МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКА УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени АБУ РАЙХАН БЕРУНИ

ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

(Учебное пособие)

ТАШКЕНТ -2008

УДК 681.865.8

Основы искусственного интеллекта : Учебное пособие /Х.Н.Назаров
Ташкентский государственный технический университет ,Ташкент 2008 , 85с

В данном учебном пособии описываются принципы построения систем искусственного интеллекта(СИИ); рассматриваются интеллектуальные задачи и структура интеллектуальной

робототехнической системы; архитектура и основные составные части СИИ; логической подход к построению СИИ; экспертной системы и их разновидности, методология построения экспертных систем; модели представления знаний, логические методы представления знаний, фреймы, системы продукций, семантические сети; методы поиска решений; алгоритмы эвристического поиска, методы поиска решений на основе исчисления предикатов; распознавания изображений, методы обучения распознаванию образов; нейронные сети и персептроны ; а также робототехнические системы с элементами искусственного интеллекта.

Учебное пособие предназначено для студентов технических вузов обучающихся по направлению бакалавриатуры 5521800 - «Автоматизация и управление» и магистрантов специальностей 5А521814 - «Управление робототехническими системами и комплексами» и 5А521801 - «Управление в технических системах».

Илл. 23 Табл.2. Библиогр. 20 назв.

Кафедра «Автоматизация и управления »

Печатается по решению научно-методического совета Ташкентского государственного технического университета имени Абу Райхана Беруни.

Рецензенты: д.т.н., проф .,зав.кафедрой Ташкентского Университета Информационных технологий Каримов М.М, д.т.н. проф. Азимов Р.К., проф кафедры «Метрология, стандартизация и сертификация» Ташкентского государственного технического университета**.**

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ. ПОНЯТИЕ О СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА.

В современном мире прогресс производительности программиста практически достигается только в тех случаях, когда часть интеллектуальной нагрузки берут на себя компьютеры. Одним из способов достигнуть максимального прогресса в этой области, является "искусственный интеллект", когда компьютер берет на себя не только однотипные, многократно повторяющиеся операции, но и сам сможет обучаться. Кроме того, создание полноценного "искусственного интеллекта" открывает перед человечеством новые горизонты развития.

Целью изучения дисциплины является подготовка специалистов в области автоматизации сложно формализуемых задач, которые до сих пор считаются прерогативой

человека. Задачей изучения дисциплины является приобретение знаний о способах мышления человека, а так же о методах их реализации на компьютере.

Основным предметом изучения дисциплины являются мыслительные способности человека и способы их реализации техническими средствами.

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.

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 Дартмутском колледже (США). Приведем некоторые из этих определений. Д. Люггер 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».

Введем определения, которые будем использовать в данной книге в качестве рабочих определений. 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.

Системой будем называть 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].

Под алгоритмом будем понимать последовательность заданных действий, которые однозначно определены и выполнимы на современных ЭВМ за приемлемое время для решаемой задачи. Под ***ИС*** будем понимать *адаптивную систему*, 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).
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.

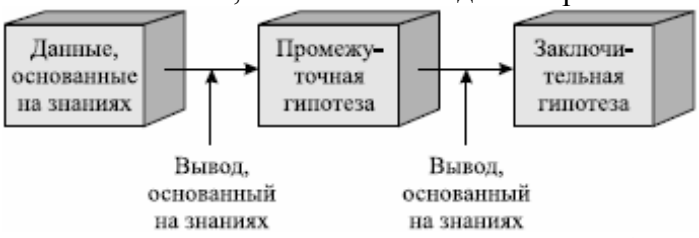


Рис.1. Процесс логического вывода в ИС

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.

2. ТЕРМИНОЛОГИЯ ПО СИИ. ИСТОРИЧЕСКИЙ ОБЗОР РАБОТ В ОБЛАСТИ ИИ.

Термин интеллект (intelligence) происходит от латинского intellectus — что означает ум, рассудок, разум; мыслительные способности человека. Соответственно искусственный интеллект (artificial intelligence) — ИИ (AI) обычно толкуется как свойство автоматических систем брать на себя отдельные функции интеллекта человека, например, выбирать и принимать оптимальные решения на основе ранее полученного опыта и рационального анализа внешних воздействий.

Мы, в нашем курсе, интеллектом будем называть способность мозга решать (интеллектуальные) задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.

В этом определении под термином "знания" подразумевается не только ту информацию, которая поступает в мозг через органы чувств. Такого типа знания чрезвычайно важны, но

недостаточны для интеллектуальной деятельности. Дело в том, что объекты окружающей нас среды обладают свойством не только воздействовать на органы чувств, но и находиться друг с другом в определенных отношениях. Ясно, что для того, чтобы осуществлять в окружающей среде интеллектуальную деятельность (или хотя бы просто существовать), необходимо иметь в системе знаний модель этого мира. В этой информационной модели окружающей среды реальные объекты, их свойства и отношения между ними не только отображаются и запоминаются, но и, как это отмечено в данном определении интеллекта, могут мысленно "целенаправленно преобразовываться". При этом существенно то, что формирование модели внешней среды происходит "в процессе обучения на опыте и адаптации к разнообразным обстоятельствам".

Мы употребили термин интеллектуальная задача. Для того, чтобы пояснить, чем отличается интеллектуальная задача от просто задачи, необходимо ввести термин "алгоритм" — один из краеугольных терминов кибернетики.

Под алгоритмом понимают точное предписание о выполнении в определенном порядке системы операций для решения любой задачи из некоторого данного класса (множества) задач. Термин "алгоритм" происходит от имени узбекского математика Аль-Хорезми, который еще в IX веке предложил простейшие арифметические алгоритмы. В математике и кибернетике класс задач определенного типа считается решенным, когда для ее решения установлен алгоритм. Нахождение алгоритмов является естественной целью человека при решении им разнообразных классов задач. Отыскание алгоритма для задач некоторого данного типа связано с тонкими и сложными рассуждениями, требующими большой изобретательности и высокой квалификации. Принято считать, что подобного рода деятельность требует участия интеллекта человека. Задачи, связанные с отысканием алгоритма решения класса задач определенного типа, будем называть интеллектуальными.

Что же касается задач, алгоритмы решения которых уже установлены, то, как отмечает известный специалист в области ИИ М. Минский, "излишне приписывать им такое мистическое свойства, как "интеллектуальность". В самом деле, после того, как такой алгоритм уже найден, процесс решения соответствующих задач становится таким, что его могут в точности выполнить человек, вычислительная машина (должным образом запрограммированная) или робот, не имеющие ни малейшего представления о сущность самой задачи. Требуется только, чтобы лицо, решающее задачу, было способно выполнять те элементарные операции, их которых складывается процесс, и, кроме того, чтобы оно педантично и аккуратно руководствовалось предложенным алгоритмом. Такое лицо, действуя, как говорят в таких случаях, чисто машинально, может успешно решать любую задачу рассматриваемого типа.

Поэтому представляется совершенно естественным исключить их класса интеллектуальных такие задачи, для которых существуют стандартные методы решения. Примерами таких задач могут служить чисто вычислительные задачи: решение системы линейных алгебраических уравнений, численное интегрирование дифференциальных уравнений и т. д. Для решения подобного рода задач имеются стандартные алгоритмы, представляющие собой определенную последовательность элементарных операций, которая может быть легко реализована в виде программы для вычислительной машины. В противоположность этому для широкого класса интеллектуальных задач, таких, как распознавание образов, игра в шахматы, доказательство теорем и т. п., напротив это формальное разбиение процесса поиска решения на отдельные элементарные шаги часто оказывается весьма затруднительным, даже если само их решение несложно.

Таким образом, мы можем перефразировать определение интеллекта как универсальный сверхалгоритм, который способен создавать алгоритмы решения конкретных задач.

Еще интересным замечанием здесь является то, что профессия программиста, исходя из наших определений, является одной из самых интеллектуальных, поскольку продуктом деятельности программиста являются программы — алгоритмы в чистом виде. Именно поэтому, создание даже элементов ИИ должно очень сильно повысить производительность его труда.

Деятельность мозга (обладающего интеллектом), направленную на решение интеллектуальных задач, мы будем называть мышлением, или интеллектуальной деятельностью. Интеллект и мышление органически связаны с решением таких задач, как доказательство теорем, логический анализ, распознавание ситуаций, планирование поведения, игры и управление в условиях неопределенности. Характерными чертами интеллекта, проявляющимися в процессе решения задач, являются способность к обучению, обобщению, накоплению опыта (знаний и навыков) и адаптации к изменяющимся условиям в процессе решения задач. Благодаря этим качествам интеллекта мозг может решать разнообразные задачи, а также легко перестраиваться с решения одной задачи на другую. Таким образом, мозг, наделенный интеллектом, является универсальным средством решения широкого круга задач (в том числе неформализованных) для которых нет стандартных, заранее известных методов решения.

Следует иметь в виду, что существуют и другие, чисто поведенческие (функциональные) определения. Так, по А. Н. Колмогорову, любая материальная система, с которой можно достаточно долго обсуждать проблемы науки, литературы и искусства, обладает интеллектом. Другим примером поведенческой трактовки интеллекта может служить известное определение А. Тьюринга. Его смысл заключается в следующем. В разных комнатах находится люди и машина. Они не могут видеть друг друга, но имеют возможность обмениваться информацией (например, с помощью электронной почты). Если в процессе диалога между участниками игры людям не удается установить, что один из участников — машина, то такую машину можно считать обладающей интеллектом.

Кстати интересен план имитации мышления, предложенный А. Тьюрингом. "Пытаясь имитировать интеллект взрослого человека, — пишет Тьюринг, — мы вынуждены много размышлять о том процессе, в результате которого человеческий мозг достиг своего настоящего состояния… Почему бы нам вместо того, чтобы пытаться создать программу, имитирующую интеллект взрослого человека, не попытаться создать программу, которая имитировала бы интеллект ребенка? Ведь если интеллект ребенка получает соответствующее воспитание, он становится интеллектом взрослого человека… Наш расчет состоит в том, что устройство, ему подобное, может быть легко запрограммировано… Таким образом, мы расчленим нашу проблему на две части: на задачу построения "программы-ребенка" и задачу "воспитания" этой программы".

Забегая вперед, можно сказать, что именно этот путь используют практически все системы ИИ. Ведь понятно, что практически невозможно заложить все знания в достаточно сложную систему. Кроме того, только на этом пути проявятся перечисленные выше признаки интеллектуальной деятельности (накопление опыта, адаптация и т. д.).

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, Стэндфордский университет) расшифровывала данные масс-спектрографического анализа.

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

ЭС PROSPECTOR (1974-1983 годы, Стэндфордский 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. Об этом мы будем говорить подробнее в 6 и 7 лекциях.

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*.

3.ИНТЕЛЛЕКТУЛЬНЫЕ МОДЕЛИ И ЗАДАЧИ В ТЕОРИИ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Современные достижения моделирования на ЭВМ, автоматизация проектирования и новой информационной технологии требуют рассмотрения основных задач (описания, анализа и синтеза) теории робототехнических систем с общих позиций теории алгебраических систем и искусственного интеллекта. Анализ и синтез сводится к выбору типа алгебры выполнению процедур, связанных с прямыми отображениями исследуемой системы, (объекта) в множество моделей состояния объекта, модели состояний объекта в носитель алгебры через операции алгебры и алгебраической. модели в совокупность новых данных и знаний об объекте и обратными отображениями (рис.2).

Рис.2 Составные части интеллектуальной модели.

Если, в прямых методах анализа (синтеза) систем используются прямые отображения $\varphi_{OH}, \varphi_{H\Omega}, \varphi_{A3,}$ (обратные отображения φ_{OH}^{-1}, $\varphi_{H\Omega}^{-1}$, φ_{A3}^{-1}, то непрямых методах анализа (синтеза) - и прямые? и обратные отображения. Поэтому методы анализа и синтеза цепей и систем главным образом, различаются типами моделей состояний, алгебр ($<H, \Omega>$) и испольэуемми видами отображений, что и определяет главенствующее место 'моделей состояний алгебр и способов реализации отображений в теории цепей и систем*

Алгебраическая модель (алгебра <H, Ω >о выделенными в ее носителе H моделями состояний) объекта, является наиболее подходящей моделью для реализации на ЭВМ, но несмотря на наличие в ней мощного набора операций (в том числе, программ процедур) над моделями состояний объектов для решения вычислительных задачу она становится бессильной для автоматического решения интеллектуальных задач (задач с априори неизвестными алгоритмами решения).

Этого можно избежать, дополнив алгебраическую модель объекта интеллектуальными знаниями. Последнее в системах искусственного интеллекта составляет содержание планировщика призванного имитировать элементы мыслительной деятельности человека. Полученные таким образом* модели объектов в работе названы интеллектуальными. В них модели целевых, исходных и промежуточных состояний объекта соответствуют базе данных и целей, а выражения операций базе знаний интеллектуальных систем.

По отношению к интеллектуальной модели (рис.I) задача является неинтеллектуальной (элементарной), если в наборе операций (в базе знаний) модели имеется соответствующая алгоритму решения. поставленной задачи операция (схема решения программа, процедура).В противном случае задача воспринимается моделирующей системой интеллектуальной в виду отсутствия в базе знаний модели готовой схемы решения задачи. В дальнейшем будем полагать что при построении интеллектуальных моделей в базу знаний введены все известные в данной предметной области алгоритмы. Тогда к интеллектуальным относятся те задачи, для которых не существует готовых схем решения в рассматриваемой предметной области А

Формально интеллектуальную задачу в области теории робототехнических систем будем представлять четверкой /I/

$$ИЗ = < M_{Ц}, M_{И}, \Omega_{З}, \rho >,$$

где $M_{Ц}$ – модель целевых состояний объекта, моделирования;

$M_{и}$ - модель исходного состояния объекта;

$\Omega_{З} \leq \Omega$ - набор операции в множестве состояний объекта моделирования используемых для решения данной задачи; ρ -план (алгоритм, программа, схема решения) задач.

$M_{и}$ представляется в виде непустого набора выражений не которого языка Я, на котором представлен носитель Н алгебраической модели(рис.I).

$M_{Ц}$ может быть задана либо как множество моделей: состояний, либо как модель единственного состояния. В последнем случае $M_{Ц}$ представляется как непустой набор выражений на языке Я (т.е на том же уровне представления, что и $M_{и}$). В случае задания $M_{Ц}$ как множество моделей, она может быть представлена либо перечислением элементов $M_{Ц}$, представленных на языке Я, либо указанием свойства множества $M_{Ц}$ на языке более высокого уровня, чем язык представления $M_{и}$.

Элементы набора операций C?$_5$ также как другие элементы набора операций *Q* алгебраической модели (рис.I) задаются в форме "операнды -результат", где операнды и результат подмножества множества моделей состояний, представленных на языке *Я.*

План ρ представляется как упорядоченная совокупность пар имен операций и операндов. Она в интеллектуальной задаче всегда априори неизвестна и находится в результате моделирования задачи.

Интеллектуальные задачи, в зависимости от используемых уровней языков для представления $M_{Ц}$ и $M_{и}$ могут быть одноуровневыми или двухуровневыми, это определяется степенью известное $M_{Ц}$ и $M_{и}$-задачами с одним (когда известны $M_{и}$ и $M_{Ц}$)или двумя (когда известны $M_{Ц}$ и $M_{и}$) закрепленными концам В задачах с двумя закрепленными концами новым знанием в результате моделирования является схема решения задачи, а в задачах с двумя закрепленным концом - новыми знаниями и данными будут и схема решения задачи, и найденное неизвестное состояние объекта $M_{Ц}$ и $M_{и}$

Интеллектуальная модель должна содержать предметные интеллектуальные знания соответственно. Предметные знания условимся представлять набором операций 52 в множестве состояний объектов предметной области, Другими неотъемлемыми компонентами интел-лектуальной модели, являются наличные условия $M_{и}, M_{Ц}$) решаемой в процессе моделирования интеллектуальной задачи, которые являются подмножествами или элементами объединения множеств состояний объектов предметной, области. Последнее рассматривается как основное множество (носитель Н) некоторой алгебры, алгебры, которой отображают предметные знания.

На основе приведенных соображений интеллектуальную модель будем представлять как

$$ИМ = < H, M_{и}, M_{ц}\Omega, З_{и} >$$

При этом полагается существование некоторого многоуровневого языка Я, для задания компонент интеллектуальной модели /2/. Здесь, Н и $М_Ц$ и операнды и результаты операций из представляются выражениями языка Я на едином уровне. Уровень задания $М_Ц$ в общем случае отличается от уровня задания носителя Н алгебры.
Лишь для одноуровневых задач с двумя закрепленными концами задается на одном уровне с $М_и$.

4. О РАЗВИТИИ РОБОТОТЕХНИКИ. СТРУКТУРНАЯ СХЕМА ИНТЕЛЛЕКТУАЛЬНОЙ РОБОТОТЕХНИЧЕСКОЙ СИСТЕМЫ

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. 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. 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 Иокогама, прошла четвертая по счету выставка роботов «Robodex» (рис. 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».

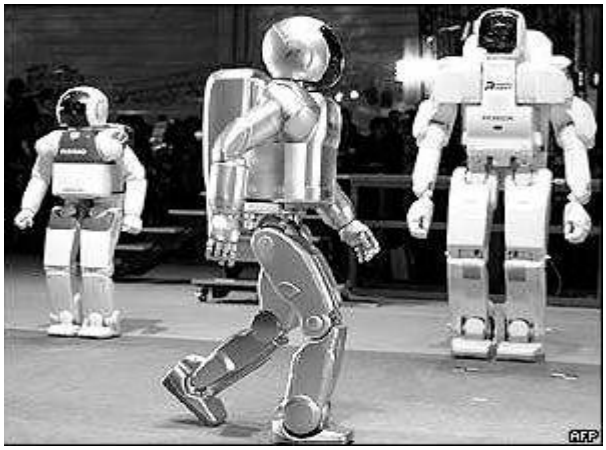


Рис.4. На выставке «Robodex»

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 Все это подтверждает уверенность в том, что самые интересные достижения в этой области еще впереди.

