая абстрактную сеть ПФ. «Верхние уровни» ПФ фиксированные и содержат факты, всегда истинные в предполагаемой ситуации. Нижние уровни содержат много терминалов, то есть «ячеек», которые надо заполнить конкретными данными. В каждом терминале могут перечисляться условия, которым такие присваивания значений обязаны удовлетворять. Например, при анализе зрительных сцен различные фреймы в системе соответствуют различным точкам зрения на нее, а переходы от одного фрейма к другому отражают эффект перемещения наблюдателя из одного места в другое. Важно, что различные фреймы одной и той же системы, будь то ЛФ или ПФ, используют общее множество терминалов. Благодаря этому становится возможным координировать информацию, полученную с различных точек зрения в широком смысле.

Однако именно на этом этапе возникает множество вопросов с пониманием смысла и пониманием причинно-следственных связей. Рассмотрим лишь несколько фраз и связанных с ними вопросов, отмеченных в книге П.Уинстона [5]:

«Робби нравится ставить эту пирамиду на красный блок».

Это то же самое, что и фраза «Робби счастлив, когда эту пирамиду помещают на красный блок.»? В таком случае имеет место

(фрейм ИЗМЕНЕНИЕ СОСТОЯНИЯ
(объект РОББИ)
(текущее состояние ())
(результирующее состояние (БОЛЕЕ СЧАСТЛИВ))
(действие (ФРЕЙМ ДЕЙСТВИЕ)))

Предположим, что кто-то сказал:

«Робби успокоил Суззи».

Ясно, что здесь также присутствует изменение состояния (Суззи стала менее грустной). Но что именно сделал Робби? Поговорил ли он с ней, взял на прогулку или просто передвинул пирамиду? Это неизвестно, и поэтому в слоте «действие» ссылка будет отсутствовать.

Рассмотрим еще один пример:

«Суззи была травмирована результатом экзамена.»

Ясно, что это образное выражение, экзамен не повредил Суззи физически, а полученная оценка заставила ее почувствовать себя плохо. Приведенный пример снова показывает, что глагол в предложении может указывать на изменение состояния и не отражать действия.

Все сказанное пока о *проблемном анализе* - это рассуждения о моделях на уровне здравого смысла. Именно здесь проблемы *естественного языка* попадают в область проблем мышления и понимания и происходит обращение к большим базам знаний. Именно здесь предстоит еще решить множество трудных задач.

Конкретный алгоритм заполнения сети ПФ на основе сети ЛФ читателю предлагается придумать самостоятельно. При этом предлагается подумать также над следующими вопросами:

- Как производится сопоставление фреймов, когда имеются некоторые различия

при рассмотрении вещей с различных точек зрения в широком смысле?

- Сколько фреймов нужно для того, чтобы справиться с различными предметными областями, такими, как мир кубиков, мир финансов, политики и окружающий нас мир?
- Как можно усвоить новые фреймы через обобщение старых? Как аналогия может связать различные миры так, чтобы новые фреймы для одного из них можно было бы автоматически строить по фреймам другого?

Завершая наше краткое рассмотрение вопросов общения с ЭВМ на *естественном языке* следует назвать в качестве примеров ранних эффективных отечественных систем для ОЕЯ-общения системы ПОЭТ [70] и АДАЛИТ [71]. В области лингвистической ЕЯ-обработки за последний десяток лет также имеются несомненные успехи [73], [74]: появились коммерческие системы машинного перевода (Stylus, Socrat, Pars, Lingvo и др.), поиска информации в ЕЯ-текстах и аннотирования («Следопыт», «Либретто») и др., создан широкий спектр экспериментальных систем обработки ЕЯ-текстов. Г. Хахалин отмечает также [73] задачи, требующие дальнейшей проработки: трансляция связных ЕЯ-текстов в пределах абзацев и более; полноценный лингвистический синтез текста; автоматизация процесса наполнения моделей; методы проверки ЛТ и лингвистических моделей на полноту, корректность и разнообразие. Следует также отметить недостаточную проработанность вопросов унификации моделей проблемной среды, механизмов вывода для ЛТ.

5.3. Системы речевого общения

В системах ЕЯ-общения обычно предполагается, что в качестве средства общения используется текст или письменная речь. Поэтому в системах ЕЯ-общения под текстом понимается орфографический текст (как пишется), а в системах речевого общения (СРО) используется фонемный текст (как слышится). В СРО решаются задачи преобразования «текст - речевой сигнал» (синтезатор речи) и «речевой сигнал - текст» (анализатор речи). Синтез речи - это возможность обработки текстовой или числовой информации, согласно установленным правилам произношения для конкретного языка, и преобразование ее в синтезированный голос, по восприятию близкий к человеческому. Анализ речи - это распознавание отдельных слов или слитной человеческой речи, с последующим ее преобразованием в текст либо последовательность команд.

На рис. 5.3 показано общее место анализатора и синтезатора речевых сообщений в потоке информации.

Рис. 5.3. Анализатор и синтезатор речевых сообщений в потоке информации

Первые СРО стали появляться в конце 70-х годов. Это было связано со

следующими преимуществами СРО:

1. удобство, простота и естественность процедуры общения, требующей минимума специальной подготовки;
2. возможность использования для связи с ЭВМ обычных телефонных аппаратов и телефонных сетей;
3. устранение ручных манипуляций с одновременным увеличением скорости ввода информации (в 3--5 раз быстрее по сравнению с клавиатурным вводом) и разгрузка зрения при получении информации. Первое и второе преимущество с наибольшим эффектом стали находить применение в автоматизированных системах управления (АСУ). Третье свойство весьма эффективно может применяться при создании систем оперативного человеко-машинного управления сложными объектами (управление движением, энергетическими установками и т.д.).

Обучающие системы, синхронный перевод с одного языка на другой, говорящие книжки, говорящие компьютеры для слепых, управление голосом инвалидными колясками, приборы для генерации и восприятия речи глухонемыми - вот лишь неполный перечень применения СРО.

В основе СРО лежит работа с фонемами. Фонема - это минимальная смысловая единица речи. В русском языке 42 фонемы: 6 гласных и 36 согласных. В английском языке 20 гласных (из них 5 дифтонгов) и 24 согласных, во французском - 16 гласных и 20 согласных.

Акустические характеристики фонем обусловлены местом и способом их образования. По месту образования фонемы делятся на губные (п, б, ф, в, у, м), зубные и межзубные (д, о), альвеолярные (с, з, р, а), заальвеолярные (ш, ж, щ, э), небные (к, г, х, и, ы) и фарингальные (гортанный, например, английское h). По способу образования фонемы делятся на взрывные (п, б, д), аффрикаты (ц, и), щелевые (ф, с, х, в, з, ш, ж,…), дрожащие (р), носовые (м, н), боковые (л), плавные (й), гласные (у, о, а, э, и, ы).

В потоке речи характеристики фонем меняются, что приводит к появлению у них оттенков - аллофонов , например, огубление согласных перед гласными, а также это обусловлено положением фонемы по отношению к ударному слогу, концу и началу слова и т.д.

Интонация и ударение определяют направленность высказывания, логический смысл, выделение главного и общего (рема и тема), вычленение семантически связанных отрезков речи. Интонация и ударение определяют просодию речи с помощью следующих акустических средств:

- мелодика - изменение частоты основного тона голоса;
- ритмика - текущее изменение длительности звуков и пауз;
- энергетика - текущее изменение интенсивности звука.

Акустические характеристики фонем. Речевой аппарат человека в виде ротового и носового параллельных каналов образует единую акустическую систему, возбуждаемую периодическими колебаниями голосовых связок либо турбулентным шумом. Распространение акустических волн в такой системе описывается уравнением Вебстера

$$\frac{1}{S(x)} \cdot \frac{d}{dx}\left(S(x)\frac{dp}{dx}\right) = \frac{1}{c^2} \cdot \frac{d^2p}{dt^2},$$

где S(x) - функция площади сечения голосового тракта вдоль оси x распространения волн; p - давление; c - скорость звука; t - время.

Анализ этого уравнения приводит к передаточным функциям по амплитуде и по частоте (некоторые из них представлены на рис. 5.4).

