МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Брестский государственный технический университет»

Кафедра интеллектуальных информационных технологий

Л.П. Матюшков, В.А. Головко, В.Н. Шуть

ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебно-методический комплекс по дисциплине «Основы искусственного интеллекта»

Рекомендовано редакционно-издательским Советом учреждения образования «Брестский государственный технический университет» для студентов специальности 1-40 03 01 «Искусственный интеллект»

Брест 2010

УДК 004.8 (07)
ББК 32 813
М 33

Рецензенты:
заведующий кафедрой ИИТ БГУИР, доктор технических наук, профессор Голенков В.В.;
ведущий научный сотрудник НАН Беларуси кандидат технических наук, старший научный сотрудник Дудкин В.В.

Матюшков Л.П., Головко В.А., Шуть В.Н.
М 33 Основы искусственного интеллекта: учебно-методический комплекс. – Брест: Издательство УО «БрГТУ», 2010. – 116 с.

ISBN 978-985-493-155-5

В данном учебно-методическом пособии содержится информация по основам знаний в области искусственного интеллекта. Особое внимание уделяется рассмотрению базовых понятий и прикладным аспектам использования теоретических положений в современной технике и обществе, приводится учебная программа для изучения курса, перечень вопросов к зачету и задачи для практических занятий.

Издание адресовано студентам и другим специалистам, интересующимся проблемами развития теории искусственного интеллекта и ее использования в современных информационных технологиях.

УДК 004.8 (07)
ББК 32 813

ISBN 978-985-493-155-5

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное производство и быт человека все в возрастающем объеме насы ются «умными» системами, приборами и машинами, «интеллектуальный» уровень ко рых непрерывно повышается. Технические средства, способные выполнять функц которые человек обычно относит к творческим, стали называть интеллектуальными с темами. Их возможности, разумеется, отличаются в зависимости от оснащенности со ветствующим программным обеспечением, реализующим сложные алгоритмы, вып нение которых поддерживается хитроумными конструкциями современных ЭВМ средств связи. Появились японские разработки роботов, имитирующих не только те ческие возможности человека, но и его образ и движения.

«Электронные» правительства, «безлюдные» производства, «безбумажное» формационное обеспечение, программы-консультанты для принятия решений, слож ния стихов и музыки, доказательства теорем, шахматные системы, побеждающие г ших гроссмейстеров; нейросетевые системы, способные по предъявленным образц находить автоматически алгоритмы для решения задач прогнозирования, распозна ния образов и т.д. Бытовая техника (утюги, стиральные машины, микроволновые печ т.п.) вошли в наш обиход уже под именем «умных» машин.

Вершиной техники стало создание гибких производственных и управляющих с тем, способных перенастраиваться на потребности человека: системы по продаже ав и железнодорожных билетов, роботы-луноходы и сварщики по заданному контуру, «у ные» светофоры; системы, способные изменять структуру технических и программ средств для эффективного решения конкретной поставленной задачи.

Без достаточного уровня компьютерной грамотности ориентация современного ловека (а тем более создателя интеллектуальных систем) невозможна в мире слож автоматики.

Любые системы машин быстро совершенствуются, и им на смену всегда прихо более сложные и эффективные, но законы и принципы, лежащие в основе их создани функционирования, более долговечны и составляют фундамент, воздвигаемый поко ниями людей. Задача данного пособия и состоит в том, чтобы в краткой и доступ форме ознакомить с основами теории этих знаний. Их составляют теория алгоритмо автоматов; формальных языков, баз знаний и данных, операционных систем и сист управления базами данных, нейронных сетей и нейро-ЭВМ; сетей ЭВМ и современн интеллектуальных технологий и др. Цель книги – ознакомить читателя с основны идеями и положениями этих теорий. Одновременно в книге кратко охарактеризуем р зультаты некоторых выдающихся создателей этих теорий и вклад белорусских ученых развитие теории и практики различных аспектов науки об искусственном интеллекте.

Перечисление заслуг выдающихся ученых всего мира могло бы занять мног страницы, поэтому ограничимся небольшим количеством имен ученых, внесших знач тельный вклад в теорию и практику создания систем искусственного интеллекта, а так остановимся на достижениях и работах белорусских ученых в этой области как во ве дении, так и в соответствующих главах книги, где излагается суть рассматриваем достижений.