Структурная схема интеллектуальной робототехнической системы

Рассмотрим подробнее систему управления *ИРС*, структурная схема которой представлена на рис.5. 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*).

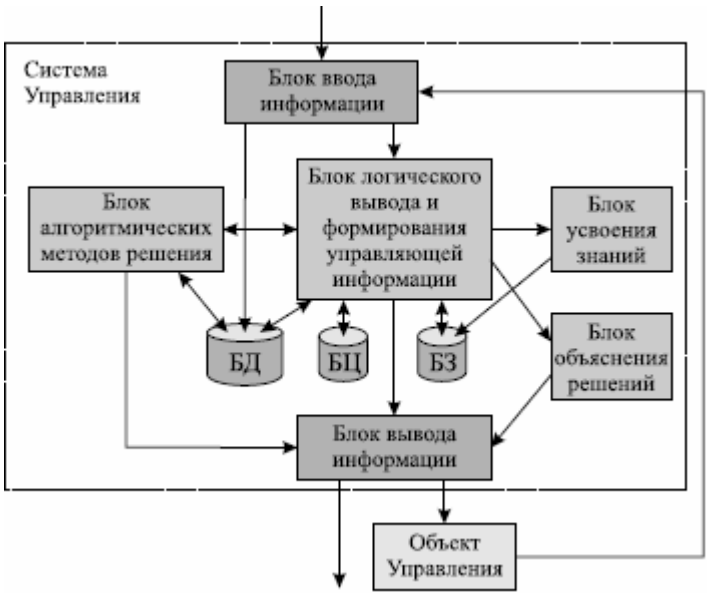


Рис. 5. Структурная схема интеллектуальной робототехнической системы

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*.

В заключение лекции сформулируем **основные темы** *ИИ*, на которые будет обращено внимание в последующих лекциях 2-7 исходя **из задач создания интеллектуальных робототехнических систем**:

- модели представления *знаний* предметной области;
- методы выбора и принятия решений в условиях неопределенности;
- обработка изображений, являющихся значительным и важным потоком входной информации в робототехнические системы и комплексы;
- общение с ЭВМ на естественном языке и его подмножествах;

экспертные системы, как инструмент создания *интеллектуальных систем* управления робототехническими системами и комплексами

5. АРХИТЕКТУРА И ОСНОВНЫЕ СОСТАВНЫЕ ЧАСТИ СИСТЕМ ИИ. РАЗЛИЧНЫЕ ПОДХОДЫ К ПОСТРОЕНИЮ СИСТЕМ ИИ

Существуют различные подходы к построению систем ИИ. Это разделение не является историческим, когда одно мнение постепенно сменяет другое, и различные подходы существуют и сейчас. Кроме того, поскольку по-настоящему полных систем ИИ в настоящее время нет, то нельзя сказать, что какой-то подход является правильным, а какой-то ошибочным. Для начала кратко рассмотрим **логический подход**. Почему он возник? Ведь человек занимается отнюдь не только логическими измышлениями. Это высказывание конечно верно, но именно способность к логическому мышлению очень сильно отличает человека от животных.

Основой для данного логического подхода служит Булева алгебра. Каждый программист знаком с нею и с логическими операторами с тех пор, когда он осваивал оператор IF. Свое

дальнейшее развитие Булева алгебра получила в виде исчисления предикатов — в котором она расширена за счет введения предметных символов, отношений между ними, кванторов существования и всеобщности. Практически каждая система ИИ, построенная на логическом принципе, представляет собой машину доказательства теорем. При этом исходные данные хранятся в базе данных в виде аксиом, правила логического вывода как отношения между ними. Кроме того, каждая такая машина имеет блок генерации цели, и система вывода пытается доказать данную цель как теорему. Если цель доказана, то трассировка примененных правил позволяет получить цепочку действий, необходимых для реализации поставленной цели. Мощность такой системы определяется возможностями генератора целей и машиной доказательства теорем.

Конечно можно сказать, что выразительности алгебры высказываний не хватит для полноценной реализации ИИ, но стоит вспомнить, что основой всех существующих ЭВМ является бит — ячейка памяти, которая может принимать значения только 0 и 1. Таким образом было бы логично предположить, что все, что возможно реализовать на ЭВМ, можно было бы реализовать и в виде логики предикатов. Хотя здесь ничего не говорится о том, за какое время.
Добиться большей выразительности логическому подходу позволяет такое сравнительно новое направление, как нечеткая логика. Основным ее отличием является то, что правдивость высказывания может принимать в ней кроме да/нет (1/0) еще и промежуточные значения — не знаю (0.5), пациент скорее жив, чем мертв (0.75), пациент скорее мертв, чем жив (0.25). Данный подход больше похож на мышление человека, поскольку он на вопросы редко отвечает только да или нет. Хотя правда на экзамене будут приниматься только ответы из разряда классической булевой алгебры.

Для большинства логических методов характерна большая трудоемкость, поскольку во время поиска доказательства возможен полный перебор вариантов. Поэтому данный подход требует эффективной реализации вычислительного процесса, и хорошая работа обычно гарантируется при сравнительно небольшом размере базы данных.
Под **структурным подходом** мы подразумеваем здесь попытки построения ИИ путем моделирования структуры человеческого мозга. Одной из первых таких попыток был перцептрон Френка Розенблатта. Основной моделируемой структурной единицей в перцептронах (как и в большинстве других вариантов моделирования мозга) является нейрон.
Позднее возникли и другие модели, которые в простонародье обычно известны под термином "нейронные сети" (НС). Эти модели различаются по строению отдельных нейронов, по топологии связей между ними и по алгоритмам обучения. Среди наиболее известных сейчас вариантов НС можно назвать НС с обратным распространением ошибки, сети Хопфилда, стохастические нейронные сети.

НС наиболее успешно применяются в задачах распознавания образов, в том числе сильно зашумленных, однако имеются и примеры успешного применения их для построения собственно систем ИИ, это уже ранее упоминавшийся ТАИР.
Для моделей, построенных по мотивам человеческого мозга характерна не слишком большая выразительность, легкое распараллеливание алгоритмов, и связанная с этим высокая производительность параллельно реализованных НС. Также для таких сетей характерно одно свойство, которое очень сближает их с человеческим мозгом — нейронные сети работают даже при условии неполной информации об окружающей среде, то есть как и человек, они на вопросы могут отвечать не только "да" и "нет" но и "не знаю точно, но скорее да".

Довольно большое распространение получил и **эволюционный подход**. При построении систем ИИ по данному подходу основное внимание уделяется построению начальной модели, и правилам, по которым она может изменяться (эволюционировать). Причем модель может быть составлена по самым различным методам, это может быть и НС и набор логических правил и любая другая модель. После этого мы включаем компьютер и он, на основании проверки моделей отбирает самые лучшие из них, на основании которых по самым различным правилам генерируются новые модели, из которых опять выбираются самые лучшие и т. д.

В принципе можно сказать, что эволюционных моделей как таковых не существует, существует только эволюционные алгоритмы обучения, но модели, полученные при эволюционном подходе имеют некоторые характерные особенности, что позволяет выделить их в отдельный класс.

Такими особенностями являются перенесение основной работы разработчика с построения модели на алгоритм ее модификации и то, что полученные модели практически не сопутствуют извлечению новых знаний о среде, окружающей систему ИИ, то есть она становится как бы вещью в себе.

Еще один широко используемый подход к построению систем ИИ — **имитационный**. Данный подход является классическим для кибернетики с одним из ее базовых понятий — "черным ящиком" (ЧЯ). ЧЯ — устройство, программный модуль или набор данных, информация о внутренней структуре и содержании которых отсутствуют полностью, но известны спецификации входных и выходных данных. Объект, поведение которого имитируется, как раз и представляет собой такой "черный ящик". Нам не важно, что у него и у модели внутри и как он функционирует, главное, чтобы наша модель в аналогичных ситуациях вела себя точно так же.

Таким образом здесь моделируется другое свойство человека — способность копировать то, что делают другие, не вдаваясь в подробности, зачем это нужно. Зачастую эта способность экономит ему массу времени, особенно в начале его жизни.

Основным недостатком имитационного подхода также является низкая информационная способность большинства моделей, построенных с его помощью.

С ЧЯ связана одна очень интересная идея. Кто бы хотел жить вечно? Я думаю, что почти все ответят на этот вопрос "я".

Представим себе, что за нами наблюдает какое-то устройство, которое следит за тем, что в каких ситуациях мы делаем, говорим. Наблюдение идет за величинами, которые поступают к нам на вход (зрение, слух, вкус, тактильные, вестибулярные и т. д.) и за величинами, которые выходят от нас (речь, движение и др.). Таким образом человек выступает здесь как типичный ЧЯ.

Далее это устройство пытается отстроить какую-то модель таким образом, чтобы при определенных сигналах на входе человека, она выдавала на выходе те же данные, что и человек. Если данная затея будет когда-нибудь реализована, то для всех посторонних наблюдателей такая модель будет той же личностью, что и реальный человек. А после его смерти она, будет высказывать те мысли, которые предположительно высказывал бы и смоделированный человек.

Мы можем пойти дальше и скопировать эту модель и получить брата близнеца с точно такими же "мыслями".

Можно сказать, что "это конечно все интересно, но при чем тут я? Ведь эта модель только для других будет являться мной, но внутри ее будет пустота. Копируются только внешние атрибуты, но я после смерти уже не буду думать, мое сознание погаснет (для верующих людей слово "погаснет" необходимо заменить на "покинет этот мир") ". Что ж это так. Но попробуем пойти дальше.

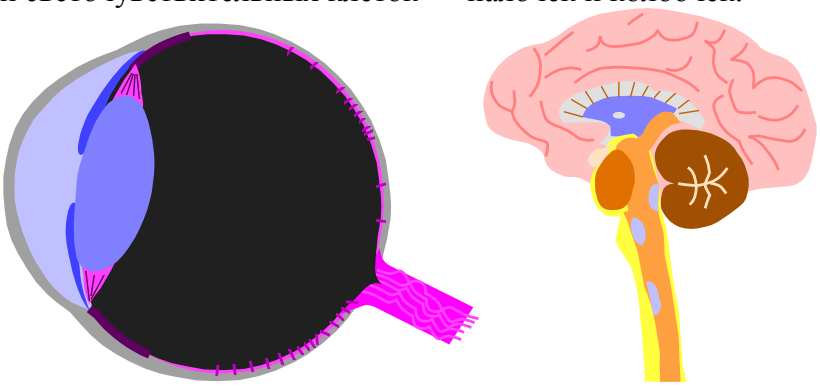
Согласно философским представлениям автора данного курса, сознание представляет собой сравнительно небольшую надстройку над нашим подсознанием, которая следит за активностью некоторых центров головного мозга, таких как центр речи, конечной обработки зрительных образов, после чего "возвращает" эти образы на начальные ступени обработки данной информации. При этом происходит повторная обработка этих образов, мы как бы видим и слышим, что думает наш мозг. При этом появляется возможность мысленного моделирования окружающей действительности при нашем "активном" участии в данном процессе. И именно наш процесс наблюдения за деятельностью этих немногих центров является тем, что мы называем сознанием. Если мы "видим" и "слышим" наши мысли, мы в сознании, если нет, то мы находимся в бессознательном состоянии.

Если бы мы смогли смоделировать работу именно этих немногих "сознательных" нервных центров (работа которых правда основана на деятельности всего остального мозга) в качестве одного ЧЯ, и работу "супервизора" в качестве другого ЧЯ, то можно было бы с уверенностью говорить, что "да, данная модель думает, причем так же, как и я". Здесь я ничего не хочу говорить о том, как получить данные о работе этих нервных центров, поскольку на мой взгляд сегодня нет ничего такого, что позволило бы следить за мозгом человека годами и при этом не мешало бы его работе и жизни.

И заканчивая беглое ознакомление с различными методами и подходами к построению систем ИИ, хотелось бы отметить, что на практике очень четкой границы между ними нет. Очень часто встречаются смешанные системы, где часть работы выполняется по одному типу, а часть по другому.

Вспомогательные системы нижнего уровня (распознавание образов зрительных и звуковых, идентификация, моделирование, жесткое программирование) и их место в системах ИИ

Для того, чтобы человек сознательно воспринял информацию (для примера возьмем чертеж), она должна пройти довольно длительный цикл предварительной обработки. Вначале свет попадает в глаз. Пройдя через всю оптическую систему фотоны в конце концов попадают на сетчатку — слой светочувствительных клеток — палочек и колбочек.



Уже здесь — еще очень далеко от головного мозга, происходит первый этап обработки информации, поскольку, например, у млекопитающих, сразу за светочувствительными клетками находится обычно два слоя нервных клеток, которые выполняют сравнительно несложную обработку.

Теперь информация поступает по зрительному нерву в головной мозг человека, в так называемые "зрительные бугры". То, что именно сюда приходит видеоинформация для дальнейшей обработки, показывают многочисленные опыты над людьми во время различных операций, в ходе которых производилась трепанация черепа. При этом пациентам раздражали область зрительных бугров слабым электрическим полем, что вызывало у них различные световые галлюцинации. Причем, что интересно, при изменении места раздражения, пропорционально смещению, смещались и места галлюцинаций, т. е. на зрительные бугры как бы проецируется то, что мы видим.

Некоторые исследователи пошли дальше, и вживляли слепым людям целую матрицу электродов, напряжения на которых соответствовали освещенности соответствующих участков видеокамеры, размещенной на голове пациента. После операции, слепые начинали различать крупные фигуры (квадрат, треугольник, круг) и даже читать текст (при вживлении матрицы 10*10). Широкому распространению данного метода лечения слепоты препятствуют как недостаточно высокий наш технический уровень, так и чрезвычайно высокая опасность операций на открытом мозге. Такого рода опыты проводятся только попутно с операцией, вызванной другими причинами.

Далее зрительная информация поступает в отделы мозга, которые уже выделяют из нее отдельные составляющие — горизонтальные, вертикальные, диагональные линии, контуры, области светлого, темного, цветного. До этих пор мы можем без труда смоделировать работу

мозга применяя различные графические фильтры. Постепенно образы становятся все более сложными и размытыми, но графический образ картины пройдет еще долгий путь, прежде чем достигнет уровня сознания. Причем на уровне сознания у нас будет не только зрительный образ, к нему примешаются еще и звуки, запахи (если картина представляет собой натюрморт) и вкусовые ощущения. Дальнейшие ассоциации каждый может додумать сам.

Смысл всего сказанного заключается в том, чтобы показать, что в системах ИИ имеются подсистемы, которые мы уже сейчас можем реализовать даже не зная о том, как они реализованы у человека. Причем можем это сделать не хуже, чем у прототипа, а зачастую и лучше. Например, искусственный глаз (а равно и блок первичной обработки видеоинформации, основанные на простейших фильтрах или др. сравнительно несложных устройствах) не устает, может видеть в любом диапазоне волн, легко заменяется на новый, видит при свете звезд.
Устройства обработки звука позволяют улавливать девиацию голоса человека в 1-2 Герца. Данное изменение частоты происходит при повышенном возбуждении вегетативной нервной системы, которое в свою очередь часто обусловлено волнением человека. На данном принципе основаны современные детекторы лжи, которые позволяют обнаружить с высокой вероятностью даже записанные на пленку много лет назад ложные высказывания.

Современные системы управления электродвигателем позволяют с высокой точностью держать заданные координаты даже при ударном изменении нагрузки. А ведь это примерно тоже, что держать на длинной палке баскетбольный мяч, по которому то слева, то справа кидают теннисные мячи.

За одно и тоже время, компьютер произведет гораздо больше арифметических операций и с большей точностью, чем человек.

Антиблокировочная система на автомобилях позволяет держать тормоза на грани заклинивания колеса, что дает наибольшее трение с дорогой, а это без АБС по силам только очень опытным водителям.

В принципе такие примеры, где техника оказывается ничуть не хуже человека, можно продолжать до бесконечности. Общий же смысл сказанного в том, что при конструировании ИИ, мы не связаны одним набором элементарных составляющих, как природа. В каждом конкретном случае желательно применять то, что даст самый большой эффект. В той области, где у человека господствуют рефлексы (чихание, быстрое напряжение быстро растягиваемой мышцы, переваривание пищи, регулировка температуры), мы вообще можем применить жесткие системы управления, с раз и навсегда заданным алгоритмом функционирования. При этом вполне можно ожидать увеличения точности и уменьшение времени обучения их до нуля. При этом ядро нашей системы ИИ будет решать уже не настолько глобальные задачи.

Данный принцип разбиения задачи на подзадачи уже давно используется природой. К примеру, мы далеко не полностью используем все возможности наших мышц в области разнообразия движений. Мы не можем заставить наши глаза смотреть в разные стороны, не говоря уже о том, чтобы делать это на разном уровне (левый глаз — влево-вверх, правый — вправо-вниз). При ходьбе мы часто используем далеко не оптимальный набор движений и далеко не все сочетания вариантов напряжения мышц мы опробуем. Попробуйте к примеру сделать волну животом. В принципе здесь нет ничего сложного, поскольку каждый пучок мышц пресса иннервируется отдельно, но если Вы этого не делали ранее, то получить необходимый результат будет не просто — в повседневной жизни это действие ненужно, а значит его нет и в "словаре движений", а на обучение необходимо определенное время. А по поводу оптимальности походки существуют расчеты, что если бы человек всегда рассчитывал оптимально траекторию движения в которой существует более 200 степеней свобод, то он бы не ходил, а в основном бы только думал о том, как надо ходить.