Речевой сигнал может быть описан как периодическое колебание y(t), создаваемое движением голосовых связок со спектром:

$$y(t) = A\sum_{k=1}^{n} a_k \cos(k\omega_1 t + \varphi_k),$$

где A - среднеквадратичное значение амплитуд спектральных составляющих, a_k - нормированные амплитуды k-х гармоник, ω_1 - частота первой гармоники, φ_k - фазовые сдвиги k-х гармоник, n - число гармоник.

Для разных компонент речевого сигнала (интонация, тембр, громкость, тон, темп) используются разные виды модуляции - амплитудная, частотная, фазовая.

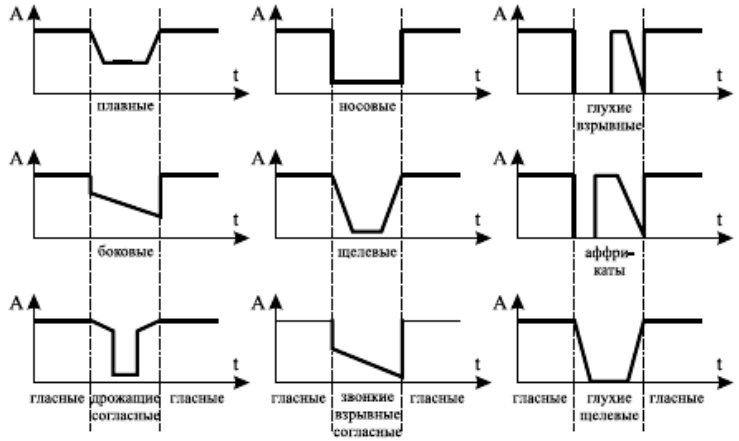


Рис. 5.4. Передаточные функции по амплитуде

Исходя из вышеизложенного, требования к анализатору СРО могут быть сформулированы следующим образом.

1. При анализе заданного элемента информационной структуры осуществляется демодуляция (детектирование) речевого сигнала по каждому виду модуляции, посредством которой ведется его передача. Таким образом, на входе приемного устройства «речевой» системы связи должны быть: демодулятор длительности, амплитудный демодулятор, частотный демодулятор, демодулятор типа переносчика, демодулятор формы спектров.
2. Результат детектирования по каждому виду модуляции должен быть инвариантен относительно остальных видов модуляции. Возможным методом достижения такого рода инвариантности является осуществление предварительной нормализации речевого сигнала.
3. Если с помощью данного вида модуляции осуществляется передача других элементов информационной структуры, то полученный в результате демодуляции сигнал должен быть подвергнут дальнейшим операциям разделения с помощью соответствующих декодеров: декодер информации о фонемном составе, декодер информации об интонации речи, декодер информации об индивидуальности голоса, декодер информации о характеристиках среды, декодер информации о физическом и эмоциональном состоянии.

В настоящее время появляется много интересных разработок в области СРО. Одна из таких разработок - системы синтеза речи Sakrament text-to-speech engine компании «Сакрамент» (Минск, Беларусь, http://www.sakrament.com), созданные с использованием собственных уникальных алгоритмов обработки звука, что позволило добиться высокого качества звучания синтезируемой речи и максимально приблизить

компьютерную речь к человеческой. Эти системы синтеза речи ориентированы на применение в качестве голосовых информаторов в онлайновых телефонных информационных и справочных службах, всевозможных программных приложениях, Интернет-сервисах, бытовых и промышленных приборах и т.д. Система распознавания речи Sakrament Speech Recognition Engine выделяется хорошим качеством распознавания речи, низкой себестоимостью, а также возможностью дальнейшей модификации и настройки. Основная область применения - создание программ, управляющих действиями компьютера или другого электронного устройства с помощью голосовых команд, а также при организации телефонных справочных и информационных служб.В целом вопросом синтеза речи занимается в настоящее время большое число исследовательских групп, каждая из которых создает в конечном итоге свой программный продукт. «Клуб голосовых технологий» МГУ и фирма ПРОМТ создали «Magic Goody», компания Microsoft - Speech SDK, AT&T Германского исследовательского центра искусственного интеллекта - Verbmobil. Ведутся разработки также в Бийском технологическом институте совместно с Томским университетом систем радиоуправления и радиоэлектроники; в «Центре речевых технологий» г.С-Петербург; в компании «Истра-софт» г.Истра и других коллективах и компаниях [75], [76].

Контрольные вопросы:

1. Опишите систему речевого общения?
2. Анализ текстов на естественном языке.
3. Опишите проблемы понимания естественного языка.
4. В чем суть проблемы временных противоречий.

6. МЕТОДОЛОГИЯ ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ

Рассматриваются определения, классификация и структура экспертных систем (ЭС), а также трудности разработки ЭС и методология построения экспертных систем. Описываются примеры ЭС - система G2, OMEGAMON и ЭС диагностирования цифровых устройств.

6.1. Экспертные системы: Определения

Одним из наиболее значительных достижений искусственного интеллекта стала разработка мощных компьютерных систем, получивших название «экспертных» или основанных на «знаниях» систем. В современном обществе при решении задач управления сложными многопараметрическими и сильносвязанными системами, объектами, производственными и технологическими процессами приходится сталкиваться с решением неформализуемых либо трудноформализуемых задач. Такие задачи часто возникают в следующих областях: авиация, космос и оборона, нефтеперерабатывающая промышленность и транспортировка нефтепродуктов, химия, энергетика, металлургия, целлюлозно-бумажная промышленность, телекоммуникации и связь, пищевая промышленность, машиностроение, производство цемента, бетона и т.п. транспорт, медицина и фармацевтическое производство, административное управление, прогнозирование и *мониторинг*. Наиболее значительными достижениями в этой области стало создание систем, которые ставят диагноз заболевания, предсказывают месторождения полезных ископаемых, помогают в проектировании электронных устройств, машин и механизмов, решают задачи управления реакторами

и другие задачи [77], [78]. Примеры *экспертных систем* в различных предметных областях приводятся в конце темы

Эспертная система (ЭС) - программа, которая использует знания специалистов (*экспертов*) о некоторой конкретной узко специализированной предметной области и в пределах этой области способна принимать решения на уровне *эксперта*-профессионала.

Осознание полезности систем, которые могут копировать дорогостоящие или редко встречающиеся человеческие знания, привело к широкому внедрению и расцвету этой технологии в 80-е, 90-е годы прошлого века. Основу успеха *ЭС* составили два важных свойства, отмечаемые рядом исследователей [79], [80]:

- в *ЭС* знания отделены от данных, и мощность *экспертной системы* обусловлена в первую очередь мощностью базы знаний и только во вторую очередь используемыми методами решения задач;
- решаемые *ЭС* задачи являются неформализованными или слабоформализованными и используют эвристические, экспериментальные, субъективные знания *экспертов* в определенной предметной области.

Основными категориями решаемых *ЭС* задач являются: диагностика, управление (в том числе технологическими процессами), интерпретация, прогнозирование, проектирование, отладка и ремонт, планирование, наблюдение (*мониторинг*), обучение.

6.2. Основные компоненты ЭС

На рис. 2 представлены основные компоненты ЭС: БЗ (knowledge base), решатель (inference engine), «классная доска» (blackboard), интерфейс (user interface), подсистема объяснения (explanation subsystem).

ЭС может работать в двух режимах: накопления знаний и консультирования (правая и левая части на рис. 2). Процесс приобретения знаний заключается в извлечении знаний инженером по знаниям из экспертов (или других источников) и внесение их в БЗ в виде фактов и правил. В режиме консультации пользователь посылает ЭС запрос через интерфейс. ЭС реагирует на запрос с помощью механизма вывода (решателя). В БЗ активизируются факты и правила, соответствующие возникшей ситуации, они влекут за собой активизацию других фактов и правил, логически следующих из ситуации и так далее, пока не будет найдено решение. Промежуточные данные фиксируются на «классной доске», параллельно генерируется сценарий объяснения принятия решения. Решение и, при необходимости, объяснения представляются пользователю в удобном виде.

6.3. Типы решаемых задач ЭС:

1) интерпретация, определение смыслового содержания входных данных;
2) предсказание последствий наблюдаемых ситуаций;
3) диагностика неисправностей (заболеваний) по симптомам;
4) конструирование объекта с заданными свойствами при соблюдении установленных ограничений;
5) планирование последовательности действий, приводящих к желаемому состоянию объекта;
6) слежение (наблюдение) за изменяющимся состоянием объекта и сравнение его параметров с установленными или желаемыми;
7) управление объектом с целью достижения желаемого поведения;

8) поиск неисправностей;
9) обучение.