Среди крупнейших ученых, стоявших у истоков создания интеллектуальных си тем, следует отметить Н. Винера – отца кибернетики, обратившего внимание на необх

димость изучать связь живых организмов с машиной, которая способна принимать решения, приводящие в изумление их создателей; К.Шеннона, который разработал основы теории обработки и передачи информации, а также показал, как из ненадежных элементов строить надежные машины; Т. Саймона, Д.Шоу, опубликовавших результаты работ по созданию универсальной программы «Общий решатель задач» (GPS); Розенблата – автора теории персептрона и многих других.

Большой вклад в создание основ безбумажной информатики внес В.М. Глушков. Диапазон его работ простирался от теории систем автоматизированного управления, информационных технологий до построения систем искусственного интеллекта, использующих для общения с ними человека естественные языки с планированием целенаправленных действий.

В Белоруссии зачинателем теории создания инженерных интеллектуальных систем был Г.К. Горанский, который создал первую в мире лабораторию автоматизации инженерного и управленческого труда (1962), а потом и институт технической кибернетики (1965), где при его участии и руководстве развивалась теория автоматизации инженерного труда с применением элементов искусственного интеллекта в принятии проектных и технологических решений. Большой вклад в создание многомашинных вычислительных комплексов САПР внес О.И. Семенков, с его участием издана монография «Автоматизация поискового конструирования: искусственный интеллект в машинном проектировании».

В.С. Танаев получил ряд выдающихся результатов в теории расписаний,в методах оптимизации и исследования операций. Им совместно с Матюшковым Л.П. разработаны модели и методы генерирования допустимых расписаний с использованием методов искусственного интеллекта.

С.В. Абламейко сделал большой вклад в теорию обработки изображений. На основе его результатов создан ряд проблемно-ориентированных автоматизированных систем обработки графической информации в картографии, медицине, машиностроении и дистанционном исследовании земной поверхности.

А.В.Тузиков получил ряд значимых результатов по распознаванию и анализу стохастических данных и цифровых изображений.

В.В. Анищенко, уроженец Брестского района, внес большой личный вклад в разработку белорусско-российского суперкомпьютера «СКИФ» и семейство аппаратно-программных средств фискализации и защиты информации для компьютерно-кассовых систем.

П.Н. Бибило в теории автоматизации проектирования дискретных устройств и цифровых сверхбольших интегральных схем (СБИС) с применением методов искусственного интеллекта получил важные прикладные результаты, внедренные в НПО «Интеграл» и других организациях.

А.А. Дудкин разработал информационную технологию обработки, анализа и идентификации видеоизображений элементов электронных изделий для задач проектирования и контроля изготовления интегральных схем, которая используется концернами «Планар» и НПО «Интеграл».

А.Д. Закревский заложил основы компьютерной дискретной математики для синтеза и анализа логического проектирования дискретных устройств, распознавания образов, защиты информации и решения логических уравнений. Его результаты внедрены в

системах автоматизированного логического проектирования в ЦКБ «Алмаз» (г. Мос НИИ ЭВМ, НПО «Интеграл».

Б.М. Лобанов исследует вопросы автоматического распознавания и синтеза р его результаты использовались как у нас в стране, так и за рубежом при выполнени вместных научных проектов «Двуязычный синтезатор речи» (с ФРГ), «Распознав зашумленной речи» (с Францией), «Распознавание речи по телефону» (с США) и др.

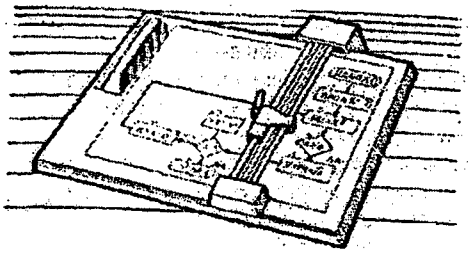
Р.Х. Садыхов получил ряд результатов в теории цифровой обработки сигнал изображений, идентификации и распознавании объектов, защите информации, мод рованию параллельных архитектур вычислительных систем. Им внедрен ряд аппара программных комплексов для стендовых испытаний на вибрацию сложных техниче объектов и их узлов (автомобили, тракторы, ракеты), распознавания рукописных си лов и сигнатур (подписей); цифровая система с использованием нейронных сетей настройки растра телевизионных приемников.