На самом деле наша система управления построена по иерархическому принципу, когда задача распределяется между несколькими уровнями. Высший уровень нервной системы (связанный с большими полушариями мозга) ставит лишь общую задачу, скажем, переложить книгу на стол. Этот уровень вообще не контролирует действие отдельных двигательных

единиц, направленных на решение поставленной задачи. Здесь уместна аналогия: командующий армией, ставя перед своими войсками некую общую задачу, отнюдь не предписывает каждому солдату и офицеру, что именно он должен делать в каждый момент операции.

Детализация построения движений у человека происходит на уровнях более низких, чем командный уровень коры больших полушарий. Более того, в некоторых случаях (когда мы отдергиваем руку, прикоснувшись к горячему предмету, даже не успев осознать ситуацию) все управление формируется на нижележащих уровнях, связанных с различными отделами спинного мозга.

В общем ситуация схожа с той, когда программист использует библиотеку подпрограмм. При этом ему не важно, какой алгоритм они используют, если программа работает нормально. А на написание своей библиотеки тратится драгоценное время. Кроме того, еще не известно, будет ли она работать так же хорошо.

Общий вывод данной лекции состоит в том, что в настоящее время существуют методы, алгоритмы и устройства, которые позволяют нам довольно неплохо смоделировать нижние уровни человеческого интеллекта, причем совсем не обязательно на таком же физическом принципе. И если бы это была не лекция, а тост, то я бы закончил его: " …так выпьем же за протестированные, правильно работающие и бесплатные библиотеки подпрограмм".

6. ЛОГИЧЕСКИЙ ПОДХОД К ПОСТРОЕНИЮ СИСТЕМ ИИ.

НЕФОРМАЛЬНЫЕ ПРОЦЕДУРЫ

Говоря о неформальных процедурах мы обычно хорошо понимаем, что имеется в виду, и без затруднений можем привести примеры таких процедур, связанных с пониманием текстов естественного языка, переводом с одного естественного языка на другой, информационным поиском по смыслу и т. д.

Трудности возникают при попытке точного определения подобных процедур. Так, если рассматривать неформальные процедуры всего лишь как абстрактные функции, которые для каждого значения аргумента "выдают" некоторое значение, то категория неформальности вообще исчезает из рассмотрения.

Неформальная процедура — это особый способ представления функций. Чтобы в какой-то степени приблизиться к этому "человеческому" способу представления функций,

рассмотрим прежде всего традиционные алгоритмические модели и попытаемся понять, в чем состоит основная трудность их применения для имитации неформальных процедур.

Алгоритмические модели

Алгоритмические модели основаны на понятии алгоритма. Исторически первые точные определения алгоритма, возникшие в 30-х годах, были связаны с понятием вычислимости. С тех пор было предложено множество, как выяснилось, эквивалентных определений алгоритма.

В практике программирования алгоритмы принято описывать с помощью алгоритмических языков программирования. Широко используются также разного рода блок-схемы алгоритмов, позволяющие представить алгоритмы в наглядном и общедоступном виде, не привлекая в тоже время сложных конструкций из конкретных языков программирования.

Пролог

Синтаксис

1. Термы

Объекты данных в Прологе называются *термами.* Терм может быть *константой, переменной* или *составным* термом (структурой). Константами являются целые и действительные числа, например:

0, -l, 123.4, 0.23E-5,

(некоторые реализации Пролога не поддерживают действительные числа).

К константам относятся также атомы, такие, как:

голди, а, атом, +, :, 'Фред Блогс', [].

Атом есть любая последовательность символов, заключенная в одинарные кавычки. Кавычки опускаются, если и без них атом можно отличить от символов, используемых для обозначения переменных. Приведем еще несколько примеров атомов:

abcd, фред, ':', Джо.

Полный синтаксис атомов описан ниже.

Как и в других языках программирования, константы обозначают конкретные элементарные объекты, а все другие типы данных в Прологе составлены из сочетаний констант и переменных.

Имена переменных начинаются с заглавных букв или с символа подчеркивания "_". Примеры переменных:

X, Переменная, _3, _переменная.

Если переменная используется только один раз, необязательно называть ее. Она может быть записана как *анонимная* переменная, состоящая из одного символа подчеркивания "_". Переменные, подобно атомам, являются элементарными объектами языка Пролог.

Завершает список синтаксических единиц сложный терм, или структура. Все, что не может быть отнесено к переменной или константе, называется сложным термом. Следовательно, сложный терм состоит из констант и переменных.

Теперь перейдем к более детальному описанию термов.

2. Константы

Константы известны всем программистам. В Прологе константа может быть атомом или числом.

3. Atom

Атом представляет собой произвольную последовательность символов, заключенную в одинарные кавычки. Одинарный символ кавычки, встречающийся внутри атома, записывается дважды. Когда атом выводится на печать, внешние символы кавычек обычно не печатаются. Существует несколько исключений, когда атомы необязательно записывать в кавычках. Вот эти исключения:

1) атом, состоящий только из чисел, букв и символа подчеркивания и начинающийся со строчной буквы;

2) атом, состоящий целиком из специальных символов. К специальным символам относятся:

+ - * / ^ = : ; ? @ $ &

Заметим, что атом, начинающийся с /*, будет воспринят как начало комментария, если он не заключен в одинарные кавычки.

Как правило, в программах на Прологе используются атомы без кавычек.

Атом, который необязательно заключать в кавычки, может быть записан и в кавычках.

Запись с внешними кавычками и без них определяет один и тот же атом.

Внимание: допустимы случаи, когда атом не содержит ни одного символа (так называемый *'нулевой атом')* или содержит непечатаемые символы. (В Прологе имеются предикаты для построения атомов, содержащих непечатаемые или управляющие символы.) При выводе таких атомов на печать могут возникнуть ошибки.

4. Числа

Большинство реализации Пролога поддерживают целые и действительные числа. Для того чтобы выяснить, каковы диапазоны и точность, чисел следует обратиться к руководству по конкретной реализации.

5. Переменные

Понятие переменной в Прологе отличается от принятого во многих языках программирования. Переменная не рассматривается как выделенный участок памяти. Она служит для обозначения объекта, на который нельзя сослаться по имени. Переменную можно считать локальным именем для некоторого объекта.

Синтаксис переменной довольно прост. Она должна начинаться с прописной буквы или символа подчеркивания и содержать только символы букв, цифр и подчеркивания.

Переменная, состоящая только из символа подчеркивания, называется анонимной и используется в том случае, если имя переменной несущественно.

6. Область действия переменных

Областью действия переменной является утверждение. В пределах утверждения одно и то же имя принадлежит одной и той же переменной. Два утверждения могут использовать одно имя переменной совершенно различным образом. Правило определения области действия переменной справедливо также в случае рекурсии и в том случае, когда несколько утверждений имеют одну и ту же головную цель. Этот вопрос будет рассмотрен в далее.

Единственным исключением из правила определения области действия переменных является анонимная переменная, например, «_» в цели любит(X,_). Каждая анонимная переменная есть отдельная сущность. Она применяется тогда, когда конкретное значение переменной несущественно для данного утверждения. Таким образом, каждая анонимная переменная четко отличается от всех других анонимных переменных в утверждении.

Переменные, отличные от анонимных, называются *именованными,* а неконкретизированные (переменные, которым не было присвоено значение) называются *свободными.*

7. Сложные термы, или структуры

Структура состоит из атома, называемого главным функтором, и последовательности термов, называемых компонентами структуры. Компоненты разделяются запятыми и заключаются в круглые скобки.

Приведем примеры структурированных термов:

собака(рекс), родитель(X,У).

Число компонент в структуре называется арностью структуры. Так, в данном примере структура собака имеет арность 1 (записывается как собака/1), а структура родитель - арность 2 (родитель/2). Заметим, что атом можно рассматривать как структуру арности 0.

Для некоторых типов структур допустимо использование альтернативных форм синтаксиса. Это синтаксис операторов для структур арности 1 и 2, синтаксис списков для структур в форме списков и синтаксис строк для структур, являющихся списками кодов символов.

8. Синтаксис операторов

Структуры арности 1 и 2 могут быть записаны в операторной форме, если атом, используемый как главный функтор в структуре, объявить оператором (см. гл. 6).

9. Синтаксис списков

В сущности, список есть не что иное, как некоторая структура арности 2. Данная структура становится интересной и чрезвычайно полезной в случае, когда вторая компонента тоже является списком. Вследствие важности таких структур в Прологе имеются специальные средства для записи списков. Возможности обработки списков рассматриваются в разд. 5.1.

10. Синтаксис строк

Строка определяется как список кодов символов. Коды символов имеют особое значение в языках программирования. Они выступают как средство связи компьютера с внешним миром. В большинстве реализации Пролога существует специальный синтаксис для записи строк. Он подобен синтаксису атомов. Строкой является любая последовательность символов, которые могут быть напечатаны (кроме двойных кавычек), заключенная в двойные кавычки. Двойные кавычки в пределах строки записываются дважды «».

В некоторых реализациях Пролога строки рассматриваются как определенный тип объектов подобно атомам или спискам. Для их обработки вводятся специальные встроенные предикаты. В других реализациях строки обрабатываются в точности так же, как списки, при этом используются встроенные предикаты для обработки списков. Поскольку все строки могут быть определены как атомы или как списки целых чисел, и понятие строки является чисто синтаксическим, мы не будем более к нему возвращаться.

11. Утверждения

Программа на Прологе есть совокупность утверждений. Утверждения состоят из целей и хранятся в базе данных Пролога. Таким образом, база данных Пролога может рассматриваться как программа на Прологе. В конце утверждения ставится точка «.». Иногда утверждение называется предложением.

Основная операция Пролога - доказательство целей, входящих в утверждение.

Существуют два типа утверждений:

факт: это одиночная цель, которая, безусловно, истинна;

правило: состоит из одной головной цели и одной или более хвостовых целей, которые истинны при некоторых условиях.

Правило обычно имеет несколько хвостовых целей в форме конъюнкции целей.

Конъюнкцию можно рассматривать как логическую функцию И. Таким образом, правило согласовано, если согласованы все его хвостовые цели.

Примеры фактов:

собака(реке). родитель(голди.рекс).

Примеры правил:

собака (X) :- родитель (X.Y),собака (Y). человек(X) :-мужчина(X).

Разница между правилами и фактами чисто семантическая. Хотя для правил мы используем синтаксис операторов (более подробное рассмотрение операторного и процедурного синтаксисов выходит за рамки нашего курса), нет никакого синтаксического различия между правилом и фактом.

Так, правило

собака (X) :- родитель(X,У),собака(У). может быть задано как

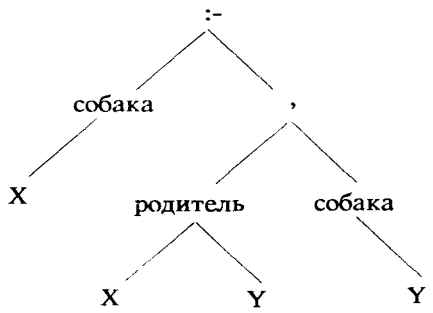
:-собака (X) ',' родитель(X.У) .собака (Y).

Запись верна, поскольку :- является оператором «при условии, что», а ',' - это оператор конъюнкции. Однако удобнее записывать это как

собака (X) :-родитель (X.Y),собака (Y).

и читать следующим образом: « X - собака при условии, что родителем X является Y и Y - собака».

Структуру иногда изображают в виде дерева, число ветвей КОТОРОГО равно арности структуры.



12. Запросы

После записи утверждений в базу данных вычисления могут быть инициированы вводом запроса. Запрос выглядит так же, как и целевое утверждение, образуется и обрабатывается по тем же правилам, но он не входит в базу данных (программу). В Прологе вычислительная часть программы и данные имеют одинаковый синтаксис. Программа обладает как декларативной, так и процедурной семантикой. Мы отложим обсуждение этого вопроса до последующих глав. Запрос обозначается в Прологе утверждением ?-, имеющим арность 1. Обычно запрос записывается в операторной форме: за знаком ?- следует ряд хвостовых целевых утверждений (чаще всего в виде конъюнкции).

Приведем примеры запросов:

?-собака(X). ?- родитель(X.У),собака (Y).

или, иначе,

'?-'(собака(X)) C?-') ','(родитель(X,,У»,собака (Y)).

Последняя запись неудобна тем, что разделитель аргументов в структуре совпадает с символом конъюнкции. Программисту нужно помнить о различных значениях символа ','.

Запрос иногда называют управляющей командой (директивой), так как он требует от Пролог-системы выполнения некоторых действий. Во многих реализациях Пролога для управляющей команды используется альтернативный символ, а символ ?- обозначает приглашение верхнего уровня интерпретатора Пролога. Альтернативным символом является :-. Таким образом,

:-write(собака).

- это управляющая команда, в результате выполнения которой печатается атом собака. Управляющие команды будут рассмотрены ниже при описании ввода программ.

13. Ввод программ

Введение списка утверждений в Пролог-систему осуществляется с помощью встроенного предиката consult. Аргументом предиката consult является атом, который обычно интерпретируется системой как имя файла, содержащего текст программы на Прологе. Файл открывается, и его содержимое записывается в базу данных. Если в файле встречаются управляющие команды, они сразу же выполняются. Возможен случай, когда файл не содержит ничего, кроме управляющих команд для загрузки других файлов. Для ввода утверждений с терминала в большинстве реализации Пролога имеется специальный атом, обычно user. С его помощью утверждения записываются в базу данных, а управляющие команды выполняются немедленно.

Помимо предиката consult, в Прологе существует предикат reconsult. Он работает аналогичным образом. Но перед добавлением утверждений к базе данных из нее автоматически удаляются те утверждения, головные цели которых сопоставимы с целями, содержащимися в файле перезагрузки. Такой механизм позволяет вводить изменения в базу данных. В Прологе имеются и другие методы добавления и удаления утверждений из базы данных. Некоторые реализации языка поддерживают модульную структуру, позволяющую разрабатывать модульные программы.

В заключение раздела дадим формальное определение синтаксиса Пролога, используя форму записи Бэкуса-Наура, иногда называемую бэкусовской нормальной формой (БНФ).

запрос ::- голова утверждения

правило ::– голова утверждения :- хвост утверждения
факт ::- голова утверждения
голова утверждения ::-атом | структура
хвост утверждения ::- атом структура,
термы ::-терм [,термы]
терм ::- число | переменная | атом | структура
структура ::-атом (термы)

Данное определение синтаксиса не включает операторную, списковую и строковую формы записи. Полное определение дано в приложении А. Однако, любая программа на Прологе может быть написана с использованием вышеприведенного синтаксиса. Специальные формы только упрощают понимание программы. Как мы видим, синтаксис Пролога не требует пространного объяснения. Но для написания хороших программ необходимо глубокое понимание языка.

7. ЭКСПЕРТНЫЕ СИСТЕМЫ: ОПРЕДЕЛЕНИЯ И КЛАССИФИКАЦИЯ.

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 лекции.

Под ***экспертной системой (ЭС)*** будем понимать программу, 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.

Обобщенная схема *ЭС* приведена на рис. 6. 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.

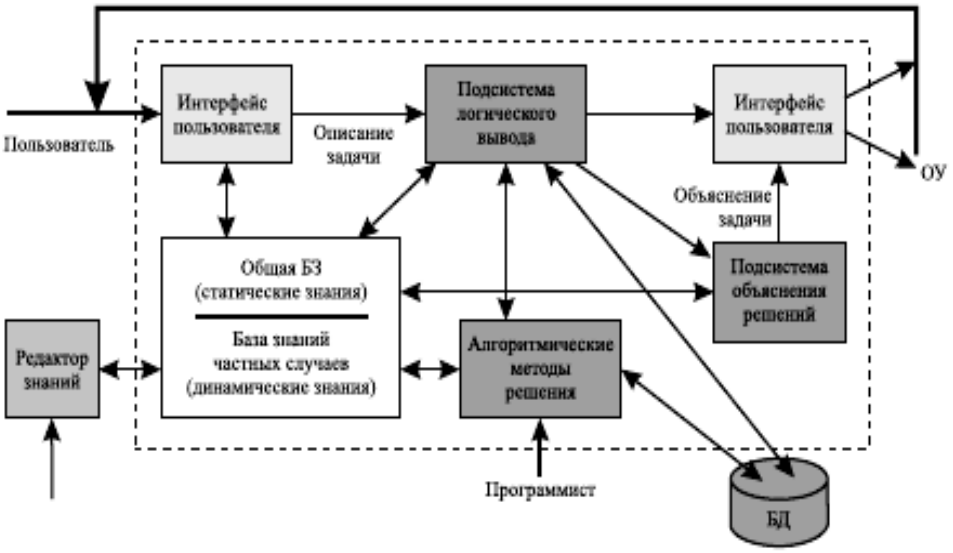


Рис. 6. Структура экспертной системы

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].

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.

Трудности при разработке экспертных систем

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

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

принимают правильные решения на интуитивном уровне и не могут аргументированно объяснить, почему принято то или иное решение. Иногда *эксперты* не могут прийти к взаимопониманию (фраза «встретились два геолога, у них было три мнения» - не шутка, а реальная жизнь). В таких ситуациях поможет выбор *эксперта*, умеющего ясно формулировать свои мысли и легко объяснять другим свои идеи.