6.4. Ограничения и недостатки ЭС:

.знания экспертов зачастую трудно формализуемы;

.процесс извлечения знаний из экспертов весьма трудоемкий (2-3 года для типовой ЭС);

.часто мнения экспертов не совпадают;

.пользователи ЭС имеют естественные интеллектуальные ограничения, поэтому многие не могут использовать всю мощь ЭС;

.ЭС хорошо работают только на ограниченных предметных областях;

.большинство ЭС не имеют независимой подсистемы проверки решений, хотя решения выдают верные;

.интерфейс ЭС основан на ограниченном словаре специальных терминов данной предметной области, которые понятны не всем пользователям ЭС;

.в разработке ЭС участвуют инженеры по знаниям, которые являются редкими и дорогостоящими специалистами, это делает ЭС слишком дорогими;

.для ЭС допустимо принятие неверных решений, иногда это отталкивает конечных пользователей.

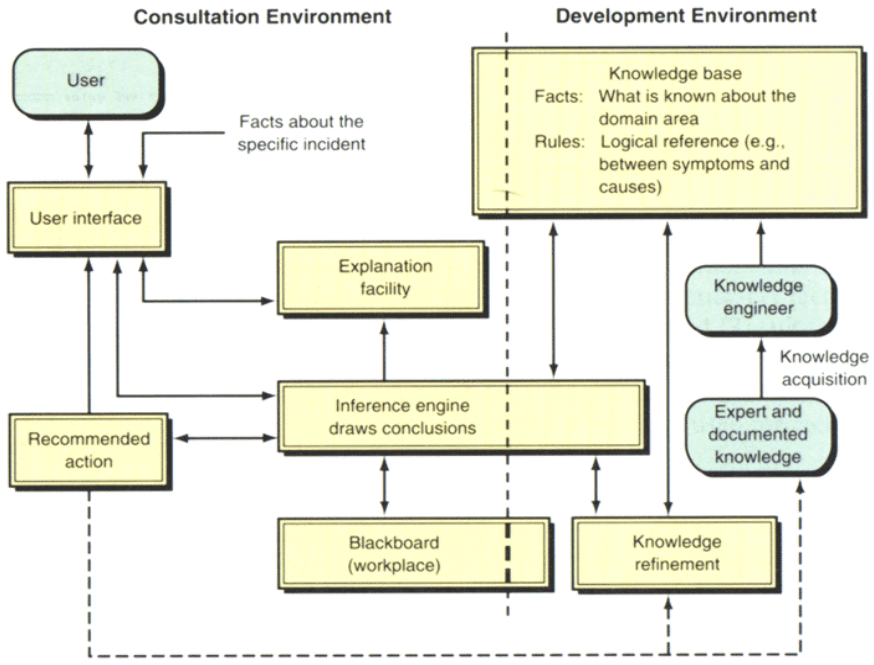


Рис. Структура ЭС

6.5. Обобщенная схема *ЭС*

Обобщенная схема *ЭС* приведена на рис. 6.1. Основу *ЭС* составляет подсистема логического вывода, которая использует информацию из базы знаний (БЗ), генерирует рекомендации по решению искомой задачи. Чаще всего для представления знаний в *ЭС* используются системы продукций и семантические сети. Допустим, БЗ состоит из фактов и правил (если <посылка> то <заключение>). Если *ЭС* определяет, что посылка верна, то правило признается подходящим для данной консультации и оно запускается в действие. Запуск правила означает принятие заключения данного правила в качестве составной части процесса консультации.

Обязательными частями любой *ЭС* являются также модуль приобретения знаний

и модуль отображения и объяснения решений. В большинстве случаев, реальные *ЭС* в промышленной эксплуатации работают также на основе баз данных (БД). Только одновременная работа со знаниями и большими объемами информации из БД позволяет *ЭС* получить неординарные результаты, например, поставить сложный диагноз (медицинский или технический), открыть месторождение полезных ископаемых, управлять ядерным реактором в реальном времени.

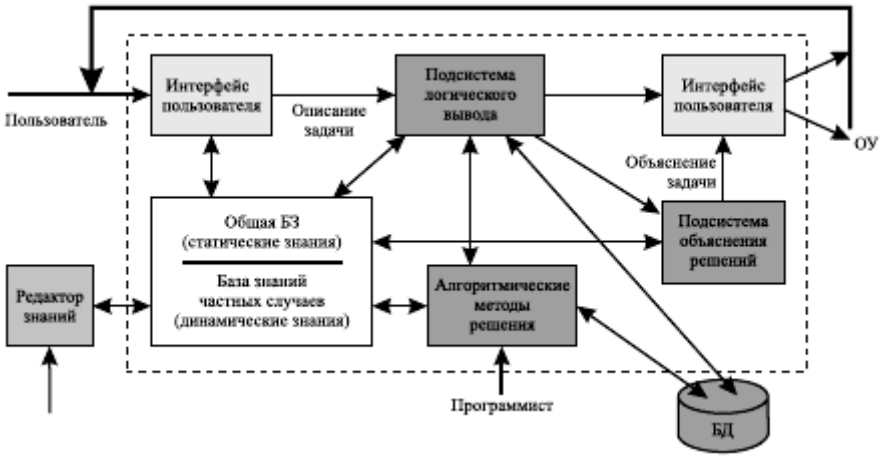


Рис. 6.1. Структура экспертной системы

Важную роль при создании *ЭС* играют инструментальные средства. Среди инструментальных средств для создания *ЭС* наиболее популярны такие языки программирования, как LISP и PROLOG, а также *экспертные системы-оболочки* (ЭСО): KEE, CENTAUR, *G2* и GDA, CLIPS, AT_ТЕХНОЛОГИЯ, предоставляющие в распоряжение разработчика - инженера по знаниям широкий набор для комбинирования систем представления знаний, языков программирования, объектов и процедур [81], [82].

6.6. Экспертные системы: классификация

Рассмотрим различные способы *классификации ЭС*.

По назначению *ЭС* делятся на:

- *ЭС* общего назначения.
- Специализированные *ЭС*:

1. проблемно-ориентированные для задач диагностики, проектирования, прогнозирования

2. предметно-ориентированные для специфических задач, например, контроля ситуаций на атомных электростанциях.

По степени зависимости от внешней среды выделяют:

- Статические *ЭС*, не зависящие от внешней среды.
- Динамические, учитывающие динамику внешней среды и предназначенные для решения задач в реальном времени. Время реакции в таких системах может задаваться в миллисекундах, и эти системы реализуются, как правило, на языке C++.

По типу использования различают:

- Изолированные *ЭС*.
- *ЭС* на входе/выходе других систем.
- Гибридные *ЭС* или, иначе говоря, *ЭС* интегрированные с базами данных и другими программными продуктами (приложениями).

По сложности решаемых задач различают:

- Простые *ЭС* - до 1000 простых правил.
- Средние *ЭС* - от 1000 до 10000 структурированных правил.

- Сложные *ЭС* - более 10000 структурированных правил.

По стадии создания выделяют:

- Исследовательский образец *ЭС*, разработанный за 1-2 месяца с минимальной БЗ.
- Демонстрационный образец *ЭС*, разработанный за 2-4 месяца, например, на языке типа LISP, PROLOG, CLIPS
- Промышленный образец *ЭС*, разработанный за 4-8 месяцев, например, на языке типа CLIPS с полной БЗ.
- Коммерческий образец *ЭС*, разработанный за 1,5-2 года, например, на языке типа C++, Java с полной БЗ.