В.В. Голенков получил ряд теоретических и прикладных результатов по разраб и созданию графодинамических и семантических моделей, методам и средствам пар лельной обработки сложноструктурированных знаний в интеллектуальных системах, теллектуальным технологиям в геоинформационных системах. Под его руководст создана и внедрена в Беларуси и России система TESTER для психодиагностики и троля знаний.

В г. Бресте В.А. Головко создана научная школа в области нейронных сетей и кусственного интеллекта. Среди результатов можно отметить разработку теории обу ния нейронных сетей и их практическое использование в управлении автономными бильными роботами, прогнозировании различных процессов, обнаружении компьют ных атак и вирусов, диагностики эпилептиформной активности на основе обработки с налов электроэнцефалограмм головного мозга и др.

В.Н. Шуть разработал и внедрил на Брестском производственном объедине средств вычислительной техники комплекс приборов диагностики ЭВМ: Логоскоп, Э лон, Дукат, УТКД-Брест1 и др. Его исследования свойств графов типа «деревья» пользовались для генерации тестов контроля многослойных печатных плат и при созд нии станции диагностики объемного монтажа ЭВМ.

В.Н. Ярмолик разработал теорию многоканальных синатурных анализаторов, ко рые были успешно использованы В.Н. Шутем в диагностических устройствах контро алгоритмических тестов (ДУКАТ). Им также были разработаны методы встроенного те тирования цифровых узлов ЭВМ, в особенности, оперативных ЗУ. Предложенные им нераторы псевдослучайных последовательностей применены в криптографии, а также схемах самотестирования.



Глава 1. АЛГОРИТМЫ И ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ

1.1. Автоматы и алгоритмы

Предшественниками вычислительных машин являются автоматы – устройства, выполняющие некоторый процесс без участия человека. Появление автоматов относится к глубокой древности. Это были различные механические игрушки, автоматы для открывания дверей в храмах и т.п. Одно из отличий вычислительных устройств наших дней от традиционных средств состоит в том, что они стали машинами, пригодными для автоматизации умственной деятельности человека. Теория автоматов вошла в обиход в 50-е годы XX столетия, хотя соответствующая проблематика начала складываться еще в тридцатые годы в рамках теории автоматов и релейных устройств. Тогда в теории алгоритмов были сформулированы понятия вычислительного автомата (машины Тьюринга и Поста). Было установлено, что для осуществления всевозможных преобразований информации вовсе не обязательно строить каждый раз специализированные автоматы: все это можно сделать на одном универсальном автомате при помощи подходящей программы и соответствующего кодирования. Этот теоретический результат позже получил инженерное воплощение в виде современных универсальных вычислительных машин.

Различие между задачами теории алгоритмов и теории автоматов можно кратко охарактеризовать как различие между вопросами о том, что могут делать автоматы и как они это делают.

Обычно вычисления или машинные преобразования, связанные с решением задачи, представляют собой многошаговый процесс последовательных действий. До начала вычислений надо выбрать метод решения задачи и иметь предписание об операциях над исходными данными и порядке их выполнения. Предписание, определяющее порядок выполнения операций над данными с целью получения искомого результата, в нестрогом понимании называется алгоритмом.

Процесс подготовки решения задачи на ЭВМ часто называют алгоритмизацией. Обычно под алгоритмизацией процесса решения некоторого класса задач понимают описание последовательности действий, которые необходимо выполнить, чтобы задать процесс в виде однозначно определенной цепочки операций на языке математических символов. Чаще всего приходится начинать его со словесной формулировки при постановке задачи. Затем с помощью формул, таблиц, схем и т.п. описывается метод решения задачи. В соответствии с предлагаемым методом выбирается алгоритм решения.

Естественно, что понятие алгоритма в теории алгоритмизации занимает центральное место. Над уточнением понятия «алгоритм» работали многие ученые. На основе

этих работ развилась теория вычислимых функций, конечных и бесконечных автоматов, являющихся математическими моделями вычислительных машин.

Вычислительная машина, вообще говоря, перерабатывает входную последовательность знаков в выходную. Операции над этими последовательностями практически сводятся к применению ряда подстановок, которые позволяют выполнять различные преобразования: кодирование информации в вид, удобный для восприятия машиной, переработку информации по некоторому алгоритму, воспроизведение результата в удобной для человека форме.