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

7.МЕТОДОЛОГИЯ ПОСТРОЕНИЯ ЭКСПЕРТНЫХ СИСТЕМ.

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.

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 Стэндфордском 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 Стэндфордском 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 Стэндфордском 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. 7 показано, что такая ЭСД работает совместно с втоматизированной 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.

Рис. 7. Общая структура экспертной системы диагностирования

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.

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.

8.ЛОГИЧЕСКИЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЯ. ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ.

Для описания внешнего мира и поиска решений в искусственном интеллекте широко используются язык и аппарат исчисления предикатов. Исчисление (или алгебра) предикатов представляет собой развитие исчисления высказываний и включает его полностью как составную часть. Поэтому знакомство с исчислением предикатов начнем с рассмотрения исчисления высказываний.

Исчисление высказываний.

В математической логике *высказыванием* называется такое предложение, которое истинно или ложно. Условимся высказывания обозначать прописными латинскими буквами без индексов и с индексами *A, B, C, ...* и называть *пропозициональными.*

В соответствии с тем, что высказывания могут быть истинными и ложными, пропозициональные буквы могут принимать *истинностные значения — И* (истина) и *Л* (ложь). Иногда, в частности в теории конечных автоматов, вместо истинностных значений *И* и *Л* используют соответственно 1 и 0.

Более сложные высказывания можно получить, используя истинностное функциональные зависимости *P (A, B, ...*), которые определяются истинностными значениями *A, B, ...* Функции *P (A, B, ...*) обычно задаются с помощью таблиц, которые называются *истинностными таблицами,* и являются истинностными функциями. *Истинностной функцией* (или *функцией алгебры логики)* от *n* аргументов называется всякая функция от *n* истинностных аргументов, принимающая истинностные значения *И* или Л (1 или 0). Истинностными аргументами назовем аргументы, принимающие истинностные значения *И* или *Л (*\ или 0).

Л о г и ч е с к и е о п е р а ц и и .

На основе заданных высказываний с помощью логических операций образуются сложные (составные) высказывания. Простейшими составными высказываниями являются следующие:

1. $\overline{A}$ —*отрицание A* читается также «не А». Высказывание $\overline{A}$ истинно , если высказывание *A* ложно ; $\overline{A}$ ложно, если *A* истинно;
2. $A \wedge B$ (другие обозначения: *A&B) —конъюнкция* или *логическое умножение A* и В; читается также «Л и *И».* Это высказывание истинно в том и только в том случае, когда истинны как *A,* так и В; А и В называются *конъюнктивными членами;*
3. $A \vee B$ — *дизъюнкция* или *логическое сложение A* и В; читается также «А или В». Это высказывание истин но в том и только в том случае, если истинно хотя бы одно из высказываний А и В; А и В называются *дизъюнктивными членами;*
4. $A \rightarrow B$—*импликация* или следование В из А; читается также «если *A,* то В». Это высказывание ложно в том и только в том случае, если А истинно и В ложно;
5. $A \leftrightarrow B$—*эквиваленция* или *эквивалентность-,* читается «А тогда и только тогда, когда В». Это высказывание истинно тогда и только тогда, когда А и В имеют одно и то же истинностное значение.

Символы логических операций —, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$называются также *пропозициональными знаками.*

Ниже приведена истинностная таблица, определенная введенные элементарные составные высказывания (табл. 2.1 или табл. 2.2).

Т а б л и ц а 2.1

A	B	$\overline{A}$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$

Л	*Л*	*И*	*Л*	*Л*	*И*	*И*
Л	*И*	*И*	*Л*	*И*	*И*	*Л*
И	*Л*	*Л*	*Л*	*И*	*Л*	*Л*
И	*И*	*Л*	*И*	*И*	*И*	*И*

Таблица 2.2

A	B	$\overline{A}$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Пропозициональная форма.

Всякое сложное высказывание, составленное из некоторых исходно высказываний посредством логических операций, называют *формулой алгебры логики* или *пропозициональны формой.*

Более точно пропозициональная форма (или форму/ алгебры логики) определяется следующим образа (рекурсивное определение):

1. Все пропозициональные буквы—суть пропозициональные формы;
2. Если *P и Q* — пропозициональные формы, то *1* $(P \wedge AQ)$, $(P \vee C)$, $(P \rightarrow C)$ и $(P \leftrightarrow Q)$ -- пропозициональные формы;
3. Никакие другие выражения не являются пропозициональными формами.

Для краткости пропозициональные формы также будем называть *формами* или *формулами*.

Каждая форма задает истинностную функцию. При чем различные формы могут задавать одну и ту ж истинностную функцию. Две различные формы, задающие одну и ту же истинностную функцию, называются *равносильными*. Если *P* и *Q* — равносильные формы то условимся писать: *P=C*. Форма, которая истинна при любых истинностных значениях ее аргументов, называется *тавтологией.*

9.ФРЕЙМЫ ДЛЯ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.

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.

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. 8) в виде *фреймов*.