6.7. Трудности при разработке экспертных систем

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

1. Проблема *извлечения знаний экспертов*. Ни один специалист никогда просто так не раскроет секреты своего профессионального мастерства, свои сокровенные знания в профессиональной области. Он должен быть заинтересован материально или морально, причем хорошо заинтересован. Никто не хочет рубить сук, на котором сидит. Часто такой специалист опасается, что, раскрыв все свои секреты, он будет не нужен компании. Вместо него будет работать *экспертная система*. Избежать эту проблему поможет выбор высококвалифицированного *эксперта*, заинтересованного в сотрудничестве.
2. Проблема *формализации знаний экспертов*. *Эксперты*-специалисты в определенной области, как правило, не в состоянии формализовать свои знания. Часто они принимают правильные решения на интуитивном уровне и не могут аргументированно объяснить, почему принято то или иное решение. Иногда *эксперты* не могут прийти к взаимопониманию (фраза «встретились два геолога, у них было три мнения» - не шутка, а реальная жизнь). В таких ситуациях поможет выбор *эксперта*, умеющего ясно формулировать свои мысли и легко объяснять другим свои идеи.
3. Проблема нехватки времени у *эксперта*. Выбранный для разработки *эксперт* не может найти достаточно времени для выполнения проекта. Он слишком занят. Он всем нужен. У него есть проблемы. Чтобы избежать этой ситуации, необходимо получить от *эксперта*, прежде чем начнется проект, согласие тратить на проект время в определенном фиксированном объеме.
4. Правила, формализованные *экспертом*, не дают необходимой точности. Проблему можно избежать, если решать вместе с *экспертом* реальные задачи. Не надо придумывать «игрушечных» ситуаций или задач. В условиях задач нужно использовать реальные данные, такие как лабораторные данные, отчеты, дневники и другую информацию, взятую из практических задач. Постарайтесь говорить с *экспертом* на одном языке, используя единую терминологию. *Эксперт*, как правило, легче понимает правила, записанные на языке, близком к естественному, а не на языке типа LISP или PROLOG.
5. Недостаток ресурсов. В качестве ресурсов выступают персонал (инженеры знаний, разработчики инструментальных средств, *эксперты*) и средства построения *ЭС* (средства разработки и средства поддержки). Недостаток благожелательных и грамотных администраторов порождает скептицизм и нетерпение у руководителей.

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

6. Неадекватность инструментальных средств решаемой задаче. Часто определенные типы знаний (например, временные или пространственные) не могут быть легко представлены на одном ЯПЗ, так же как и разные схемы представления (например, фреймы и продукции) не могут быть достаточно эффективно реализованы на одном ЯПЗ. Некоторые задачи могут быть непригодными для решения по технологии *ЭС* (например, отдельные задачи анализа сцен). Необходим тщательный анализ решаемых задач, чтобы определить пригодность предлагаемых инструментальных средств и сделать правильный выбор.

О других трудностях и ловушках при создании *ЭС* более подробно можно прочитать в книге Д.Уотермана [77] и учебнике [3].

6.8. Методология построения экспертных систем

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

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

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

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

Введем следующие определения. Разные неисправности ОД проявляются во внешней среде информационными параметрами. Совокупность значений информационных параметров определяет «информационный образ» (ИО) неисправности ОД. ИО может быть полным, то есть содержать всю необходимую информацию для постановки диагноза, или, соответственно, неполным. В случае неполного ИО постановка диагноза носит вероятностный характер.

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

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

1. Выделить множество всех неисправностей ОД, которые должна различать *ЭДС*.

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

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

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

- светится ярко P1 = +++ (или P1 = 3),
- светится слабо P1 = ++ (или P1 = 2),
- не светится P1 = + (или P1 = 1).

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

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

Таблица 6.1. Диагностические правила

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

Для записи правил с учетом изменений по времени следует ввести еще один параметр P0 - время (еще один столбец в таблице). В этом случае диагноз может ставиться на основе нескольких строк таблицы, а в графе Примечания могут быть указаны использованные тесты. Диагностическая таблица в этом случае представлена в таблице 6.1.

Таблица 6.2. Динамические диагностические правила

Номер	P0	P1	P2	P3	Диагноз	Вероятность диагноза	Примечания
1	12:00	+	+	+			тест T1
2	12:15	++	++	+	Неисправен блок A3	0.90	

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

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

ЕСЛИ: P2 = 1

ТО: тест = T1, T3, T7

где T1, T3, T7 - тестовые процедуры, подаваемые на ОД при активизации (срабатывании) соответствующей продукции.

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

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

6.9. Примеры экспертных систем

Для начала совершим краткий экскурс в историю создания ранних и наиболее известных *ЭС*. В большинстве этих *ЭС* в качестве СПЗ использовались системы продукций (правила) и прямая цепочка рассуждений. Медицинская *ЭС* MYCIN разработана в Стэнфордском университете в середине 70-х годов для диагностики и лечения инфекционных заболеваний крови. MYCIN в настоящее время используется для обучения врачей.

ЭС DENDRAL разработана в Стэнфордском университете в середине 60-х годов для определения топологических структур органических молекул. Система выводит молекулярную структуру химических веществ по данным масс-спектрометрии и ядерного магнитного резонанса.

ЭС PROSPECTOR разработана в Стэнфордском университете в 1974--1983 годах для оценки геологами потенциальной рудоносности района. Система содержит более 1000 правил и реализована на INTERLISP. Программа сравнивает наблюдения геологов с моделями разного рода залежей руд. Программа вовлекает геолога в диалог для извлечения дополнительной информации. В 1984 году она точно предсказала существование молибденового месторождения, оцененного в многомиллионную сумму.

Рассмотрим *экспертную систему* диагностирования (ЭСД) цифровых и цифроаналоговых устройств [83], [84], [85], в которой использовались системы продукций и фреймы, а также прямая и обратная цепочка рассуждений одновременно. В качестве объекта диагностирования (ОД) в ЭСД могут использоваться цифровые устройства (ЦУ), БИС, цифро-аналоговые устройства. На рис. 6.2 показано, что такая ЭСД работает совместно с автоматизированной системой контроля и диагностирования (АКД), которая подает в динамике воздействия на ОД (десятки, сотни и тысячи воздействий в секунду), анализирует выходные реакции и дает заключение: годен или не годен. В случае, если реакция проверяемого ОД не соответствует эталонным значениям, то подключается основанная на знаниях подсистема диагностирования. ЭСД запрашивает значения сигналов в определенных контрольных точках и ведет оператора по схеме ОД, рекомендуя ему произвести измерения в определенных контрольных точках или подтвердить промежуточный диагноз, и в результате приводит его к месту неисправности. Исходными данными для работы ЭСД являются результаты машинного моделирования ОД на этапе проектирования. Эти результаты моделирования передаются в ЭСД на магнитных

носителях в виде тысяч продукционных правил. Движение по контрольным точкам осуществляется на основе модели, записанной в виде сети фреймов для ОД.

Такая ЭСД не была бы интеллектуальной системой, если бы она не накапливала опыт. Она запоминает найденную неисправность для данного типа ОД. В следующий раз при диагностике неисправности ОД этого типа она предлагает проверить сразу же эту неисправность, если реакция ОД говорит о том, что такая неисправность возможна. Так поступают опытные мастера радиоэлектронной аппаратуры (РЭА), знающие «слабые» места в конкретных типах РЭА и проверяющие их в первую очередь. ЭСД накапливает вероятностные знания о конкретных неисправностях с целью их использования при логическом выводе. При движении по дереву поиска решений на очередном шаге используется критерий - максимум отношения вероятности (коэффициента уверенности) постановки диагноза к трудоемкости распознавания неисправности. Коэффициенты уверенности автоматически корректируются во время работы ЭСД при каждом подтверждении или не подтверждении диагноза для конкретных ситуаций диагностирования. Трудоемкости элементарных проверок первоначально задаются *экспертом*, а затем автоматически корректируются в процессе работы ЭСД.

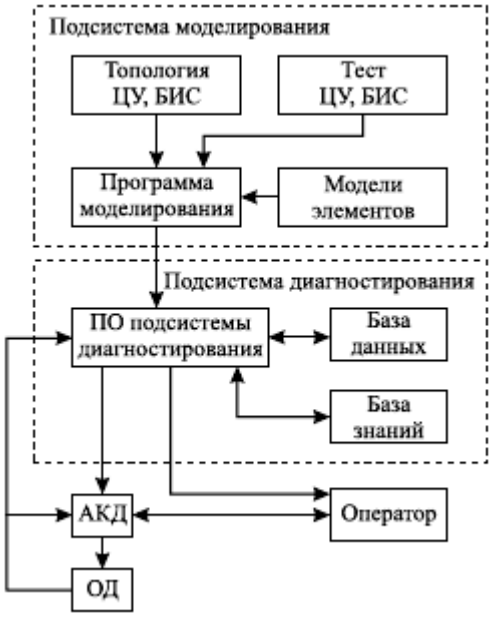


Рис. 6.2. Общая структура экспертной системы диагностирования

ЭСД не была реализована в виде ИРС по экономическим соображениям. Небольшая серийность проверяемой аппаратуры, недостаточная унификация и дешевая рабочая сила (последний фактор и в наше время играет в России немаловажную роль) помешали реализовать полностью автоматическое диагностирование.