При переработке информации рассматриваются конечные и бесконечные последовательности различных символов (букв). Конечные последовательности легко обозримы, но что значит «рассмотреть» бесконечную последовательность?

Представление о такой последовательности можно составить лишь по описанию тех или иных ее свойств. Например, определить по номеру места символ в бесконечной последовательности 01 01 01 ... . Если рассматривать только начало последовательности, то никаких представлений о ней в целом сделать нельзя. Однако такое представление создается, если указывается, что 0 и 1 чередуются всегда. Теперь можно предсказать, какой символ, например, стоит на 93-м и 1026-м месте.

В этом примере свойство, благодаря которому имелась возможность составить представление о всей бесконечной последовательности, заключалось в существовании процедуры, позволяющей узнавать по номеру места цифру, стоящую на этом месте. Точнее можно сказать, что для этого примера имелся алгоритм распознавания символа по номеру места.

Вообще под алгоритмом часто понимается некоторое формальное предписание, действуя согласно которому, можно получить нужное решение задачи. Эта формулировка соответствует интуитивной точке зрения на алгоритм, сложившейся еще в древности.

Классическим примером для ее иллюстрации является алгоритм Евклида для нахождения наибольшего общего делителя положительных чисел a и b.

1. Обозревай данные числа a и b. Переходи к указанию 2 (ПКУ2).

2. Сравни обозреваемые числа a = b, или a<b, или a> b. ПКУ3.

3. Если обозреваемые числа равны, то каждое из них дает искомый результат, процесс вычислений останавливается. Если нет, то ПКУ4.

4. Если первое обозреваемое число меньше второго, то переставить их местами. ПКУ5.

5. Вычитай второе число из первого и обозревай два числа: вычитаемое и остаток. ПКУ2 (процесс повторяется по вышеназванной схеме).

Рассмотрим пример, иллюстрирующий применение алгоритма Евклида, указывая результаты и номера выполняемых операций для чисел a = 15 и b = 20.

1. ПКУ2.
2. 15<20. ПКУ3.
3. ПКУ4.
4. 20, 15. ПКУ5.
5. 20-15 = 5. ПКУ2.
2. 15>5. ПКУ3.
3. ПКУ4.
4. ПКУ5.
5. 15-5 = 10. ПКУ2.
2. 5<10. ПКУ3.

3. ПКУ4.
4. 10,5. ПКУ5.
5. 10-5 = 5.ПКУ2.
2. 5 = 5. ПКУ3.
3. Остановка процесса, результат 5.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к арифметическим действиям, называются численными. Численными являются также любые алгоритмы (схемы) для решения некоторого класса задач, если формулами полностью описан как состав действий, так и порядок их выполнения.

Указания, задающие элементарные действия, называются операторами. Реализация оператора сводится к переработке некоторой информации и к определению оператора, выполняемого вслед за данным.

Понятие элементарного действия (операции) существенно зависит от путей реализации алгоритма. Если алгоритм рассчитан на выполнение человеком, то ему может соответствовать совершенно другой уровень описания, чем для ЭВМ. Уровень описания алгоритма зависит от квалификации будущего исполнителя или же оснащенности вычислительной машины специальными программами и техническими средствами. Чем ниже будет квалификация исполнителя алгоритма или же оснащенность вычислительной машины, тем детальнее необходимо задавать алгоритм. Каждый шаг алгоритма должен описываться строго однозначно.

Наряду с арифметическими операциями над числами, часто встречаются различные логические операции. Алгоритмы с преимущественным использованием этих операций называют логическими.

Классическим примером логического алгоритма является задача поиска пути в лабиринте.

Лабиринт задается в виде конечной системы площадок, от которых расходятся коридоры. Каждый коридор соединяет только две смежные площадки. Геометрически лабиринт можно представить в виде кружков, соединенных отрезками (граф).

Коридору будет соответствовать ребро графа, а площадке – вершина. Будем говорить, что площадка У достижима с площадки Х, если существует путь, ведущий от Х к У через промежуточные коридоры и площадки. Очевидно, что если площадка У вообще достижима с площадки Х, то она достижима посредством простого пути, когда каждая из площадок проходится один раз, например (рис. 1.1.1), простой путь с площадки А в D: ABD или ABCD. Площадка К с площадки А вообще недостижима.