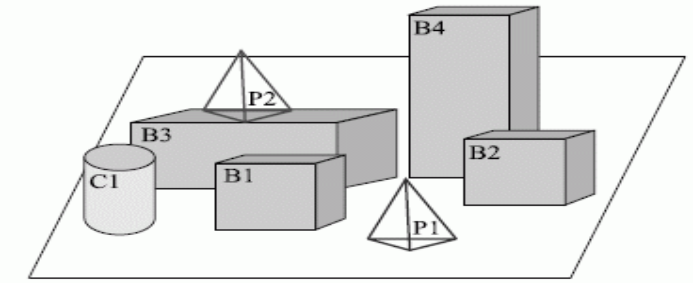


Рис. 8. "Мир блоков"

```
(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.

10. ПРИМЕНЕНИЕ ИСЧИСЛЕНИЯ ПРЕДИКАТОВ ДЛЯ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ В СИИ

Традиционная булева алгебра и исчисление высказываний 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. 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: $\neg$ - не, $\vee$- или, $\wedge$- и, $\supset$- если, а также квантор $\exists$существования и квантор всеобщности $\forall$

$\exists$x(...) - существует такой x, что ...

$\forall$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:

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

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

$\exists x$ (владеет(Джон, x) $\supset$ машина(x) $\wedge$ красный(x))

Рассмотрим вывод, дающий заключение на основе двух предпосылок:

Предпосылка 1: Все люди смертны

$\forall x$ (человек(x) $\supset$ смертен(x))

$\forall x$ (p(x) $\supset$ 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:

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

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*.

11. СИСТЕМЫ ПРОДУКЦИЙ. СЕМАНТИЧЕСКИЕ СЕТИ. ЛИНГВИСТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОТНОШЕНИЯ.

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 х 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)

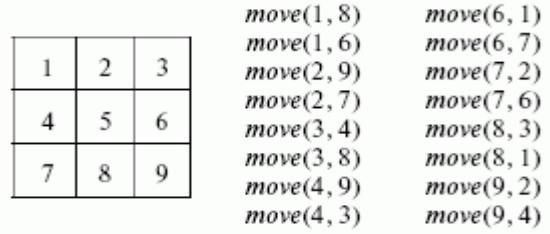


Рис. 9. Шахматная доска 3х3 для задачи хода конем с допустимыми ходами

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.

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. 10.

Рис. 10. 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 на рис. 11.

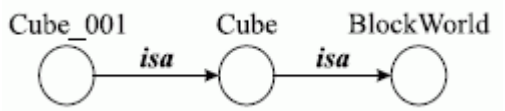


Рис. 11. Процедура сопоставления в СС

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. 12. Произведем сопоставление вопроса с сетью, представленной на рис.13. В результате сопоставления получается ответ - "Пирамида".

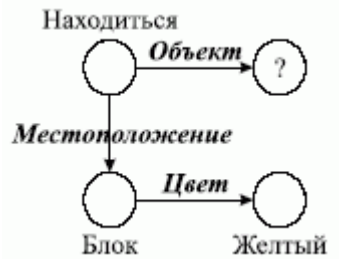


Рис. 12. Вопрос в виде СС

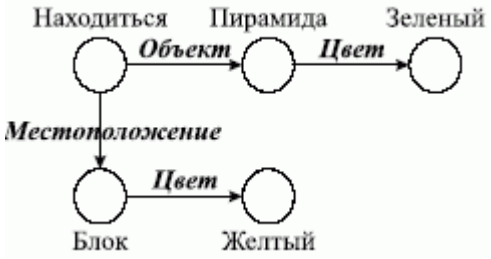


Рис. 13. Процедура сопоставления в СС

Лингвистические и логические отношения.

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. 14).

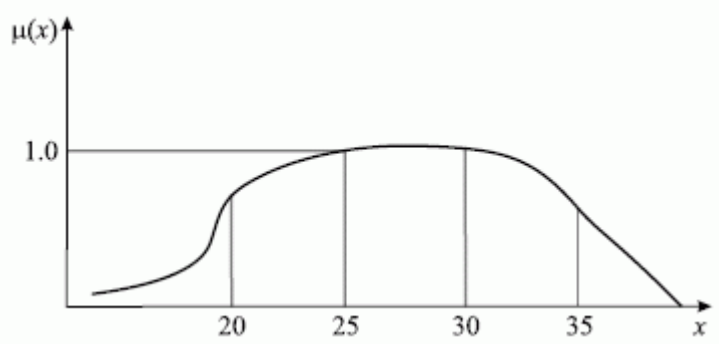


Рис. 14. График функции принадлежности

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.

12. МЕТОДЫ ПОИСКА РЕШЕНИЙ В СИИ

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.

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. 15. 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***).

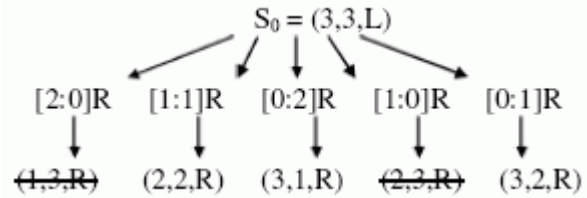


Рис. 15. Переходы из исходного состояния

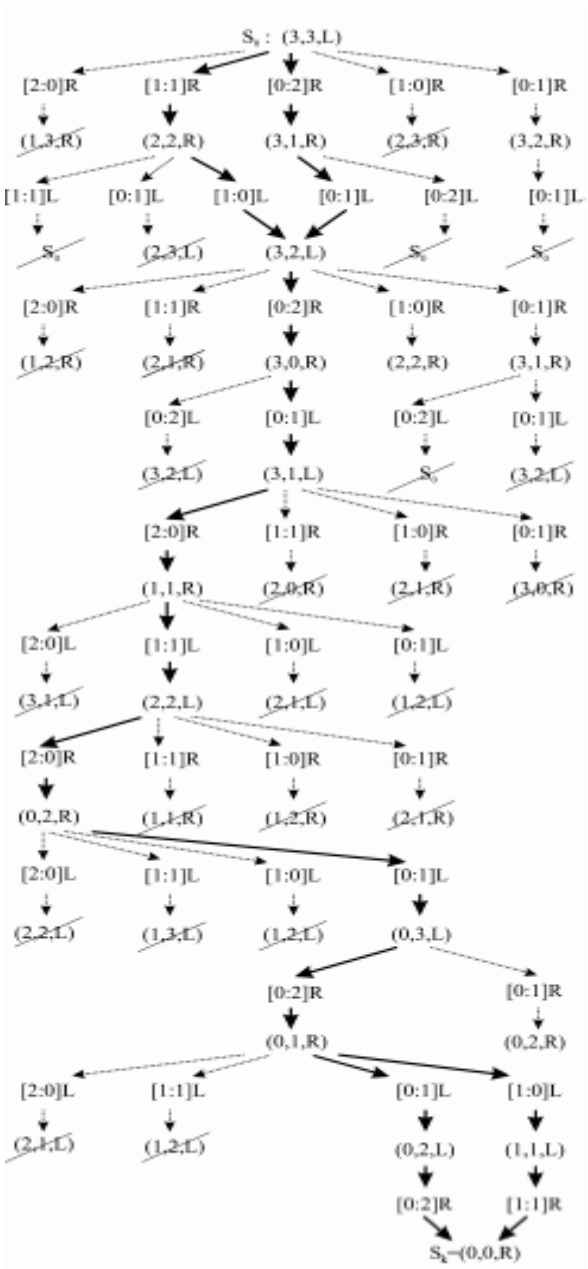


Рис. 16. Метод поиска в пространстве состояний

Теперь мы можем проанализировать полностью алгоритм простейшего *поиска решений в проблемном пространстве*, описанный группами состояний и переходами между состояниями на рис. 16. Решение задачи выделено на рис.16 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*.

13. 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. 17).

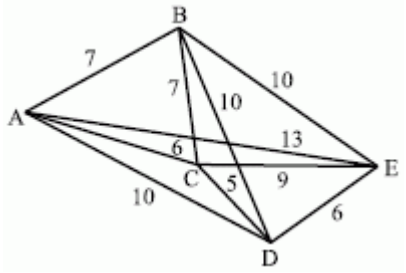


Рис.17.

Задача состоит в том, чтобы, начиная с города А, найти минимальный путь, проходящий через все остальные города только один раз и приводящий обратно в А. Идея метода исключительно проста - из каждого города идем в ближайший, где мы еще не были. Решение задачи показано на рис. 18.

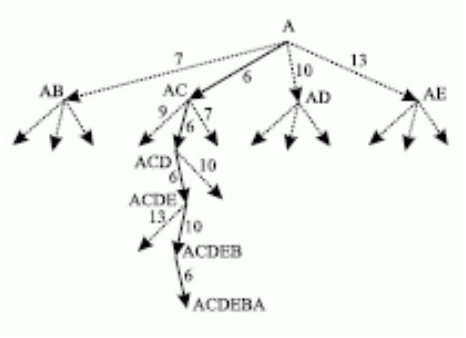


Рис. 18.

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).

2	8	3
1	6	4
7	*	5

1	2	3
8	*	4
7	6	5

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. 19 приведен пример анализа дерева ходов с помощью *метода минимакса* (выбранный путь решения отмечен жирной линией).

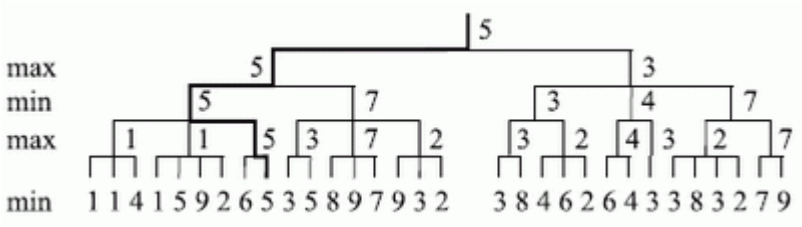


Рис. 19. Дерево ходов

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. 20. 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).

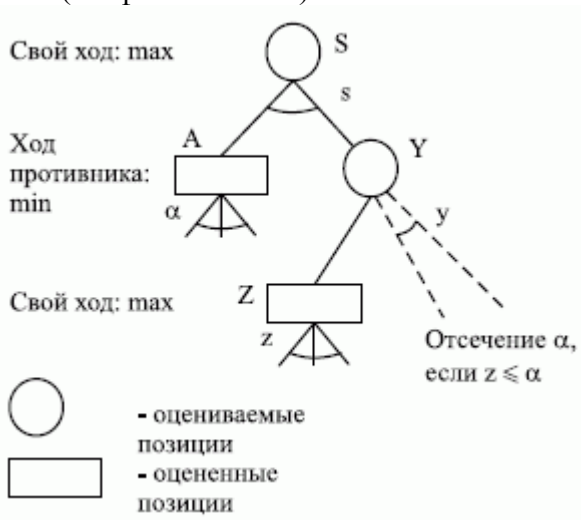


Рис. 20- отсечение

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 +∞ и -∞, 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 α всех ее родительских MAX вершин;
4. можно не проводить поиска на поддереве, растущем из всякой MAX вершины, у которой значение α не меньше значения β всех ее родительских MIN вершин.

На рис. 21 показаны α-β отсечения для конкретного примера. Таким образом, α-β-алгоритм дает тот же результат, что и *метод минимакса*, но выполняется быстрее.

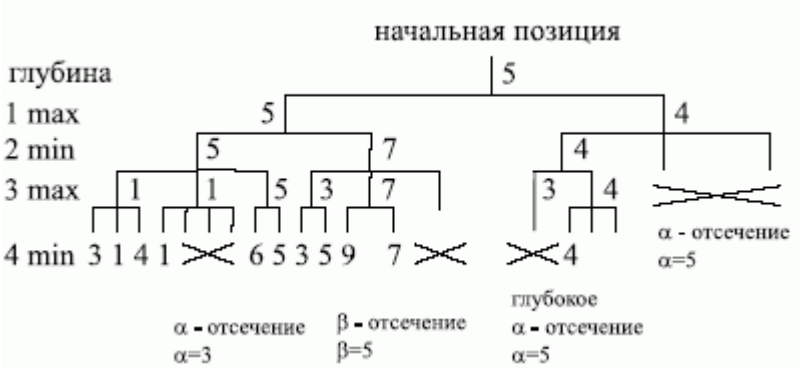


Рис. 21 a-b отсечение для конкретного примера

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.

14. 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 32 Требуется ответить на вопрос: "Существует ли человек, живущий интересной жизнью?" В виде предикатов эти утверждения записаны во втором столбце таблицы. Предполагается, что $\forall X(smart(X)=\neg stupid(X))$ и $\forall Y(wealthy(Y)=\neg poor(Y))$. В третьем столбце таблицы записаны дизъюнкты.

Таблица 32. Интересная жизнь

Утверждения и заключение	**Предикаты**	**Предложения(дизъюнкты)**
1. Все небедные и умные люди счастливы	$\exists X(\neg poor(X) \wedge smart(X) \rightarrow happy(X))$	$poor(X) \wedge \neg smart(X) \& happy(X)$
2. Человек, читающий книги, - неглуп	$\forall Y (read(Y) \rightarrow smart(Y))$	$\neg read(Y) \& smart(Y)$

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.

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

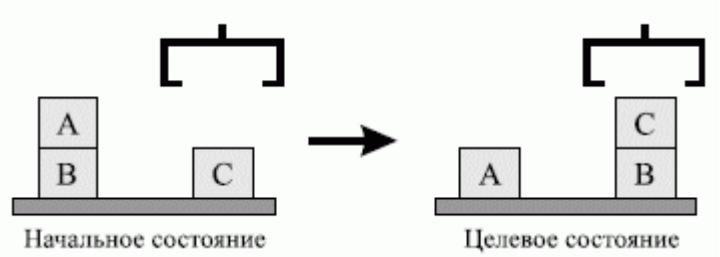


Рис.22. Начальное и целевое состояния задачи из мира кубиков

Предметная область из мира кубиков представлена на рис. 22 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.

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 таблица 3.3 ниже.

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]

15. РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ.

В лекции 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].

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.23 .

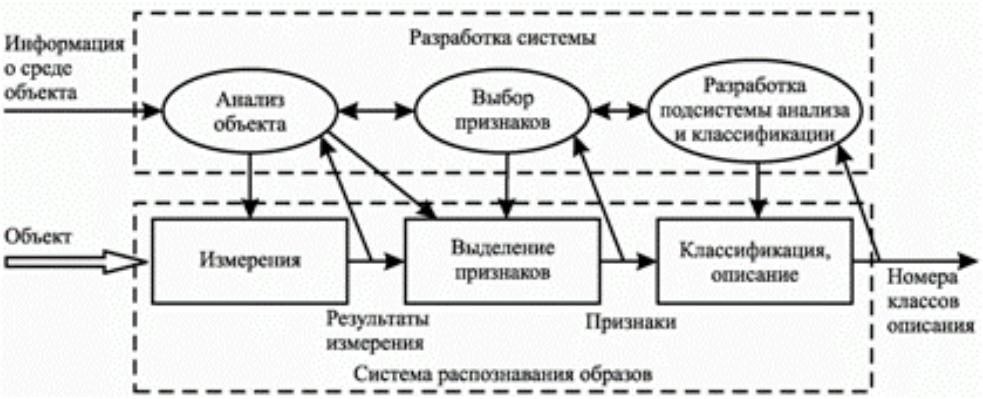


Рис 23. Структура системы распознавания

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.

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 объектаI(ω)=($x_1(\omega)$, ..., $x_N(\omega)$) 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.

Таблица 4.1. Таблица обучения

Объект	***Признаки* и их значения**			**Класс**
	x_1	x_j	x_n	
ω1	α_{11}	α_{1j}	α_{1n}	Ω_1
...				
ω_{r1}	r11	α_{r1j}	α_{r1n}	
...				
ω_{rk}	α_{rk1}	α_{rkj}	α_{rkn}	Ω_m
...				
ω_{rm}	α_{rm1}	α_{rmj}	α_{rmn}	

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. 24 а). Требуется определить *признаки таблицы обучения*, *пороги* и построить оценки близости для классов объектов, показанных на рис. 24 б. 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.

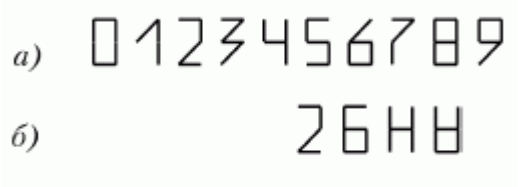


Рис. 24 . Пример задачи по распознаванию

На рис. 25 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.25. Таблица обучения для задачи по распознаванию

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

Объект	**x_1**	**x_2**	**x_3**	**x_4**	**Результат *распознавания***
Объект 1	1	2	1		Цифра 2
Объект 2	3	3	0	1	Цифра 8 или 5
Объект 3	4	1	0		
Объект 4	4	2	0	1	

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.

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.*

16. МЕТОДЫ ОБУЧЕНИЯ РАСПОЗНАВАНИЮ ОБРАЗОВ.

Одним из самых интересных свойств человеческого мозга является способность отвечать на бесконечное множество состояний внешней среды конечным числом реакций. Может быть, именно это свойство позволило человеку достигнуть высшей формы существования живой материи, выражающейся в способности к мышлению, т. е. активному отражению объективного мира в виде образов, понятий, суждений и т. д. Поэтому проблема ОРО возникла при изучении физиологических свойств мозга.

Методы обучения распознаванию образов - перцептроны, нейронные сети, метод потенциальных функций, метод группового учета аргументов, метод предельных упрощений, коллективы решающих правил.

Адаптация — это процесс изменения параметров и структуры системы, а возможно, и управляющих воздействий на основе текущей информации с целью достижения определенного состояния системы при начальной неопределенности и изменяющихся условиях работы.
Обучение — это процесс, в результате которого система постепенно приобретает способность отвечать нужными реакциями на определенные совокупности внешних воздействий, а адаптация — это подстройка параметров и структуры системы с целью достижения требуемого качества управления в условиях непрерывных изменений внешних условий.

Перцептроны

Пока о проблеме обучения распознаванию образов удавалось говорить в общих чертах, не выделяя конкретные методы или алгоритмы, не возникало и трудностей, появляющихся всяких раз, когда приходится в огромном множестве конкретных примеров, характеризующиеся общими подходами к решению проблемы ОРО. Коварство самой проблемы состоит в том, что на первый взгляд все методы и алгоритмы кажутся совершенно различными и, что самое неприятное, часто никакой из них не годится для решения той задачи, которую крайне необходимо срочно решить. И тогда появляется желание выдумать новый алгоритм, который, может быть, достигнет цели. Очевидно, именно это привело к возникновению огромного множества алгоритмов, в котором не так-то легко разобраться.
Одним из методов решения задач обучения распознаванию образов основан на моделировании гипотетического механизма человеческого мозга. Структура модели заранее постулируется. При таком подходе уровень биологических знаний или гипотез о биологических механизмах является исходной предпосылкой, на которой базируются модели этих механизмов. Примером такого направления в теории и практике проблемы ОРО является класс устройств, называемых перцептронами. Нужно отметить, что перцептроны на заре своего возникновения рассматривались только как эвристические модели механизма мозга. Впоследствии они стали основополагающей схемой в построении кусочно-линейных моделей, обучающихся распознаванию образов.

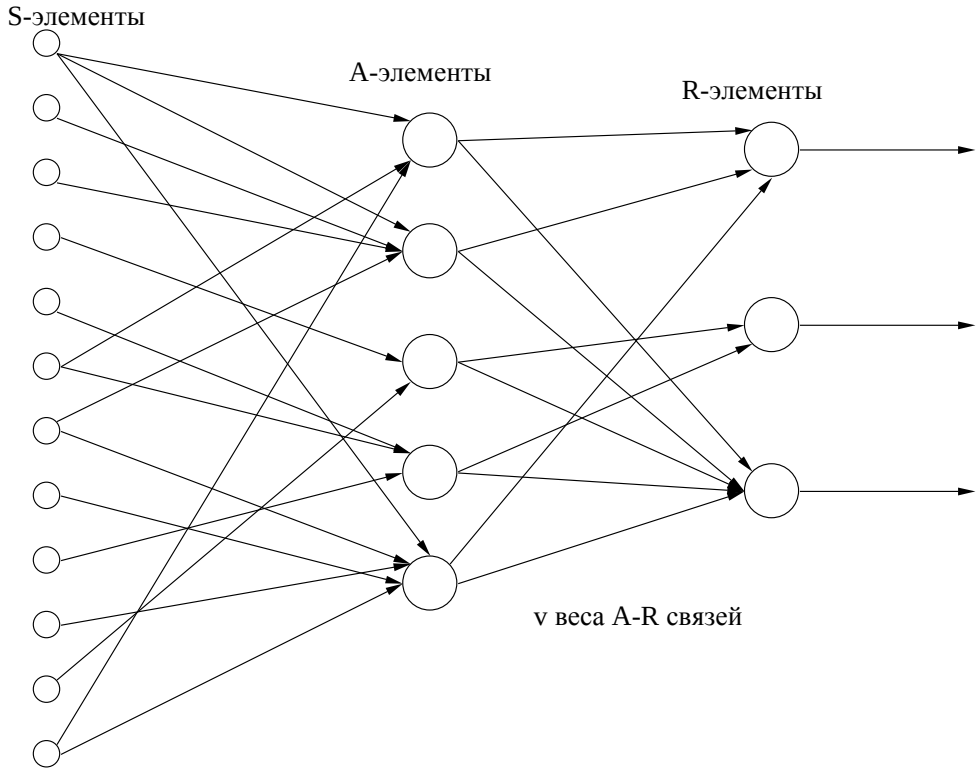


Рис.26 Перцептрон

В наиболее простом виде перцептрон (**Ошибка! Источник ссылки не найден.**26) состоит из совокупности чувствительных (сенсорных) элементов (S-элементов), на которые поступают входные сигналы. S-элементы случайным образом связаны с совокупностью ассоциативных элементов (A-элементов), выход которых отличается от нуля только тогда, когда возбуждено достаточно большое число S-элементов, воздействующих на один A-элемент. A-элементы соединены с реагирующими элементами (R-элементами) связями, коэффициенты усиления (v) которых переменны и изменяются в процессе обучения. Взвешенные комбинации выходов R-элементов составляют реакцию системы, которая указывает на принадлежность распознаваемого объекта определенному образу. Если распознаются только два образа, то в перцептроне устанавливается только один R-элемент, который обладает двумя реакциями — положительной и отрицательной. Если образов больше двух, то для каждого образа устанавливают свой R-элемент, а выход каждого такого элемента представляет линейную комбинацию выходов A-элементов:

$$R_j = \Theta_j + \sum_{i=1}^{n} v_{ij} x_i \,, \qquad \textbf{(1)}$$

где R_j — реакция j-го R-элемента; x_i — реакция i-го A-элемента; v_{ij} — вес связи от i-го A-элемента к j-му R элементу; Θ_j — порог j-го R-элемента.
Аналогично записывается уравнение i-го A-элемента:

$$x_i = \Theta_i + \sum_{k=1}^{S} y_k \,, \qquad \textbf{(2)}$$

Здесь сигнал y_k может быть непрерывным, но чаще всего он принимает только два значения: 0 или 1. Сигналы от S-элементов подаются на входы A-элементов с постоянными весами равными единице, но каждый A-элемент связан только с группой случайно выбранных S-

элементов. Предположим, что требуется обучить перцептрон различать два образа V_1 и V_2. Будем считать, что в персептроне существует два R-элемента, один из которых предназначен образу V_1, а другой — образу V_2. Персептрон будет обучен правильно, если выход R_1 превышает R_2, когда распознаваемый объект принадлежит образу V_1, и наоборот. Разделение объектов на два образа можно провести и с помощью только одного R-элемента. Тогда объекту образа V_1 должна соответствовать положительная реакция R-элемента, а объектам образа V_2 — отрицательная.

Персептрон обучается путем предъявления обучающей последовательности изображений объектов, принадлежащих образам V_1 и V_2. В процессе обучения изменяются веса v_i A-элементов. В частности, если применяется система подкрепления с коррекцией ошибок, прежде всего учитывается правильность решения, принимаемого персептроном. Если решение правильно, то веса связей всех сработавших A-элементов, ведущих к R-элементу, выдавшему правильное решение, увеличиваются, а веса несработавших A-элементов остаются неизменными. Можно оставлять неизменными веса сработавших A-элементов, но уменьшать веса несработавших. В некоторых случаях веса сработавших связей увеличивают, а несработавших — уменьшают. После процесса обучения перцептрон сам, без учителя, начинает классифицировать новые объекты.

Если персептрон действует по описанной схеме и в нем допускаются лишь связи, идущие от бинарных S-элементов к A-элементам и от A-элементов к единственному R-элементу, то такой персептрон принято называть элементарным α-персептроном. Обычно классификация C(W) задается учителем. Персептрон должен выработать в процессе обучения классификацию, задуманную учителем.

О персептронах было сформулировано и доказано несколько основополагающих теорем, две из которых, определяющие основные свойства персептрона, приведены ниже.

Теорема 1. Класс элементарных α-персептронов, для которых существует решение для любой задуманной классификации, не является пустым.

Эта теорема утверждает, что для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) A-элементов, в котором будет осуществлено задуманное разделение обучающей последовательности при помощи линейного решающего правила ($R_j = \Theta_j + \sum_{i=1}^{n} v_{ij} x_i$).

Теорема 2. Если для некоторой классификации C(W) решение существует, то в процессе обучения α-персептрона с коррекцией ошибок, начинающегося с произвольного исходного состояния, это решение будет достигнуто в течение конечного промежутка времени.

Смысл этой теоремы состоит в том, что если относительно задуманной классификации можно найти набор A-элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто в конечный промежуток времени.

Обычно обсуждают свойства бесконечного персептрона, т. е. персептрона с бесконечным числом A-элементов со всевозможными связями с S-элементами (полный набор A-элементов). В таких персептронах решение всегда существует, а раз оно существует, то оно и достижимо в α-персептронах с коррекцией ошибок.

Очень интересную область исследований представляют собой многослойные персептроны и персептроны с перекрестными связями, но теория этих систем практически еще не разработана.

17. НЕЙРОННЫЕ СЕТИ. ИСТОРИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ НЕЙРОННЫХ СЕТЕЙ

Возвратимся немного назад, и рассмотрим историю исследований нейронных сетей.

В истории исследований в области нейронных сетей, как и в истории любой другой науки, были свои успехи и неудачи. Кроме того, здесь постоянно сказывается психологический фактор, проявляющийся в неспособности человека описать словами то, как он думает.

Способность нейронной сети к обучению впервые исследована Дж. Маккалоком и У. Питтом. В 1943 году вышла их работа “Логическое исчисление идей, относящихся к нервной деятельности", в которой была построена модель нейрона, и сформулированы принципы построения искусственных нейронных сетей.

Крупный толчок развитию нейрокибернетики дал американский нейрофизиолог Френк Розенблатт, предложивший в 1962 году свою модель нейронной сети — персептрон. Воспринятый первоначально с большим энтузиазмом, он вскоре подвергся интенсивным нападкам со стороны крупных научных авторитетов. И хотя подробный анализ их аргументов показывает, что они оспаривали не совсем тот персептрон, который предлагал Розенблатт, крупные исследования по нейронным сетям были свернуты почти на 10 лет.