Среди современных коммерческих систем хочется выделить *экспертную систему* - оболочку *G2* американской фирмы Gensym (USA) [80] как непревзойденную экспертную коммерческую *систему* для работы с динамическими объектами. Работа в реальном времени с малыми временами ответа часто необходима при анализе ситуаций в корпоративных информационных сетях, на атомных реакторах, в космических полетах и множестве других задач. В этих задачах необходимо принимать решения в течение миллисекунд с момента возникновения критической ситуации. ЭС *G2*, предназначенная для решения таких задач, отличается от большинства динамических *ЭС* такими характерными свойствами, как:

- работа в реальном времени с распараллеливанием процессов рассуждений;
- структурированный естественно-языковый интерфейс с управлением по меню и автоматической проверкой синтаксиса;
- обратный и прямой вывод, использование метазнаний, сканирование и фокусирование;
- интеграция подсистемы моделирования с динамическими моделями для различных классов объектов;
- структурирование БЗ, наследование свойств, понимание связей между объектами;
- библиотеки знаний являются ASCII-файлами и легко переносятся на любые платформы и типы ЭВМ;
- развитый редактор для сопровождения БЗ без программирования, средства трассировки и отладки БЗ;
- управление доступом с помощью механизмов авторизации пользователя и обеспечения желаемого взгляда на приложение;
- гибкий интерфейс оператора, включающий графики, диаграммы, кнопки, пиктограммы и т.п.;
- интеграция с другими приложениями (по TCP/IP) и базами данных, возможность удаленной и многопользовательской работы.

В качестве примера быстродействующей системы для отслеживания состояния корпоративной информационной сети (КИС) можно привести основанную на знаниях *систему мониторинга OMEGAMON фирмы Candle* (IBM с 2004 г.) . *OMEGAMON* - типичный представитель современных экспертных мультиагентных динамических *систем*, работающих в реальном времени. *OMEGAMON* позволяет за считанные минуты ввести и отладить правила *мониторинга* внештатных ситуаций для объектов КИС. Правило (situation) записывается как продукция. Логический вывод в такой *ЭС* реализован при помощи механизма policy, обеспечивающего построение цепочек логического вывода на основе situations. На рис. 6.3 приведен один из интерфейсов для заполнения БЗ в *ЭС OMEGAMONM*. На этом рисунке показана ситуация, определяющая критическое количество сообщений в очередях транспортной системы IBM *WebSphere MQ* (*MQSeries*).

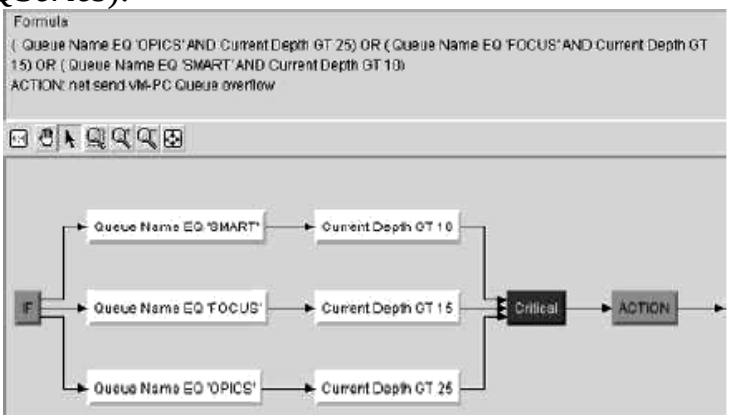


Рис. 6.3. Интерфейс OMEGAMON для заполнения БЗ

На рис. 6.4 показаны основные компоненты системы *OMEGAMON*:

- сервер сбора информации от агентов CandleManagementServer (CMS);
- сервер отображения результатов, оповещения пользователей и *настройки мониторинга* КИС CandleNetPortal Server (CNP) со своими клиентами;
- Candle Management Workstation (CMW) - рабочая станция администратора *OMEGAMON*;

- Managed Systems - компьютеры КИС, на которых работают агенты.

Агенты *OMEGAMON* работают на контролируемых системах (Managed Systems), как первоклассные шпионы: они незаметны с точки зрения использования CPU и оперативны при *мониторинге* с точки зрения времени поставки своих донесений в центр (CMS). Они фиксируют критическую ситуацию и обеспечивают реакцию (ACTION) менее чем за 1 секунду. Все определяется тем интервалом *мониторинга*, который задается *экспертом* на основе своих интуитивных знаний. В качестве ACTION при определении ситуаций можно использовать различные типы действий: посылку почтовых сообщений и sms специалистам сопровождения, посылку информации в другие системы, выполнение системных команд и т.д. Количество объектов *мониторинга* (компьютеров КИС) может достигать нескольких сотен, и на каждом объекте может быть несколько сотен контролируемых параметров. Количество платформ (типов операционных систем), на которых работают агенты, превышает 30, начиная от OS/390,,OS/400, далее различные UNIX-платформы (HP_UX, AIX, Solaris) и заканчивая Windows. На одном сервере может работать несколько агентов, например, для *мониторинга WebSphere MQ* (*MQSeries*), WebSphere Application Server, DB-2 и HP_UNIX одновременно.

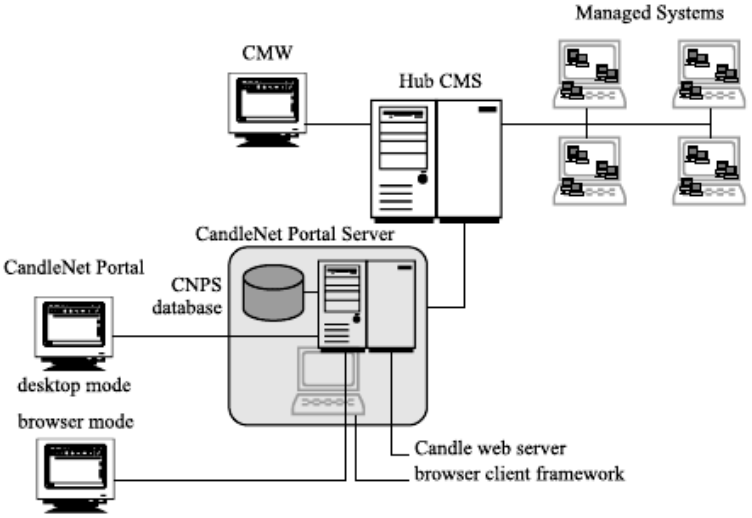


Рис. 6.4. Структурная схема ЭС OMEGAMON

Серверы CMS и CNP-servers могут работать на одном выделенном сервере, как правило, на базе операционной системы Windows. Настройка ситуаций (situations) и механизмов логического вывода (policy) производится на рабочем компьютере администратора через CNP-client. Для только что созданной ситуации вы нажимаете кнопку Apply и моментально видите отображение ACTION через CNP-client, через почту и т.д.

Следует подчеркнуть, что основанная на знаниях система *мониторинга OMEGAMON* - это весьма эффективная система управления вычислительными ресурсами, надежный и незаменимый помощник в поисках решений по оперативному устранению критических и трудных для диагностирования ситуаций, при анализе информационных потоков, анализе *производительности* и *настройке* КИС.

В следующей лекции будет рассмотрена практическая реализация *экспертной системы* управления технологическим процессом в составе ИРС.

Контрольные вопросы:

1. Приведите примеры экспертных систем.
2. Классификация экспертных систем.

3. Трудности при разработке экспертных систем
4. Методология построения экспертных систем.

7. ПРАКТИЧЕСКАЯ РАЗРАБОТКА ЭКСПЕРТНЫХ СИСТЕМ В СРЕДЕ CLIPS

Рассматривается практическая разработка конкретной экспертной системы управления технологическим процессом на базе экспертной системы - оболочки CLIPS.