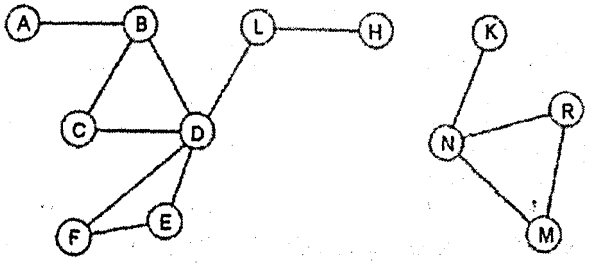


Рисунок 1.1.1 – Модель лабиринта

Решим задачу: начав поиск с площадки А, выяснить, достижима ли произво площадка У.

Если она достижима, то найти путь из А в У, а в ином случае после поиска вер ся в А. Устройство лабиринта заранее неизвестно. Под решением задачи понима указание общего метода поиска для любого расположения площадок А и У. Рассмо один из возможных методов поиска. Предположим, что в начале и конце проход коридоров можно делать любые виды пометок, по которым легко узнать, проходил данный коридор и сколько раз. Коридоры, которые не проходились ни разу, будут пометок; пройденным один раз будем присваивать пометку 1, а пройденным дваж пометку 2.

Находясь на какой-либо площадке, можно попасть на смежную двумя путями: хождением по неотмеченному коридору, пометив его начало и конец знаком 1; воз том с данной площадки по коридору с пометкой 1, заменив пометку коридора знако Находясь на какой-либо площадке, можно различать следующие признаки:

1) это площадка У (цель достигнута);

2) петля (от данной площадки расходятся, по крайней мере, два коридора с пом кой 1 и неотмеченных коридоров нет);

3) от данной площадки отходит, по крайней мере, один коридор без пометок;

4) это исходная площадка А;

5) отсутствие всех предыдущих признаков.

В соответствии с этими признаками ходы делаются до тех пор, пока не насту остановка. Относительно предложенного метода справедливы следующие утвержден

1.При любом расположении А и У за конечное число ходов наступит остановка бо в А, либо в У.

2.Если остановка на площадке А, то площадка У недостижима.

Таблица 1.1 – Метод поиска пути к площадке Y

Признак	Ход
Площадка У	Остановка
Петля	Возврат по последнему пройденному коридору
Имеется коридор без пометок	Движение по одному из коридоров без пометок
Площадка А	Остановка
Отсутствие 1-4	Возврат по пройденному коридору

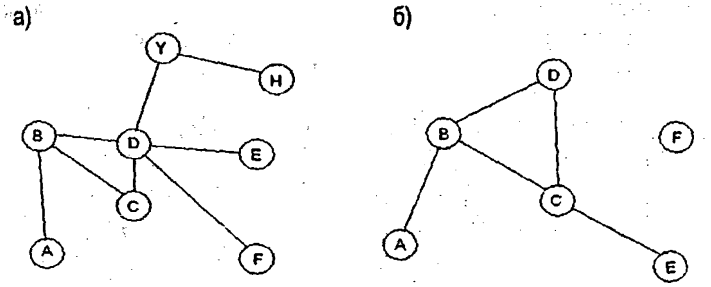


Рисунок 1.1.2 – Лабиринты: а) с достижимыми площадками; б) с недостижимыми площадками

Рассмотрим два небольших примера (табл. 1.1.2 и 1.1.3, рис. 1.1.2). Процесс поис ка пути из А в У будем фиксировать по шагам.

Таблица 1.1.2 – Поиск достижимой площадки F в лабиринте а)

Номер хода	Признак	Направление движения	Пометка коридора
1	3	AB	1
2	3	BC	1
3	3	CD	1
4	3	DY	1
5	3	YH	1
6	5	HY	2
7	5	YD	2
8	3	DB	1
9	2	BD	2
10	3	DF	1
11	1	Остановка	

Таблица 1.1.3 – Недостижимость площадки F в лабиринте б)

Номер хода	Признак	Направление движения	Пометка коридора
1	3	AB	1
2	3	BC	1
3	3	CE	1
4	5	EC	2
5	3	CD	1
6	3	DB	1
7	2	BD	2
8	2	DC	2
9	2	CB	2
10	2	BA	2
11	4	Остановка	

Этот метод поиска содержит элемент произвола, которого не было в ранее рассмотренных примерах.