Несмотря на это в 70-е годы было предложено много интересных разработок, таких, например, как когнитрон, способный хорошо распознавать достаточно сложные образы независимо от поворота и изменения масштаба изображения.

В 1982 году американский биофизик Дж. Хопфилд предложил оригинальную модель нейронной сети, названную его именем. В последующие несколько лет было найдено множество эффективных алгоритмов: сеть встречного потока, двунаправленная ассоциативная память и др.

В киевском институте кибернетики с 70-х годов ведутся работы над стохастическими нейронными сетями.

Модель нейронной сети с обратным распространением ошибки (back propagation)

В 1986 году Дж. Хинтон и его коллеги опубликовали статью с описанием модели нейронной сети и алгоритмом ее обучения, что дало новый толчок исследованиям в области искусственных нейронных сетей.

Нейронная сеть состоит из множества одинаковых элементов — нейронов, поэтому начнем с них рассмотрение работы искусственной нейронной сети.

Биологический нейрон моделируется как устройство, имеющее несколько входов (дендриты), и один выход (аксон). Каждому входу ставится в соответствие некоторый весовой коэффициент (w), характеризующий пропускную способность канала и оценивающий степень влияния сигнала с этого входа на сигнал на выходе. В зависимости от конкретной реализации, обрабатываемые нейроном сигналы могут быть аналоговыми или цифровыми (1 или 0). В теле нейрона происходит взвешенное суммирование входных возбуждений, и далее это значение является аргументом активационной функции нейрона, один из возможных вариантов которой представлен на Рис. 27.

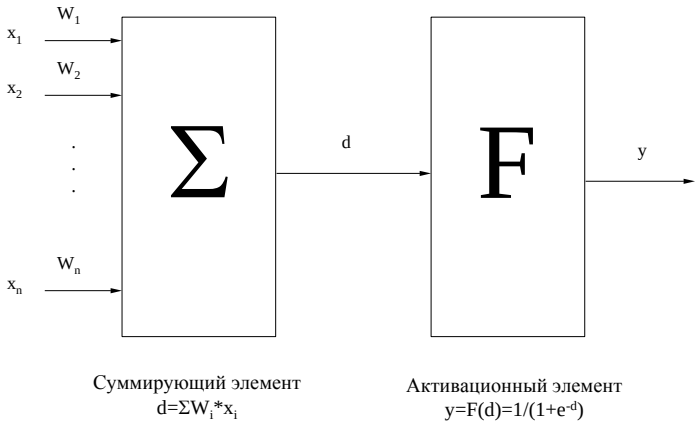


Рис. 27. Искусственный нейрон

Будучи соединенными определенным образом, нейроны образуют нейронную сеть. Работа сети разделяется на обучение и адаптацию. Под обучением понимается процесс адаптации сети к предъявляемым эталонным образцам путем модификации (в соответствии с тем или иным алгоритмом) весовых коэффициентов связей между нейронами. Заметим, что этот процесс является результатом алгоритма функционирования сети, а не предварительно заложенных в нее знаний человека, как это часто бывает в системах искусственного интеллекта.

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносв язными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синоптических связей идет в направлении, минимизирующем ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного перцептрона. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более словный перцептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод "тыка", несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, более приемлемый вариант – распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения. Именно он будет рассмотрен в дальнейшем.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \qquad (1)$$

где $y_{j,p}^{(N)}$ – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа; d_{jp} – идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (2)$$

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя n-1 с j-ым нейроном слоя n, η – коэффициент скорости обучения, $0<\eta<1$.

Как показано в [2],

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}} \qquad (3)$$

Здесь под y_j, как и раньше, подразумевается выход нейрона j, а под s_j – взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель dy_j/ds_j является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых НС. В них применяются такие гладкие функции, как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса

$$\frac{dy}{ds} = 1 - s^2 \qquad (4)$$

Третий множитель $\partial s_j/\partial w_{ij}$, очевидно, равен выходу нейрона предыдущего слоя $y_i^{(n-1)}$.

Что касается первого множителя в (3), он легко раскладывается следующим образом[2]:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \qquad (5)$$

Здесь суммирование по k выполняется среди нейронов слоя n+1.

Введя новую переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \qquad (6)$$

мы получим рекурсивную формулу для расчетов величин $\delta_j^{(n)}$ слоя n из величин $\delta_k^{(n+1)}$ более старшего слоя n+1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \qquad (7)$$

Для выходного же слоя

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \qquad (8)$$

Теперь мы можем записать (2) в раскрытом виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \qquad (9)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (9) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \qquad (10)$$

где μ – коэффициент инерционности, t – номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Напомним, что

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} \cdot w_{ij}^{(n)} \qquad (11)$$

где M – число нейронов в слое n-1 с учетом нейрона с постоянным выходным состоянием +1, задающего смещение; $y_i^{(n-1)}=x_{ij}^{(n)}$ – i-ый вход нейрона j слоя n.

$y_j^{(n)} = f(s_j^{(n)})$, где $f()$ – сигмоид (12)

$y_q^{(0)}=I_q$, (13)

где I_q – q-ая компонента вектора входного образа.

2. Рассчитать $\delta^{(N)}$ для выходного слоя по формуле (8).

Рассчитать по формуле (9) или (10) изменения весов $\Delta w^{(N)}$ слоя N.

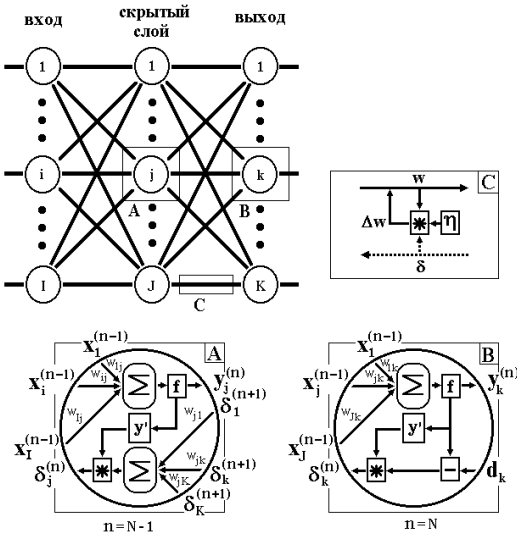
3. Рассчитать по формулам (7) и (9) (или (7) и (10)) соответственно $\delta^{(n)}$ и $\Delta w^{(n)}$ для всех остальных слоев, n=N-1,...1.

4. Скорректировать все веса в НС

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \qquad (14)$$

5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

Рис. 28. Диаграмма сигналов в сети при обучении по алгоритму обратного распространения



Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других. Алгоритм иллюстрируется Рис. 28.

Из выражения (9) следует, что когда выходное значение $y_i^{(n-1)}$ стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться[3], поэтому область возможных значений выходов нейронов [0,1] желательно сдвинуть в пределы [-0.5,+0.5], что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1+e^{-\alpha \cdot x}} \qquad (15)$$

Теперь коснемся вопроса емкости НС, то есть числа образов, предъявляемых на ее входы, которые она способна научиться распознавать. Для сетей с числом слоев больше двух, он остается открытым. Как показано в [4], для НС с двумя слоями, то есть выходным и одним скрытым слоем, детерминистская емкость сети C_d оценивается так:

$N_w/N_y<C_d<N_w/N_y \cdot \log(N_w/N_y)$ (16)

где N_w – число подстраиваемых весов, N_y – число нейронов в выходном слое.

Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов N_x и нейронов в скрытом слое N_h должно удовлетворять неравенству $N_x+N_h>N_y$. Во-вторых, $N_w/N_y>1000$. Однако вышеприведенная оценка выполнялась для сетей с активационными функциями нейронов в виде порога, а емкость сетей с гладкими активационными функциями, например – (15), обычно больше. Кроме того, фигурирующее в названии емкости прилагательное "детерминистский" означает, что полученная оценка емкости подходит абсолютно для всех возможных входных образов, которые могут быть представлены N_x входами. В действительности распределение входных образов, как правило, обладает некоторой регулярностью, что позволяет НС проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение образов, в общем случае, заранее не

известно, мы можем говорить о такой емкости только предположительно, но обычно она раза в два превышает емкость детерминистскую.

В продолжение разговора о емкости НС логично затронуть вопрос о требуемой мощности выходного слоя сети, выполняющего окончательную классификацию образов. Дело в том, что для разделения множества входных образов, например, по двум классам достаточно всего одного выхода. При этом каждый логический уровень – "1" и "0" – будет обозначать отдельный класс. На двух выходах можно закодировать уже 4 класса и так далее. Однако результаты работы сети, организованной таким образом, можно сказать – "под завязку", – не очень надежны. Для повышения достоверности классификации желательно ввести избыточность путем выделения каждому классу одного нейрона в выходном слое или, что еще лучше, нескольких, каждый из которых обучается определять принадлежность образа к классу со своей степенью достоверности, например: высокой, средней и низкой. Такие НС позволяют проводить классификацию входных образов, объединенных в нечеткие (размытые или пересекающиеся) множества. Это свойство приближает подобные НС к условиям реальной жизни.

Рассматриваемая НС имеет несколько "узких мест". Во-первых, в процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов сместят рабочую точку на пигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут в соответствие с (7) и (8) к остановке обучения, что парализует НС. Во-вторых, применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве η обычно выбирается число меньше 1, но не очень маленькое, например, 0.1, и оно, вообще говоря, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов застабилизируются, η кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

Существует и иной метод исключения локальных минимумов, а заодно и паралича НС, заключающийся в применении стохастических НС, но о них лучше поговорить отдельно.

18. РОБОТОТЕХНИЧЕСКИЕ СИСТЕМЫ С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА.

В этой лекции приводится структура обобщенной интеллектуальной робототехнической системы. Основной упор делается не на игровые, а на интеллектуальные технологические системы. Это делается не с целью принижения интеллекта игровых задач, а наоборот, чтобы показать, насколько важно наличие искусственного интеллекта в технологических системах.

Область робототехники

Робототехника, очувствление, искусственный интеллект, робот-станок, механизм параллельной структуры, электродвигатели безредукторные , высоко моментные , обобщенные координаты.

Область *робототехники* охватывает достаточно широкий класс машин, начиная от простейших игрушек до полностью автоматизированных производств (Автоматически управляемая электростанция, беспилотные космические корабли, автоматические подводные аппараты, ЭВМ, играющая в шахматы — все эти системы можно считать роботами). Поэтому, что называть роботом, к сожалению, не определено окончательно. В представленных лекциях уделим основное внимание промышленным роботам, в которых присутствуют элементы интеллектуальной деятельности.

Создание "разумных" роботов связано, как правило, с приданием им человеческих качеств. Это способность распознавать образы, участвовать в игровых операциях, ставить задачи и принимать решения. Поэтому в дальнейших лекциях остановимся более подробно на детальном рассмотрении подсистем низшего уровня, выполняющих технологические операции обработки деталей, и связь данных подсистем с подсистемами высшего уровня.

Применение вычислительной техники в системах управления и программного обеспечения позволяет реализовать интеллектуальные способности человека и заменить его в сфере оценки ситуации и принятия решений. Совокупность интеллектуальных и механических способностей робототехнической системы позволяет заменить человека в сфере его производственной деятельности.

Искусственный интеллект промышленных ИРС, рассмотренных в лекции 1, заключается в возможности распознавать детали и их поверхности с точки зрения качества и соответствия заданным геометрическим размерам по чертежу, управлять технологическим процессом и принимать решения по его изменению. В свою очередь принятие решения включает формирование промежуточных целей для выполнения поставленной задачи.

Но все вышесказанное не означает, что роль человека будет состоять только том, чтобы наслаждаться работой робототехнических систем, пребывая в полной бездеятельности. Напротив его ответственность возрастает и потребуется колоссальная нагрузка на человека, чтобы управлять сложными системами, создавать новые механизмы и обезопасить себя от любых техногенных катастроф.

Современная быстродействующая вычислительная техника позволила качественно изменить структуру технологического оборудования. **Во-первых**, благодаря высокому быстродействию вычислений появилась возможность осуществлять управление механизмами, в которых перемещения не совпадают с координатами обрабатываемой детали. Например, высоко скоростные прямолинейные перемещения можно выполнять с помощью вращательных пар. **Во-вторых**, быстродействующие средства контроля дали возможность построить системы оперативной настройки режимов обработки, получая информацию об обрабатываемой поверхности.

В *робототехнике* системы *очувствления* и *искусственного интеллекта* нашли достаточно широкое применение [99]. Следует выделить следующие направления развития интеллектуальных роботов:

1. Промышленные роботы, работающие в производственной сфере и заменяющие человека при выполнении технологических операций. Интеллект указанных роботов заключается в их способности автоматически распознавать качество обработанной поверхности,

контролировать режимы обработки и корректировать их в зависимости от поставленной цели, например, минимизировать погрешности, уменьшать энергозатраты, выбирать технологию обработки в зависимости от типа детали и требований к ее выходным характеристикам. В настоящее время это, пожалуй, основной класс роботов, которому должно быть уделено особое внимание, так как замена человека в сфере производства качественно изменит его жизнедеятельность

2. Безусловно к сугубо интеллектуальным роботам следует отнести робото-тележки, перемещающиеся по космическим планетам в условиях непредсказуемой обстановки и выполняющие операции сбора информации о местности, на основе которой они определяют направление своего движения.
3. Игровые роботы, предназначенные для тренировки спортсменов. Роботы, играющие в гольф, теннис, шахматы, соревнующиеся друг с другом, на первый взгляд указанные роботы не предназначены для замены человека на производстве. Однако, как и в человеческой деятельности, при выполнении игровых задач отрабатывается структура искусственных машин, их силовые возможности, быстродействие и интеллектуальные способности.
4. Специальные роботы, способные работать в военной обстановке, а также в условиях особо опасных для жизнедеятельности человека.

ИРС для выполнения производственных задач, так называемые *роботы-станки*, являются устройствами, которые полностью автоматизируют производство по выпуску определенного вида продукции. Данное оборудование оснащается системами контроля технологических и выходных параметров обрабатываемого изделия.

В станочном оборудовании предъявляются достаточно высокое требования к точности, надежности и ответственности выполняемой операции. При выполнении операций обработки и сборки сложных деталей невозможно требовать вероятностного результата. Как правило, такие операции строго детерминированы. Поэтому вероятностные поисковые методы возможны только на стадии обработки результатов. Принятие окончательного решения должно обеспечивать детерминированный результат, обеспечивающий поставленную цель.

Особенно высокие требования предъявляются при обработке поверхностей сложной формы. В этом случае необходимо более точное выполнение режимов обработки, контроль износа инструмента в процессе обработки и обеспечение одновременно нескольких параметров детали. В частности, для каждой точки поверхности нужно одновременно обеспечивать до шести геометрических параметров, не считая качества поверхности. Для сложных поверхностей, кроме требований к самим координатам, накладываются условия и на их производные.

Для соблюдения высоких требований к точности изготовления деталей необходимо осуществлять постоянный контроль геометрических параметров станка, размеров звеньев, температурных изменений и других параметров. Применение механизмов параллельной структуры также качественно меняет подход к проектированию станочного робототехнического оборудования.

Понятие *робот-станок* было введено в 1992 году при описании станочного оборудования, построенного на механизмах параллельной структуры и позволяющего посредством одного и того же механизма выполнять транспортные операции и операции обработки. Данные механизмы позволяют расширить функциональные возможности станочного оборудования и при наличии системы управления, оснащенной элементами *искусственного интеллекта*, делает данное оборудование близким к интеллектуальным роботам.