7.1 Постановка задачи

Создание *экспертной системы управления технологическим процессом* (ТП) может значительно ускорить процесс разработки сложной системы управления ТП, повысить качество решения задачи и дать экономию ресурсов за счет эффективного распределения функций центрального управления и локальных измерительных и управляющих подсистем. Такой эффект достигается за счет открытости системы представления знаний об объекте управления, адаптивности системы к условиям функционирования, автоматической коррекции управляющих воздействий при изменении существенных параметров в процессе функционирования.

ЭС *CLIPS* рассматривается в лекции как инструментальное средство для разработки. Выбор *CLIPS* обусловлен двумя причинами: во-первых, эта ЭС, разработанная NASA, доказала свою эффективность и **свободно распространяется через Internet**; во-вторых, реализация *CLIPS* на языке C++ позволяет переносить конкретные ЭС на различные типы операционных систем. Кроме того, может быть обеспечена возможность работы в реальном масштабе времени, когда реакция системы на возмущения должна не превышать нескольких миллисекунд.

В качестве ТП рассмотрим создание деталей сложной формы, например, вытачивание лопаток турбины. В данном случае ИРС осуществляет управление технологическим процессом через систему управления высшего уровня, способную к самостоятельному функционированию и обеспечивающую выполнение всех основных функций по управлению сбором и анализом информации и принятию оперативных решений по ходу процесса на основе разрабатываемой ЭС. В состав ИРС входит ряд локальных управляющих подсистем нижнего уровня, каждая из которых осуществляет управление одним из компонентов ТП по жесткому алгоритму в реальном времени. ЭС управления ТП, разрабатываемая в рамках данной лекции, обеспечивает организацию сбора информации об управляемом процессе от локальных управляющих подсистем, управление режимами их функционирования и принятие оперативных управляющих решений на основе информации,поступившей от систем управления нижнего уровня. В общем случае управление ТП может осуществляться полностью автоматически.

Приступим к формализации знаний экспертов по управлению ТП создания деталей сложной формы. Выделим множество информативных (существенных) параметров, влияющих на ТП и позволяющих управлять ТП с некоторой достоверностью. Одновременно для выбранных параметров выделим информативные значения или информативные диапазоны значений. Указанные параметры и их значения представлены в табл. 7.1.

Таблица 7.1. Информативные параметры ТП

№№ п/п	Обозначение параметра	Название параметра	Единица измерения	Диапазон значений

1	Vr	Скорость резания	об/мин	A, B, C
2	Vm	Подача	мм/с	10-180 с шагом 10
3	T	Виды траектории		Круговая (T=0), по участкам 1(T=1), ..., по участкам 6(T=6)
4	I	Инструмент		алмазный (I=1), на бакелитовой основе (I=2), на эльборовой основе (I=3)
5	G	Геометрические параметры инструмента		тор (G=tor), линия (G=line), макс.радиус вращения Rm, угол контакта инструмента и детали J, угол заточки S
6	Ds	Размер детали	мм	10,300,800
7	Dm	Материал детали		Титан1 (Dm=1), Титан2 (Dm=2), Жаропрочная сталь (Dm=3)
8	Da	Требования к детали по точности		1, 2, 3
9	Dar	Достигнутая точность детали		1, 2, 3

Запишем со слов экспертов информационные образы управляющих решений в алфавите значений информационных параметров. В таблице 7.2 представлена база знаний (база правил) нашей *экспертной системы управления технологическим процессом*. Здесь достоверность это уверенность эксперта, что такое воздействие позволит достичь заданных параметров обработки Ds, Dm, Da, Dar на основе данного воздействия.

Таблица 7.2. База знаний ЭС

№№ п/п	Ds	Dm	Da	Dar	Управляющее воздействие	Достоверность	Прим.
1	10	1	1		Vr=A, Vm=10, T=0, I=1, G=tor	0,98	
2	10	2	2		Vr=B, Vm=10, T=1, I=1, G=line, Rm=40, J=80, S=60	0,95	
3	300	2			Vr=B, Vm=20, T=2, I=1, G=tor	0,92	
4	300		3		Vr=C, Vm=40, T=3, I=2, G=line, Rm=50, J=75, S=75	0,97	
5	800	2	2	1	Vr=B, Vm=60, T=4, I=2, G=line, Rm=60, J=70, S=70	0,94	
6	800			< 3	Vr=C, Vm=80, T=6, I=3, G=line, Rm=60, J=60, S=75	0,90	
7	800			3	Vr=B, Vm=40, T=6, I=3, G=line, Rm=60, J=60, S=75	0,90	

Достоверность правильности управляющего воздействия должна автоматически корректироваться по результатам изготовления детали. В табл. 7.2 приведен учебный пример базы знаний, упрощенный для целей реализации. Здесь не сформулированы

задачи работы с базой данных. База целей (конфликтное множество правил) является внутренним для *CLIPS* механизмом. В общем случае, в процессе обработки производится измерение параметров, и управляющие воздействия задаются в зависимости от результатов измерений и БЗ управляющих воздействий. Например в данном примере, пока точность детали Dar < 3, работает строка 6 таблицы 7.2, как только Dar достигло значения 3, начинает работать строка 7. Это и есть простейший пример работы ЭС в реальном времени.

7.2. Основы программирования в системе CLIPS

CLIPS (C Language Integrated Production System) начала разрабатываться в космическом центре Джонсона NASA в 1984 году. Сейчас *CLIPS* и документация на этот инструмент свободно распространяется через интернет (http://www.ghg.net/clips/clips.html). Язык *CLIPS* свободен от недостатков предыдущих инструментальных средств для создания ЭС, основанных на языке LISP. Язык *CLIPS* получил большое распространение в государственных организациях и учебных заведениях благодаря низкой стоимости, мощности, эффективности и переносимости с платформы на платформу. Например, даже Web-ориентированный инструментарий JESS (Java Expert System Shell), использующий язык представления знаний *CLIPS*, приобрел достаточную известность в настоящее время.

Следует отметить, что несмотря на многочисленные преимущества функционального программирования, некоторые задачи лучше решать в терминах объектно-ориентированного программирования (ООП), для которого характерны три основные возможности: ИНКАПСУЛЯЦИЯ (работа с классами), ПОЛИМОРФИЗМ (работа с родовыми функциями, поддерживающими различное поведение функции в зависимости от типа аргументов), НАСЛЕДОВАНИЕ (поддержка абстрактных классов). ООП поддерживает многие языки, в том числе Smalltalk, C++, Java, Common LISP Object System (CLOS). Язык *CLIPS*, в свою очередь, вобрал в себя основные преимущества C++ и CLOS.

Читатель может познакомиться с языком *CLIPS*, получив через Интернет полный комплект документации на английском языке, или прочитав изданную на русском языке книгу. В данном разделе лекции дается краткое неформальное введение в *CLIPS*, необходимое для программирования учебных задач.

Отличительной особенностью *CLIPS* являются конструкторы для создания баз знаний (БЗ):

defrule	определение правил;
deffacts	определение фактов;
deftemplate	определение шаблона факта;
defglobal	определение глобальных переменных;
deffunction	определение функций;
defmodule	определение модулей (совокупности правил);
defclass	определение классов;
defintances	определение объектов по шаблону, заданному defclass;
defmessage handler	определение сообщений для объектов;
defgeneric	создание заголовка родовой функции;
defmethod	определение метода родовой функции.

Конструкторы не возвращают никаких значений, в отличии от функций, например:

```
(deftemplate person
(slot name)
(slot age)
(multislot friends))
(deffacts people
(person (name Joe) (age 20))
(person (name Bob) (age 20))
(person (name Joe) (age 34))
(person (name Sue) (age 34))
(person (name Sue) (age 20)))
```

Пример функции:

```
(deffunction factorial (?a)
  (if (or (not (integerp ? a)) (< ? a0)) then
    (printout t "Factorial Error!" crlf)
  else
    (if (= ? a0) then
      1
    else
      (*? a (factorial ($-$ ? a1))))))
```

Правила в *CLIPS* состоят из предпосылок и следствия. Предпосылки также называют ЕСЛИ-частью правила, левой частью правила или LHS правила (left-hand side of rule). Следствие называют ТО-частью правила, правой частью правила или RHS правила (right-hand side of rule).