Совмещение функций особенно актуально для сложных высокоточных операций, когда требуется обработка детали от одной базы. В данном случае получаем универсальное оборудование, позволяющее выполнять несколько различных технологических операций для широкой номенклатуры изделий.

Главной отличительной особенностью *робота-станка* от обрабатывающего центра является универсальность, точнее, более богатые кинематические возможности перемещения механизма. Безусловно, из набора *роботов-станков* можно построить распределенный обрабатывающий центр. Механизмы параллельной структуры расширили возможности исполнительных механизмов станков, сделали их более облегченными и универсальными. Наличие параллельных кинематических цепей позволяет управлять одним выходным звеном по нескольким параллельным каналам, обеспечивая одновременное управление по положению, скорости, более высоким производным, а также по силе. В работе приведены, хотя и не в полном объеме, механизмы параллельной структуры, которые с успехом можно применить в станочном оборудовании.

Последующие лекции ставят своей целью показать место интеллектуальных систем в сфере промышленной *робототехники*. При этом роботы представляются в виде технологических систем, непосредственно выполняющих операцию обработки. Мы их называем *роботами-станками*, так как их кинематическая схема позволяет выполнять транспортные операции и непосредственно обработку. Применение механизмов параллельной структуры уже на низшем уровне позволяет расширить интеллектуальные возможности технологических машин.

Структура и состав интеллектуальной робототехнической системы.

Интеллектуальная робототехническая система включает объект управления совместно со средой, в которой она работает. Объект управления представляет непосредственно механизмы перемещения инструмента и изделия. В состав манипуляторов входят исполнительные двигатели, которые осуществляют их перемещение по заданным законам $R_Д$ и $R_И$. Информация о положении выходных звеньев манипуляторов определяется датчиками, расположенными в шарнирах звеньев манипуляторов, которые получают информации о выходных координатах механизмов перемещения, их скоростях, ускорениях и силах. Основная функция системы управления манипуляторами состоит в формировании законов перемещения исполнительными механизмами манипуляторов в реальном времени $U_И(t)$ и $U_Д(t)$. Данные системы обычно работают в следящем режиме, обеспечивающем выполнение каждой степенью подвижности манипуляторов заданной траектории перемещения с требуемыми точностью, скоростью и усилием. Выходными координатами манипуляторов являются $R_Д$ и $R_И$. В результате взаимодействия инструмента с деталью создается усилие P(t), которое воздействует на исполнительные органы манипуляторов. Применительно к рассматриваемой системе в качестве объекта управления и внешней среды следует рассматривать манипуляторы перемещения изделия, инструмента и непосредственно сам технологический процесс.

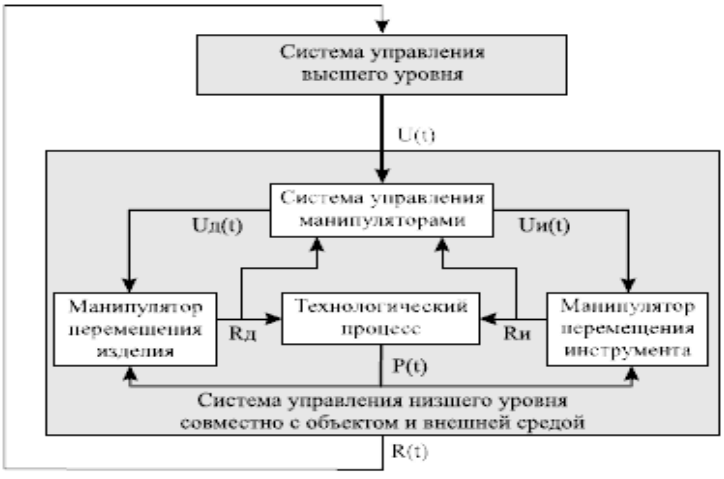


Рис. 29. Структура робототехнической системы

На рисунке 29 не раскрывается состав подсистемы управления высшего уровня. Ее структура и выполняемые функции подробно описаны в лекции 1. Общим информационным управляющим каналом на систему управления низшего уровня является канал передачи

управляющих сигналов U(t) и обратной связью от системы низшего уровня - сигнал R(t). Управляющее воздействие U(t) представляет выбранную программу действия из некоторого множества U $\in$U(t) и соответствующую заданной детали, либо обработке заданной поверхности детали. Какую из программ следует выбрать, решается системой высшего уровня как на основе информации от системы распознавания поверхности, так и на основе указаний оператора, управляющего робототехнической системой. Выбранная программа U(t) задается непрерывно в реальном масштабе времени.

Обратная связь R(t) может нести полную информацию о работе системы управления низшего уровня в виде логических сигналов о ее состоянии, непрерывную информацию о геометрических размерах, качестве обработки поверхности детали и информацию о состоянии внешней среды, например, о температуре окружающей среды или двигателей, о состоянии сопутствующих обработке других устройств.

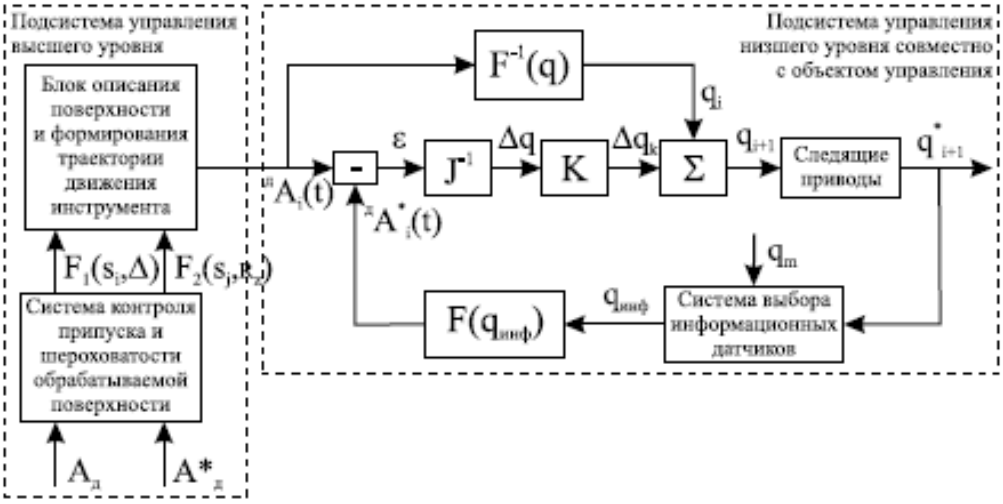


Рис. 30. Система управления робота-станка.

Представленная на рисунке 29 система является обобщенной для технологических машин широкого назначения. Более детальное представление данной системы рассмотрим на примере системы управления робота-станка (рис. 30). Отличительной особенностью рассматриваемой следящей системы управления от существующих станочных систем является наличие главной обратной связи по результату обработки поверхности (вычисление $^{Д}A_i^*(t)$). Вычисление $^{Д}A_i^*(t)$ осуществляется в системе координат детали решением прямой задачи о положении $F(q_{инф.})$ по информации датчиков, располагаемых в сочленениях звеньев механизма. Погрешность ε вычисляется сравнением программного значения управляющего воздействия $^{Д}A_i(t)$ и вычисленного реального его значения $^{Д}A_i^*(t)$. Обратный Якобиан J^{-1} и устройство К выполняют функции преобразования и решения линейной задачи вычисления приращений обобщенных координат q_i. Суммируя приращения на каждом шаге вычисления с предыдущим значением, формируется управляющее воздействие на исполнительные приводы q_i.

В качестве электродвигателей приводов манипуляторов применяются безредукторные и высокомоментные электродвигатели. Это требует применения методики синтеза приводов с учетом переменности моментов инерции, а для многостепенной механической системы требуется также учитывать взаимовлияние приводов по степеням подвижности.

Подсистема управления высшего уровня выполняет следующие функции. Получая информацию от оптической системы о состоянии обрабатываемой поверхности и ее геометрических размерах, данная подсистема выбирает требуемую программу обработки из некоторого детерминированного множества программ либо при ее отсутствии на основе анализа принимает наиболее близкую по критерию точности воспроизведения требуемой поверхности.

Оптические средства контроля геометрических размеров припуска и качества обработки (шероховатости) поверхности детали позволяют оптимизировать режимы резания. В работе

приведено описание оптической системы, построенной с применением специальной решетки и источника монохроматического света. В настоящем курсе лекций дается описание данной системы, рассматриваются вопросы построения системы распознавания зон с заданным качеством обработки и формирования на этой основе новой программы обработки поверхности.

Формирование программной траектории перемещения инструмента относительно обрабатываемой поверхности $^{Д}A_i(t)$, производится на основе информации, полученной от оптической системы контроля поверхности и экспертной оценки при выборе режимов обработки. (В лекции 7 был рассмотрен пример выбора режимов и программы обработки в среде CLIPS). Информация о геометрических размерах полученной после обработки поверхности контролируется оптической системой контроля. Эта система формирует также данные о качестве обрабатываемой поверхности. В зависимости от этой информации выбирается ограниченная область обработки поверхности.Математическая модель объекта управления совместно с окружающей средой, формируемая в системе высшего уровня на основе информации, получаемой от датчиков, включает: чертеж детали с реальными геометрическими размерами, чертеж требуемой идеальной детали и набор параметров, определяющих режимы обработки. Указанная модель позволяет, проигрывая различные ситуации, представляющие набор процедур для выполнения обработки, выбирать цель и формировать программу обработки U(t).

19. МОДУЛЬ "ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ" МОБИЛЬНОЙ РОБОТОТЕХНИЧЕСКОЙ СИСТЕМЫ

Интеллект робототехнической системы.

Основной отличительной чертой мобильной робототехнической системы (МРС) с интеллектуальным (интеллектуальным) управлением является наличие в составе ее управляющего устройства модуля "искусственный интеллект" Данный модуль представляет собой информационное и математическое обеспечение интеллектной части управляющей системы МРС и физически реализуется в виде машинных носителей информации, которые содержат знания и данные(текущие и априорные), необходимые для управления функционированием робототехнической системы в изменяющейся и заранее неопределенной среде.

Пара модулей "искусственный интеллект - вычислительное устройство "(ИИ- ВУ) образует как бы "мозг" мобильной робототехнической системы. Она на основе информации, поступающей из внешней среды и от подсистем самой робототехнической системы через модули "сенсорное устройство" (СУУ и “устройство общения” (УО), планирует и организует целенаправленное поведение автоматической питательно-исполнительной системы (ПИС), управляя ею и производственными объектами внешней среды через модуль "устройство преобразования программ" (УПП)

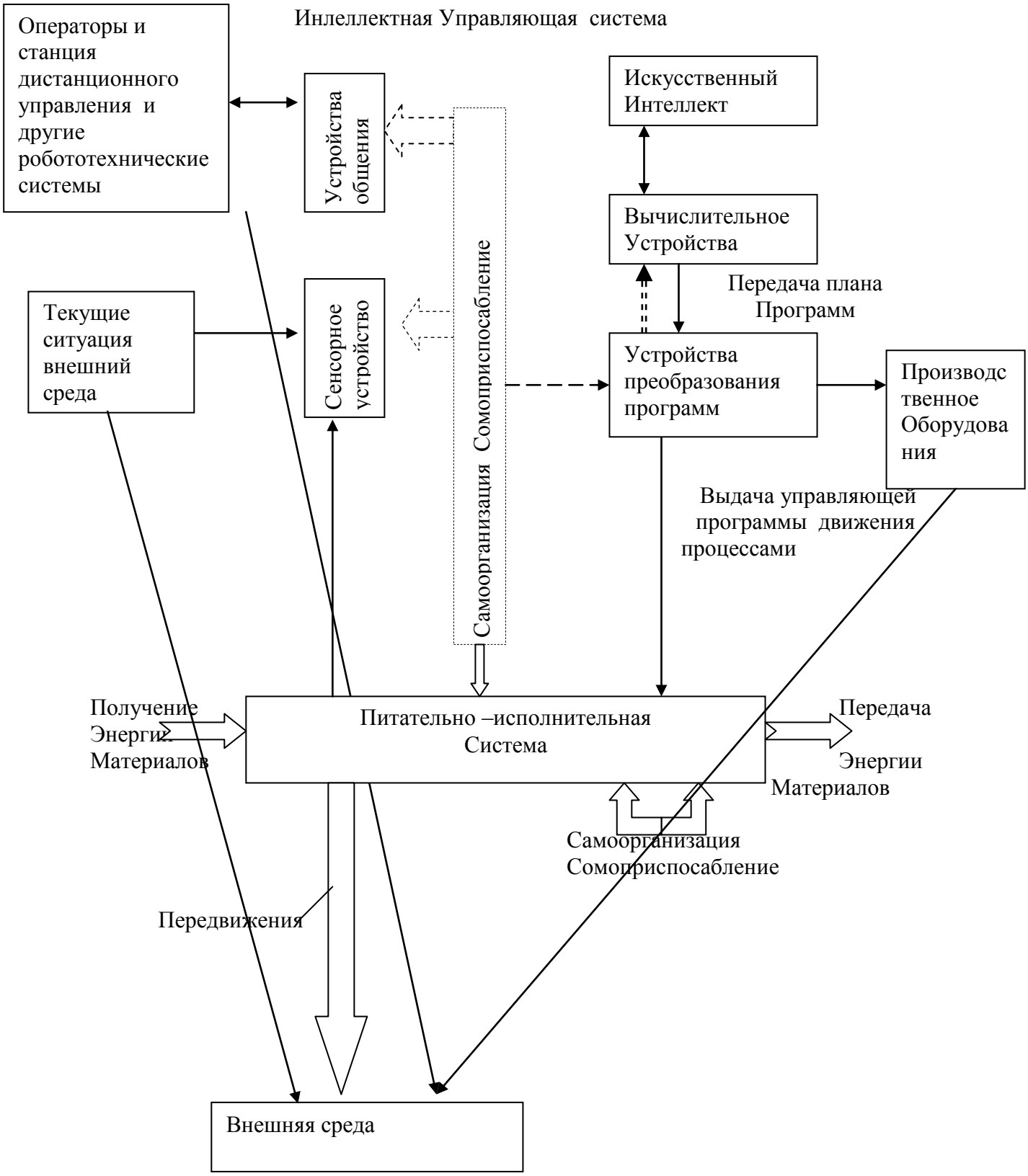
На основе интеллектного управления осуществляется целенаправленное материально-энергетическое взаимодействие ПИС с внешней средой и адаптация сенсорного устройства, устройства общения и узлов самой ПИС к изменениям внешней среды. На рис.3.1. показаны основные (сплошные линии) и допустимые (пунктирные линии) виды информационных (тонкие линии) и материально-энергетических (двойные линии) взаимодействий в системе “мобильная робототехническая система - внешняя среда”(МРС-ВС) .

Из-за открытости и динамичности внешней, по отношению к МРС, среды, внешний, по отношению к модулю "искусственный интеллект", мир (проблемная область) также является открытым, развивающимся. Проблемной областью (внешним миром) для модуля "искусственный интеллект" выступает система "МРС - ВС".

Термин "искусственный интеллект"(введен в 1956 г . ДК. Маккарти)
употребляется в следующих смыслах :как альтернатива интеллекту .
естественного происхождения (интеллект искусственных систем); как искусственная система, имитирующая решение сложных задач человеком в процессе его жизнедеятельности (интеллектная система); как область научных исследований, сопровождающих и обуславливающих создание интеллектных систем, занимающаяся изучением интеллекта систем естественного и искусственного происхождения (область искусственного интеялекта).

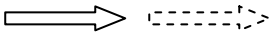
Глобальные проблемы в области искусственного интеллекта
являются важнейшими и в области интеллектных робототехни-ческих систем. К ним относятся:

1. - проблема представления знаний;
2. - проблема планирования действий и поведения МРС;
3. - проблема организации информационного (проблема общения) и материально-энергетического (проблема организации поведения) взаимно- действия МРС с внешней средой.



31 Рис . Мобильная Робототехнические система с интеллектным управлением :

-Основное ,допустимое информационное взаимодействие

 -Основное , допустимое материально- энергетическое взаимодействие

Понятие "интеллектуальная система" определяется в тесной связи с понятиями "интеллект" и "знание".

Интеллектной системой называется искусственная система, способная извлекать, накапливать, хранить, корректировать и обобщать знания, а также использовать имеющиеся знания в соответствии с поставленной целью для нахождения эффективных решений задач [6]

В широком смысле слова "интеллект" (от латинского *Intellects* -познание, понимание, разум, рассудок) есть способность мышления, рационального познания [14]. Интеллект искусственных систем, как правило, оценивается по аналогии с поведением человека в аналогичных ситуациях.

Интеллект - это то, что оценивается в интеллектуальных тес-тах [11].
Под интеллектом робототехнической системы будем понимать способность ее управляющего устройства решать интеллектные задачи, оперируя знаниями и данными из проблемной области, Эту способность система приобретает на основе знаний из области искусственного интеллекта.

Поэтому интеллектная система должна обладать знаниями по крайней мере двух уровней: знаниями, отражающими закономерности проблемной области ,и знаниями об интеллекте (иначе, интеллектными знаниями).

20. ЗНАНИЯ И ЗНАКИ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ.

Познание. В деятельности людей знание есть проверенный практикой результат познания действительности, верное ее отражение в мышлении человека *[14]* .

Процесс познания включает в себя чувственное познание и абстрактное мышление. Познание начинается с отражения окружающего мира органами чувств, дающих непосредственное знание о действительности и являющихся источником всех наших знаний [2]. Чувственное познание протекает в трех основных формах: ощущение, восприятие, представление,

Ощущение - это отражение отдельных чувственно воспринимаемых свойств предметов материального мира: цвета, формы, запаха, вкуси и т.д.

Целостный образ предмета, возникающий в результате непосредственного воздействия последнего на органы чувств, называется восприятием.

Представление - это сохранившийся в сознании чувственный образ предмета, который воспринимался раньше. Если восприятие возникает лишь в результате непосредственного воздействия предмета на органы чувств, то представление имеется тогда, когда такое воздействие уже отсутствует.

В отличие от чувственного познания мышление отражает внешний мир в абстракциях и является внешней по сравнению с чувственным познанием формой отражения действительности. Отвлекаясь от конкретного в вещах и явлениях, от их индивидуальных особенностей, абстрактное мышление способно обобщать множество однородных предметов, выделять наиболее важные свойства, раскрывать существенные связи.

Особенностями абстрактного мышления являются обобщенный и посредственный характер отражения действительности и неразрывная связь с языком [2],

Основными формами абстрактного мышления являются: понятие, суждение и умозаключение.

Понятие - это форма мышления, отражающая предметы в их существенных признаках. Признаком предмета называется то, в чем предметы сходны друг с другом или чем они друг от друга отличаются. Любые свойства, черты, состояния предмета, которые так или иначе характеризуют предмет, выделяют и позволяют распознать его, составляют его признаки. Признаками могут быть не только свойства, принадлежащие предмету: отсутствующее свойство (черта, состояние) также служит признаком [2].

Суждение - представляет собой определенный способ отражения отношений предметов действительности, выраженный в форме утверждения или в форме отрицания.

Умозаключение - это форма мышления, посредством к.торой из одного или нескольких суждений выводится новое суждение.

Формы мышления отличаются типами связываемых элементов мысли: понятие связывает признаки, суждение- связывает понятия, а умозаключение связывает суждения.

Знания. Знания в интеллектных системах представляются в формализованном виде и имеют такое же важное знамён данные (факты и идеи, представленные в формализованном I программы в традиционных системах переработки информации. Знания по отношению к данным могут выступать в роли "генератора данных". В этом смысле знания совместно с некоторыми исходными данными являются компактной формой задания совокупностей данных.

Знания, в отличие от данных, всегда содержательно интерпретируемы и характеризуются существованием иерархических и ситуативных связей между отдельными единицами (частями) знаний, В иерархической структуре знаний данные соответствуют наинизшему (нулевому) уровню. В этом смысле данные - элементарные знания. Поэтому, когда

говорится о представлении знаний, имеется в виду представление и знаний, и данных, но НУ наоборот. Представление знаний в интеллектных системах - это форма совокупного выражения знаний и описаний исходных данных и требуемых результатов(Условий) решаемой задачи. Структурными единицами знания являют понятия, доказательные и правдоподобные суждения, умозаключения и процедуры решения задач. Примерами структурных единиц являются: понятие "система"суждение "робот часть комплекса", дедуктивное умозаключение "Если робот часть комплекса , и комплекс часть системы, то робот часть системы", процедура заказа оборудования.