Пример правила представлен ниже:

```
(deftemplate data (slot x) (slot y))
(defrule twice
(data (x ? x) (y =(*2 ? x)))
⇒)
(assert (data (x2) (y4)); f-0
  (data (x3) (y9))); f-1
```

Здесь самая распространенная в *CLIPS* функция assert добавляет новые факты в список правил. В противоположность assert функция retract удаляет факты из списка фактов, например:

```
(defrule vis11
  ?doors < — (fit ? wdfit)
  (test (eq ? wdfit no))
 ⇒
  (assert (EVIDENCE OF MAJOR ACCIDENT))
  (retract ? doors))
```

В этом правиле проверяется наличие факта doors и в случае его отсутствия факт doors удаляется из списка фактов задачи.

Функция modify является также весьма распространенной. Она позволяет в определенном факте поменять значение слота, например,

```
(deftemplate age (slot value))
(assert (age (value young)))
(modify 0 (value old))
```

Следующий пример описывает представление данных в виде фактов, объектов и глобальных переменных. Примеры фактов:

```
(voltage is 220 volt)
(meeting (subject "AI") (chief "Kuzin") (Room "3240"))
```

В первой строке приведен упорядоченный факт, во второй - неупорядоченный, в

котором порядок слотов не важен.

CLIPS поддерживает следующие типы данных: integer, float, string, symbol, external-address, fact-address, instance-name, instance-address.

Пример integer:	594	23	+51	−17
Пример float:	$594e^{2}$	23.45	+51.0	$-17.5e^{-5}$

String — это строка символов, заключенная в двойные кавычки.

Пример string: "expert", "Phil Blake", "состояние -0", "quote=\"

CLIPS поддерживает следующие процедурные функции , реализующие возможности ветвления, организации циклов в программах и т.,п.:

If	оператор ветвления;
While	цикл с предусловием;
loop-for-count	итеративный цикл;
prong	объединение действий в одной логической команде;
prong$	выполнение набора действий над каждым элементом поля;
return	прерывание функции, цикла, правила и т.д.;
break	то же, что и return, но без возвращения параметров;
switch	оператор множественного ветвления;
bind	создание и связывание переменных.

Функции *CLIPS* описываются в книгах. Среди **логических функций** (возвращающих значения true или false) следует выделить следующие группы:

- функции булевой логики: and, or, not
- функции сравнения чисел: =, ≠, >, ≥, <, ≤
- предикативные функции для проверки принадлежности проверяемому типу: integerp, floatp, stringp, symbolp, pointerp (относится ли аргумент к xternal-address), numberp (относится ли аргумент к integer или float), lexemerp (относится ли аргумент к string или symbol), evenp (проверка целого а четность), oddp (проверка целого на нечетность), multifildp (является ли аргумент составным полем).
- Функции сравнения по типу и по значению: eq, neq

Среди **математических функций** следует выделить следующие группы:

- Стандартные: +, -, *, /, max, min, div (целочисленное деление), abs (абсолютное значение), float (преобразование в тип float), integer (преобразование в тип integer)
- Расширенные: sqrt (извлечение корня), round (округление числа), mod (вычисление остатка от деления)
- Тригонометрические: sin, sinh, cos, cosh, tan, tanh, acos, acosh, acot, acoth, acsc, acsch, asec, asech, asin, asinh, atan, atanh, cot, coth, csc, csch, sec, sech, deg-grad (преобразование из градусов в секторы), deg-rad (преобразование из градусов в радианы), grad-deg (преобразование из секторов в градусы), rad-deg (преобразование из радиан в градусы)
- Логарифмические: log, log10, exp, pi

Среди **функций работы со строками** следует назвать функции:

str-cat	объединение строк,
sym-cat	объединение строк в значение типа symbol,
sub-string	выделение подстроки,
str-index	поиск подстроки,
eval	выполнение строки в качестве команды *CLIPS*,
build	выполнение строки в качестве конструктора *CLIPS*,

upcase	преобразование символов в символы верхнего регистра,
lowcase	преобразование символов в символы нижнего регистра,
str-compare	сравнение строк,
str-length	определение длины строки,
check-syntax	проверка синтаксиса строки,
string-to-field	возвращение первого поля строки.

Функции работы с составными величинами являются одной из отличительных особенностей языка *CLIPS*. В их число входят:

create$	создание составной величины,
nth$	получение элемента составной величины,
members	поиск элемента составной величины,
subset$	проверка одной величины на подмножество другой,
delete$	удаление элемента составной величины,
explode$	создание составной величины из строки,
implode$	создание строки из составной величины,
subseq$	извлечение подпоследовательности из составной величины,
replace$	замена элемента составной величины,
insert$	добавление новых элементов в составную величину,
first$	получение первого элемента составной величины,
rest$	получение остатка составной величины,
length$	определение числа элементов составной величины,
delete-member$	удаление элементов составной величины,
replace-member$	замена элементов составной величины.

Функции ввода-вывода используют следующие логические имена устройств:

stdin	устройство ввода,
stdout	устройство вывода,
wclips	устройство, используемое как справочное,
wdialog	устройство для отправки пользователю сообщений,
wdisplay	устройство для отображения правил, фактов и т.,п.,
werror	устройство вывода сообщений об ошибках,
wwarning	устройство для вывода предупреждений,
wtrase	устройство для вывода отладочной информации,

Собственно функции ввода-вывода следующие:

open	открытие файла (виды доступа r, w, r+, a, wb),
close	закрытие файла,
printout	вывод информации на заданное устройство,
read	ввод данных с заданного устройства,
readline	ввод строки с заданного устройства,
format	форматированный вывод на заданное устройство,
rename	переименование файла,
remove	удаление файла.

Среди двух десятков команд *CLIPS* следует назвать основные команды при работе со средой *CLIPS*:

load	загрузка конструкторов из текстового файла,
load+	загрузка конструкторов из текстового файла без отображения,
reset	сброс рабочей памяти системы *CLIPS*,
clear	очистка рабочей памяти системы,
run	выполнение загруженных конструкторов,

save exit	сохранение созданных конструкторов в текстовый файл,
	выход из *CLIPS*.

В рамках нашего краткого описания опустим список функций для работы с методами родовых функций и список функций для работы с классами, объектами, слотами, обработчиками сообщений. С этим можно ознакомиться по документации. Список сообщений об ошибках приведен в [91].

В завершение следует иметь в виду, что *CLIPS* может не удовлетворительно работать в реальном времени, когда потребуется время реакции менее 0,1 сек.В этом случае надо исследовать на разработанном прототипе механизмы вывода для всего множества правил предметной области на различных по производительности компьютерах. Как правило, современные мощные компьютеры Intel обеспечивают работу с продукционными системами объемом 1000--2000 правил в реальном времени. Веб-ориентированные средства на базе JAVA (системы Exsys Corvid, JESS) являются более медленными, чем, например, CLIPS 6.0 или OPS-2000. Поэтому *CLIPS* - лучший на сегодня выбор для работы в реальном времени среди распространяемых свободно оболочек ЭС, разработанных на C++.

7.3. Программирование в CLIPS экспертной системы управления технологическим процессом

Программа ЭС управления ТП по обработке деталей сложной формы, разработанная на основе табл. 7.1 и табл. 7.2, выглядит следующим образом.

```
;;;======================================
;;; Control Expert System of technological process
;;;
;;; This expert system administers technological process
;;; of creations of details of the complex form
;;;
;;; CLIPS Version 6.0 Example
;;;Author: Vladimir Makushkin, vmakushkin@mail.ru
;;;
;;; To execute, merely load, reset and run.
;;;======================================
(deffacts initial-state
     (Ds 800)
     (Dm 2)
     (Da 2)
(Dar 1))

(defrule rule1
  (declare (salience 9098))
     (Ds 10)
     (Dm 1)
     (Da 1)
⇒
(printout t "Rule1: Vr=A, Vm=10, T=0, I=1, G=tor" crlf))

(defrule rule2
  (declare (salience 9095))
     (Ds 10)
     (Dm 2)
     (Da 2)
⇒
```

```
(printout t "Rule2: Vr=B, Vm=10, T=1, I=1, G=line, Rm=40, J=80, S=60" crlf))

(defrule rule3
  (declare (salience 9092))
    (Ds 300)
    (Dm 2)
⇒
(printout t "Rule3: Vr=B, Vm=20, T=2, I=1, G=tor" crlf))

(defrule rule4
  (declare (salience 9097))
    (Ds 300)
    (Da 3)
⇒
(printout t "Rule4: Vr=C, Vm=40, T=3, I=2, G=line, Rm=50, J=75, S=75" crlf))

(defrule rule5
  (declare (salience 9094))
    (Ds 800)
    (Dm 2)
    (Da 2)
    (Dar 1)
⇒
(printout t "Rule5: Vr=B, Vm=60, T=4, I=2, G=line, Rm=60, J=70, S=70" crlf))

(defrule rule6_7
  (declare (salience 9090))
    (Ds 800)
    (Dar ? num)
⇒
    (if (< ? num 3)
        then
     (printout t "Rule6: Vr=B, Vm=40, T=6, I=3, G=line, Rm=60, J=60, S=75" crlf)
        else
     (printout t "Rule7: Vr=C, Vm=80, T=6, I=3, G=line, Rm=60, J=60, S=75" crlf)))
```

Листинг 7.1. Программа ЭС управления ТП по обработке деталей сложной формы.

Эта программа сохраняется в виде файла с именем, например, robot.clp, далее в среде *CLIPS* выполняются команды: clear; load robot.clp; reset и run. Эта программа начинает работать. Входные воздействия заданы в данном примере через deffacts initial-state. Активизация правил БЗ для конкретных воздействий, заданных в программе, дает конфликтное множество (базу целей) - правила rule5 и rule6_7, а затем по критерию максимальной достоверности первым выбирается управляющее воздействие на систему низшего уровня:

Rule5: Vr=B, Vm=60, T=4, I=2, G= line, Rm=60, J=70, S=70

В реальной жизни входные воздействия поступают через оператор read (ввод данных с заданного устройства), например, следующим образом:

```
(defrule Dar_parameter
(declare (salience 9101))
  (Dar ? num)
⇒
(printout t "Ds parameter has value "crlf"
```

```
 1) 10 "crlf" 2) 300 "crlf" 3) 800 "crlf" Choose 1—3 ⇒")
  (assert (Dar =(read))))
```

Читателю предлагается самостоятельно дописать правила останова программы (halt) по условиям Da=Dar или достижению заданного значения Ds, а также правила для проверки граничных условий. Дотошный читатель, разобравшийся в программе, может спросить, зачем мне *CLIPS*, если такую простую программу я могу написать на любом языке программирования. Во-первых, это учебный пример с простейшей базой знаний. Во-вторых, в реальной жизни база знаний содержит сотни правил, управляющие параметры постоянно считываются с датчиков и видеокамер, и сразу же отрабатывается поиск в сети продукций новых управляющих воздействий. Простой программой в таком случае не обойтись.

Примером более сложной программы для решения задачи планирования последовательности действий робота (лекция 3, рис. 3.4) является фрагмент программы Д. Грим-шоу (www.ryerson.ca/~dgrimsha). Эта программа управления роботом по перекладыванию кубиков. Начальное состояние положения кубиков в стеке 1 и стеке 2 определяется путем перечисления кубиков сверху вниз. Задавая различные комбинации в deffacts initial-state, мы получим конкретные последовательности действий робота.

```
(deftemplate goal
  (slot move)
  (slot on-top-of))
(deffacts initial-state
  (stack A B C)
  (stack D E F)
  (goal (move C) (on-top-of E)))
(defrule move-directly
  ?goal < — (goal (move ?block1) (on-top-of ?block2))
  ?stack-1 < — (stack ?block1 $?rest1)
  ?stack-2 < — (stack ?block2 $?rest2)
  ⇒
(retract ?goal ?stack-1 ?stack-2)
  (assert (stack $?rest1))
  (assert (stack ?block1 ?block2 $?rest2))
  (printout t ?block1 "moved on top of" ?block2 crlf))
(defrule move-to-floor >
?goal < — (goal (move ?block1) (on-top-of floor))
?stack-1 < — (stack ?block1 $?rest)
  ⇒
  (retract ?goal ?stack-1)
  (assert (stack ?block1))
  (assert (stack $?rest))
  (printout t ?block1 "moved to the floor." crlf))
(defrule clear-upper-block
  (goal (move ?block))
  (stack ?top $? ?block $?)
  ⇒
  (assert (goal (move ?top) (on-top-of floor))))
(defrule clear-lower-block
  (goal (on-top-of ?block))
  (stack ?top $? ?block $?)
  ⇒
  (assert (goal (move ?top) (on-top-of floor))))
```

Результат работы *CLIPS* в данном случае будет следующий:

```
CLIPS$>$ (run)
A moved to the floor.
```

```
B moved to the floor.
D moved to the floor.
C moved on top of E
CLIPS$>$
```

Контрольные вопросы:

1. Программирование в CLIPS экспертной системы управления технологическим процессом.
2. Основы программирования в системе CLIPS
3. Приведите примеры программ в системе CLIPS.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Змиртович А.И. Интеллектуальные информационные системы. – Мн.: НТООО «ТетраСистемс». 2021.
2. Попов Э.В. Системы общения и экспертные системы. Искусственный интеллект. В 3-х кн. Кн. 1. Справочник / Под ред. Э.В. Попова – М.: Радио и связь. 2016.
3. Тельнов Ю.Ф. Интеллектуальные информационные системы в экономике. Учебное пособие. -М.: СИНТЕГ. 2020.
4. Turban, McLean, Wetherbe Information Technology for Management: Making Connection for Strategic Advantage 2nd Edition, John Willey & Songs, Inc., New York, 2019.
5. J. Giarratano Expert Systems: Principles and Programming. PWS Publishing Company, Boston, 2017.
6. 18. Барыкин, С.Г. Системы искусственного интеллекта: учебное пособие / С.Г. Барыкин, Н.В. Плотникова. – Челябинск: Изд-во ЮУрГУ, 2004 – 85 с. 19.
7. Барсегян, А.А. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP: учебное пособие для специальности «Информационные системы» / А.А. Барсегян. – СПб.: БХВ Петербург, 2007. – 376 с. 20.
8. Кацко, И.А. Практикум по анализу данных на компьютере: учебное пособие / И.А. Кацко, Н.Б. Паклин. – М.: КолосС, 2009.– 278 с. 21.
9. Паклин, Н.Б. Бизнес-аналитика: от данных к знаниям: учебное пособие / Н.Б. Паклин, В.И. Орешков. – СПб.: Питер, 2013. – 704 с. 22.
10. Поллак, Г.А. Современные технологии анализа информации: учебное пособие / Г.А. Поллак. – Челябинск: Издательский центр ЮУрГУ, 2013 – 115 с. 23.
11. Поллак, Г.А. Современные технологии анализа информации: учебное пособие к практическим работам / Г.А. Поллак. – Челябинск: Издательский центр ЮУрГУ, 2013 – 99 с. 24.
12. Коровин, А.М. Управление знаниями на основе ИТ-технологий: текст лекций / А.М. Коровин. – Челябинск: Издательский центр ЮУрГУ, 2013. – 48 с.

Учебное издание

КОНСПЕКТ ЛЕКЦИЙ
по дисциплине
«ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»
для студентов направления подготовки
Профессиональное обучение (по отраслям),
магистерская программа «Информационные технологии и системы»

С о с т а в и т е л ь:
Тимошенко Дарья Сергеевна

Печатается в авторской редакции.
Компьютерная верстка и оригинал-макет автора.

Подписано в печать _____
Формат 60х841/16. Бумага типограф. Гарнитура Times
Печать офсетная. Усл. печ. л. _____. Уч.-изд. л. _____
Тираж 100 экз. Изд. №_____. Заказ №_____. Цена договорная.

Издательство Луганского государственного
университета имени Владимира Даля

Свидетельство о государственной регистрации издательства
МИ-СРГ ИД 000003 от 20 ноября 2015г.

Адрес издательства: 91034, г. Луганск, кв. Молодежный, 20а
Телефон: 8 (0642) 41-34-12, **факс:** 8 (0642) 41-31-60
E-mail: izdat.lguv.dal@gmail.com **http:** //izdat.dahluniver.ru/