Структуру знаний интеллектной системы можно отобразить : ИЛИ графом (рис. 32), вершины которого указывают типы ever u частей знания. Исходящие из И - вершин дуги направлены к составным частям, а дуги, исходящие из ИЛИ - вершин, к альтернативным

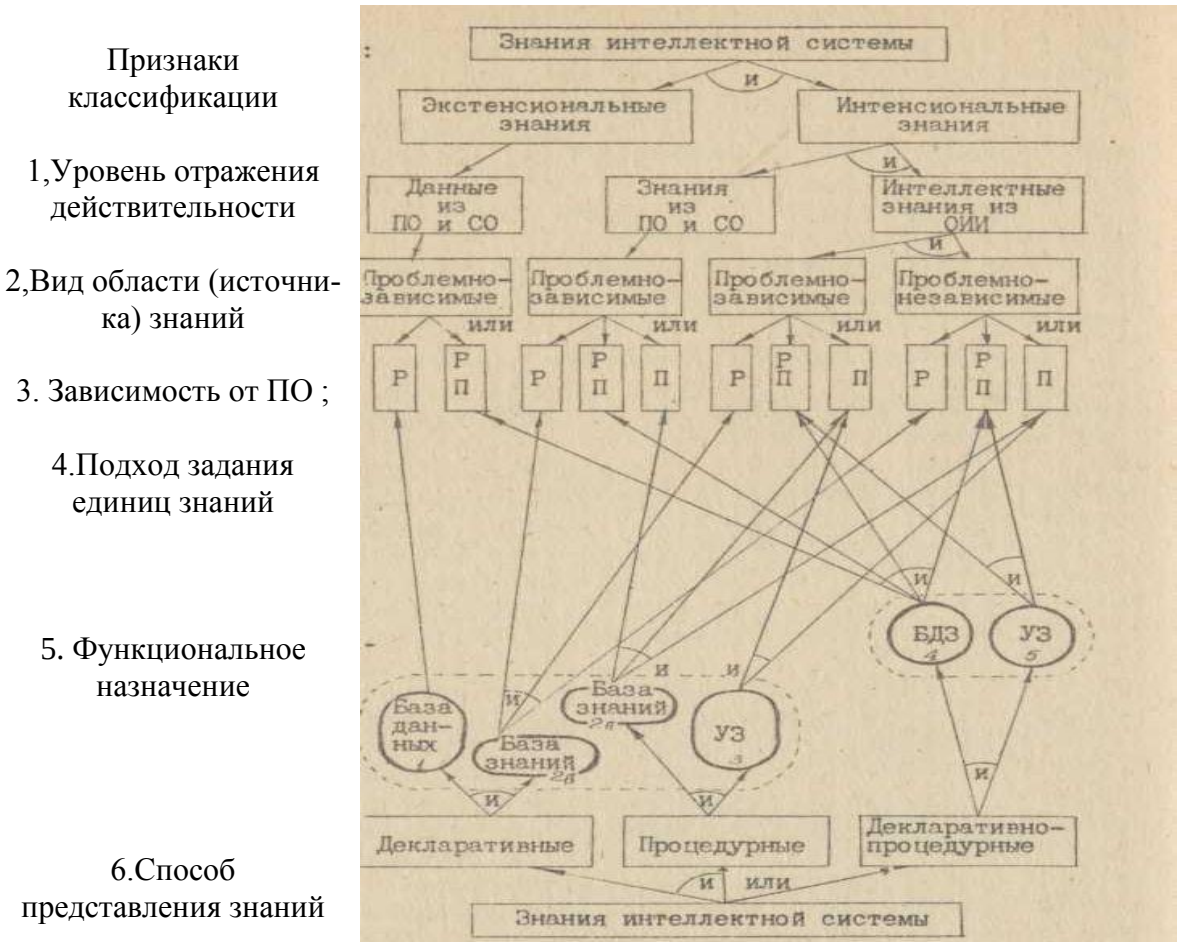


Рис. 32. Знания интеллектной системы:
Р - реляционный подход ,П - операционный подход ,РП - реляционно-операционный подход, ПО т проблемная область, Со - собственный опыт,ОИИ - область искусственного интеллект а,БДЗ - база данных и знаний ,УЗ - управляющие знания(планировщик)

вариантам реализации представления знаний (частей знаний). В И-ИЛИ вершинах исходящие дуги имеют конъюнктивные и дизъюнктивные связи.

Направляясь сверху вниз по графу до уровня четвертого признака , можно проследить виды составных частей знания и варианты их реализации, а в направлении снизу вверх до вершин четвертого признака - особенности реализации альтернативных способов представления полного знания интеллектной системы.

На И-ИЛИ графе части знаний классифицированы по уровню отражения действительности, виду области знаний, зависимости знаний от проблемной области, принятому подходу задания единиц знаний, способу разбиения полных знаний искусственного интеллекта на части, способу представления знаний.

Знаки. Материальным носителем знаний являются знаки.

Знания в интеллектных системах представляются некоторой знаковой (семиотической) системой

$$ЗС=<S,R> \quad (1)$$

где *З* - множество знаков; *R* - множество отношений между ними. В деятельности людей знак - материальный чувственно-воспринимаемый предмет (явление, действие), который выступает как представитель другого предмета, свойства или отношения [14] . Знак как разновидность материального носителя информации (сигнала) создается в самоорганизующихся системах и предназначается для передачи и хранения информации. Это отличает знак от сигналов естественного происхождения.

Знак характеризуется "обозначаемым" (денотатом), "смыслом" (концептом), т.е. свойством (атрибутами) обозначаемого объекта, экстенсионалом и интенсионалом, Интенсионалом знака называется характеристика "смысла", выраженная через общие свойства "обозначаемого" и его отношение с другими"обозначаемыми". Интенсионал знака, также как понятие "образ" в распознавании образов, за-: дается некоторой процедурой, позволяющей определить принадлежность того или иного конкретного объекта к данному знаку. Содержательно интенсионал знака совпадает с понятием, отражающим однородный класс предметов, выступающих в роли обозначаемых. Формально, интенсионал знака будем представлять набором пар вида < атрибут обозначаемого, класс значений атрибута >. Конкретную реализацию интернационала (прообраз *)* назовем фактом и будем представлять набора . пар вида <атрибут обозначаемого, значение атрибута > и назовем фактом.

Будем различать взаимнооднозначные друг другу объектный и фактический экстенсионалы. Объектным экстенсионалом знака называется конкретный класс всех его объективно-реальных "обозначаемых". Однако, охарактеризовать полностью знак его объектным экстенсионалом невозможно, так как при указании на "обозначаемого" из проблемной области неясно, какие из его свойств имеются в виду. Этого можно избежать, если использовать понятие фактического экстенсионала, определяемое как множество реализации интенсионала.

Фактическим экстенсионалом знака назовем набор конкретных фактов, являющихся истинными реализациями (прообразами) интенсионала знака рассматриваемой ситуации. Содержательно элемент фактического экстенсионала знака - это представление о конкретном предмете. Если интенсионал фиксирует некоторые закономерности, всегда присущие объектам проблемной области, то объектный экстенсионал фиксирует подлинные обозначаемые, а фактический экстенсионал - мгновенные снимки "обозначаемых", конкретные факты проявления этих закономерностей. В дальнейшем под “экстенсионалом” будем иметь в виду фактический экстенсионал.

Знаки естественного языка. Наиболее употребительными знаками естественного языка при представлении знаний о проблемной (предметной) области являются имена предметов. Имена предметов - это отдельные слова и словосочетания, обозначающие предметы и выражающие понятия о них. Понятие- это форма мышления, отражающая предметы в их существенных признаках. В понятии сосредоточено то общее, что объединяет отдельные, конкретные предметы в один класс.

В логике под предметом понимают все то, что можно мыслить: существующие в реальной действительности и воображаемые вещи, явления, события, их свойства (значение свойств) и отношение [8].

Предмет - это все то, что может находиться в отношении или обладать каким-либо свойством. Отношение - свойство одного или нескольких предметов . Свойство - категория, выражающая такую сторону предмета, которая обуславливает его различие или общность с другими предметами и обнаруживается в его отношении к ним.

При построении модели внешнего мира берется во внимание необходимое, существенное, неотъемлемое свойство предмета, называемое его атрибутом.

Вещи - это предметы материального мира, обладающие относительной независимостью и устойчивостью существования.

При представлении знаний о проблемной области к предметам будем относить вещи и те явления, события, значения атрибутов и отношения, свойства (или отношения) которых должны быть отражены в знаниях. А те действия (явления и события), значения атрибутов и отношения, свойства (или отношения) которых не учитываются в системе представления знаний, будем представлять отдельно от предметов.

На основе вышеизложенного к важнейшим знакам естественно- . го языка можно отнести следующие 4 группы слов и словосочетаний: имена предметов (знаки-предметы); имена значений атрибутов (знаки-значения атрибутов); имена действий (знаки-операции); имена отношений (знаки-отношения). Часть из последних трех групп знаков может выступать в роли знаков предметов, если свойства обозначаемых ими значений атрибутов, действий или отношений должны быть отражены в модели проблемной области.

При составлении модели по тексту на естественном языке к знакам-предметам будем относить все то, о чем говорится в предложении; к знакам-значениям атрибутов - то, что говорится о предметах и действиях; к знакам-операциям - сказуемые, выражаемые глаголом; отношениям - связи между предметами, действиями и значениями атрибутов [1]. Более подробная классификация знаков естественного языка приведена в [9].

Имена предметов бывают общими и индивидуальными (единичными). Общие имена обозначают однородные классы вещей, явлений, событий и других элементов в реальном мире.

Примерами общих имен предметов могут служить: РОБОТ, МАНИПУЛЯТОР, ТЕЛЕЖКА, ТРАКТОР, КУЛЬТИВАЦИЯ, УБОРКА, КОНЕЦ-СМЕНЫ, . ЗАТШНЕНИЕ и т.п. Здесь слово РОБОТ обозначает не какой-то конкретный предмет реального мира, а робот вообще, любой предмет, . который можно назвать роботом. Словосочетание КОНЕЦ-СМЕНЫ также не конкретизировано, и событие, описываемое данным словосочетанием, может происходить неизвестно где и неизвестно Когда [8]

В качестве общих имен предметов, как правило, используются термины. термин - это слово или словосочетание, обозначающее строго определенное понятие и характеризующееся однозначностью в данной проблемной области. Таким образом, общее имя предмета выражает абстрактное понятие, имеющее "смысл" и "обозначаемые" и реальном мире.

Единичные имена предметов служат для конкретизации тех или иных элементов действительности.

Примерами имен могут служить: манипулятор номер 2, переднее колесо номер 2, шпиндельный барабан номер 4, грядка номер 30, источник энергии данной робототехнической системы, схваты данного манипулятора и т.д.

Имена значений атрибутов (знаки-значения атрибутов) обозначают вещи, действия (явления, события) или свойства, выступающие в роли значений атрибутов предметов.
Примерами знаков -значений атрибутов могут служить: выше-рассмотренные общие и единичные имена предметов, когда используются как значения свойств; слова и словосочетания, выражающие значения параметров и характеристик предметов (10 метров , -15 кг, 60 км/час, 20 секунд, 5 ампер, красный, тяжелый и т.п.) .

Имена действий (знаки-операции, знаки-функции, знаки-действия) описывают действия и служат для выражения динамики. В [7] знаки-операции разделены на императивы (включить, снизить скорость, двигаться к... и т.п.), действия-процессы (нагружать, перевозка груза,

регулировка рабочей щели и т.п.); действия-состояния (БЫТЬ ИСПРАВНЫМ, БЫТЬ СВОБОДНЫМ и т.п.).

Имена отношений (знаки-отношения) обозначают связи между предметами и значениями их атрибутов и операциями.

В [7] знаки-отношения разбиты на II групп: отношения классификации, сравнения, принадлежности; признаковые, количественные, временные, пространственные, каузальные (причинно-следственные), инструментальные, информационные и порядковые отношения. Примерами знаков - отношений могут служить: ИМЕТЬ ИМЯ, БЫТЬ ПОДКЛАССОМ КЛАССА, ЧАСТЬ-ЛДЕЛОЕ (классификационные отношения); ИМЫГЬ ПРИЗНАК, ПРИЗНАК-ЗНАЧЕНИЕ (признаковые отношения); МЕРА- ЗНАЧЕНИЕ (количественные отношения); РАВНО, БОЛЬШЕ, МЕНЧИЕ, НЕСРАВНИМО (отношения сравнения), ОДНОВРЕМЕННО, БЫТЬ РАНЬШЕ, НАЧИНАТЬСЯ ОДНОВРЕМЕННО, КОНЧАТЬСЯ ОДНОВРЕМЕННО, СОВПАДАТЬ ВО ВРЕМЕНИ (временные отношения); БЫТЬ СЛЕВА, БЫТЬ СПРАВА, НАХОДИТЬСЯ НА, БЫТЬ ВНУТРИ, КАСАТЬСЯ(пространственные отношения); БЫТЬ ЦЕЛЫ), ПРИЧИНА-СЛЕДСТВИЯ (причинно-следственные); ВЫТЬ СРЕДСТВОМ ДЛЯ,БИТЬ ИНСТРУМЕНТОМ ДЛЯ, СПОСОБСТВОВАТЬ (инструментальные отношения); ВЫТЬ ИСТОЧНИКОМ ИНФОРМАЦИИ (информационные отношения); БЫТЬ СШВДУЮЩИМ, БЫТЬ БЛИЖАЙШИМ,БЫТЬ ОЧЕРЕДНЫМ (Порядковые отношения).

Приведенные знаки соответствуют статическим отношениям. К динамическим знакам-отношениям относятся отношения, связанные с перемещением предметов во времени и пространстве (ДЕЙСТВИЕ-МЕСТО, ДЕЙСТВИЕ-ВРЕМЯ) и рассмотренные выше знаки-операции.

Составные части знания интеллектной системы. Полные знания интеллектной системы состоят из экстенсиональных и интенсиональных знаний, представляемых соответственно экстенсионалами и интенсионалами знаков, относящихся к проблемной области и к области искусственного интеллекта (рис. 32).

Знания из проблемной области являются проблемно-зависимыми и составляют основу баз данных и знаний интеллектной системы. Часть этих знаний меняется от задачи к задаче, другая их часть остается неизменной при всех ситуациях в проблемной области.

Интеллектные знания подразделяются на проблемно-зависимые и проблемно-независимые и составляют основу управляющих знаний и баз знаний (декларативных и процедурных). Управляющие знания в основном предназначены для организации поиска решений интеллектных задач. Основная их часть носит универсальный характер (например, такие знания, как правила дедуктивного логического вывода, стратегии сведения задач к совокупности подзадач, стратегии порождения точек пространства поиска и т.п.), Часть знаний может быть более специфичной для соответствующих проблемных областей.

Структурные единицы знаний можно описать реляционным (от английского слова *Zelation* - отношение), операционным или реляционной -операционным подходом. Эти формы различаются типом используемых знаков, несущих основную смысловую нагрузку в знаковых выражениях, В реляционном подходе основными единицами (атомами) знаний выступают *n* - арные отношения (предикаты) (Р), в операционном подходе - операции (П), в реляционно-операционном подходе -отношения и операции *(Р,П).* В экстенсиональных знаниях основная смысловая нагрузка содержится в экстенсионалах знаков. В зависимости от используемого подхода задания единиц интенсиональных знаний основная смысловая нагрузка может содержаться в интенсионален знаков-отношений (Р) или (VI) знаков-операций (П;ПР).

Эти единицы знаний, в зависимости от их формы представлений и функционального назначения, служат элементами базы данных, базы знаний, базы данных и знаний или управляющей структуры, являющихся основными функциональными компонентами интеллектных систем, как это показано на рис. 32.

Способы представления знаний.

Подмножества типов вершин, принадлежащих к дугам, исходящим из вершин 5-и б-уровней на рис. 32., раскрывают содержание составных частей знаний и суть способов представления знаний в интеллектной системе. В соответствии с особенностями разбиения знаний на части и взаимодействия частей знаний различают 3 способа представления знаний: декларативные представления (ДП), процедурные представления (ПП) и декларативно-процедурные представления (ДПП).

Следует заметить, что декларативные (процедурные) знания и декларативные (процедурные) представления не одно и то же. ДП и ПП - это способы представления полных знаний. Декларативные знания основываются на реляционном подходе представления единиц знаний., а процедурные знаиия-на операционном подходе. Обычно вопрос выбора способа представления знания сводят к обсуждению баланса между декларативной и процедурной частями знаний. Однако ? Систем, имеющих только декларативные или только процедурные знания, не существует. Различие между ДП и ПП можно выразить как различие между "Знать что" и "Знать как" [5] ДП исходит из посылки, что знания из проблемной области не имеют глубоких связей со знаниями (процедурами), используемыми из области искусственного интеллекта для обработки этих знаний. ДП полагает, что интеллектность базируется на некотором универсальном множестве процедур (из области ИИ), обрабатывающих факты любого вида, и на проблемно-зависимых знаниях (множестве специфических фактов), описывающих проблемную область [5] . Б ДП текущее знание системы ИМ представляется полным описанием состояния и множеством преобразований или операторов. Полные описания состояния представляются в виде множества утверждений безотносительно к тому» как их использовать. К классу ДП относятся исчисление предикатов, продукционные и редукционные системы [13].

К достоинствам ДП относятся: отсутствие необходимости указывать, как конкретный фрагмент знания должен быть использован; легкость модификации и пополнения знаний; удобство общения.

ПП исходит из посылки, что интеллектная. деятельность есть знание о внешнем мире, вложенное в программы, т.е. знание "как” можно использовать те или иные факты проблемной области [5] , Процедурное представление основано на описании в виде процедур всех возможных манипуляций с фактами. Таким образом, текущее значение системы представляется в виде специально организованной базы данных и набора более или менее специализированных процедур, обрабатывающих соответствующие области базы данных. Решение *в* этом классе представлений сводится к построению целенаправленной последовательности процедур, как правило, имеющей сложную рекурсивную организацию. В процедурном представлении, в отличии от ДП, на каждой стадии решения задачи обрабатываются локальные области базы данных. Носителями ПП являются специальные проблемно -ориентированные языки (ПОЯ).

Целесообразность применения ПП объясняется следующими обстоятельствами:

- существует значительное количество сущностей, которое удобно представить в виде процедур и весьма трудно в чисто ДП (например, действия робота в окружающей его среде удобней представлять в вице процедур);
- ПП, в отличии от ДП, легко позволяет представить знания второго порядка, необходимые для выражения фактов о том, как использовать |>акты;
- ПП, в отличии от ДП, позволяет легко учитывать специфику конкретных проблемных областей, закладывая ее в эвристические знания, которые трудно (иногда невозможно) выразить в универсальных процедурах ДП. *[5]*

Декларативно-процедурный способ представления знаний в определенной мере сочетает преимущества ДП и ПП. К ним относятся фреймы, семантические сети, которые изложены в [5,/2]

Литература:

1. Искусственный интеллект.- В 3-х кн. Кн. 2. Модели и методы Справочник под ред. Д.А. Поспелова, - М.; 1990, –304с.
2. Назаров Х.Н. Робототехнические системы и комплексы. Т.: ТГТУ, 2004, -100с.
3. Лорьер Ж.Л. Системы искусственного интеллекта Пер. с франц. – М.: Мир, 1991. -568 с.
4. Сотник С.Л. Основы проектирования систем искусственного интеллекта - М.: Мир, 1998.
5. Попов Э.В. Экспертные системы. -М.; Наука, 1987, -288с.
6. Хасанов П.Ф. Назаров Х.Н. Мобильные робототехнические системы - ТашПИ, Т, 1987 ,101 с.
7. Тимофеев А.В. Роботы и искусственный интеллект, -М.: Наука, 1978. ,96 с.