В алгоритме Евклида операции, сделанные разными вычислителями, совпадают во всех деталях.

Здесь же от одной площадки отходит несколько коридоров, и выбор можно делать произвольно. Алгоритмом обычно называют строго детерминированную систему. Поэтому дополним алгоритм требованием выбирать первый коридор по часовой стрелке при возникновении неоднозначного продолжения.

Общие свойства алгоритмов. Из рассмотренных примеров отчетливо выступают свойства, присущие любому алгоритму.

Дискретность алгоритма. Алгоритм – процесс последовательного построения величин, идущих в дискретном времени таким образом, что в начальный момент задается исходная конечная система величин, а в каждый следующий момент система величин строится по определенному закону из системы величин, имевшихся в предыдущий момент времени.

Детерминированность алгоритма. Система величин, получаемых в какой-то не начальный момент времени, однозначно определяется системой величин, полученных в предыдущие моменты времени.

Элементарность шагов алгоритма. Закон получения последующей системы величин из предыдущей должен быть простым и локальным.

Направленность алгоритма. Если способ получения последующей величины предыдущей не дает результата, должно быть указано, что считать результатом.

Массовость алгоритма. Начальная система величин может выбираться из не торого бесконечного множества. Иными словами, алгоритм служит для решения цел класса задач.

Область применения. Областью применения алгоритма называется такая н большая область начальных данных, на которой алгоритм результативен.

Относительно перечисленных свойств алгоритма укажем их интерпретацию алгоритма Евклида:

Дискретность. В любой момент по паре чисел (a, b) строится новая пара (a, b).

Детерминированность. Пара (a, b) определяется однозначно.

Элементарность шагов. Вычитание двух чисел, сравнение, перестановка.

Направленность. Указано правило прекращения процесса и то, что считается зультатом выполнения каждого шага.

Массовость. Алгоритм применим к любой паре целых чисел a>0, b> 0.

Область применения $\{a, b\} \in \{1, 2, ...\}$.

Следует отметить, что число операций при выполнении того или иного алгорит заранее неизвестно и зависит от выбора исходных данных. Поэтому под осуществим стью алгоритма следует понимать потенциально возможный процесс для конкретн задач из области его применения, так как для решения некоторых из них может не хв тить ни времени, ни памяти.

1.2. Средства описания алгоритмов

Всякий алгоритм, как уже говорилось ранее, отражает последовательность преоб разований информации. Алгоритм обычно состоит из операционной и информационно частей.

Естественно, что при описании алгоритмов нужно задавать операторы и связи ме жду ними. Как правило, оператор представляет собой довольно простую конструкцию, поэтому наибольшие трудности при разработке алгоритма приходятся на указание свя зей между операторами, т.е. на описание структуры алгоритма.

При построении арифметических операторов предпочитают включать в каждый и них определенное законченное действие, например вычисление определителя, извле чение корня и т.п. На этом этапе последовательность размещения операторов не рас сматривается.

Для осуществления вычислений на ЭВМ существенна очередность арифметиче ских операторов, так как один из операторов может обеспечить возможность выполне ния следующего. При построении описания вычислительной схемы алгоритма необхо димо разместить арифметические операторы в такой последовательности, чтобы вы полнение вычислений одного из них обеспечивало возможность реализации следующих за ним операторов. Естественно, что в частных случаях перестановка некоторых операторов местами допускается.

Каждый разработчик алгоритма, руководствуясь накопленным опытом и знанием закономерностей алгоритмизуемого процесса, изображает его в соответствии со своими представлениями. Однако можно выделить некоторые типичные шаги этой работы и наиболее употребительные средства описания алгоритмов.

1. Разработка алгоритма начинается с изучения задания, данного для алгоритмизации. Оно часто представляется в описательной форме с использованием формул, таблиц, графиков и т.п. Перед разработчиками алгоритма возникает необходимость глубоко изучить алгоритмизуемый процесс, уяснить закономерности составляющих его явлений, установить все факторы, влияющие на его течение, оценить степень влияния отдельных явлений на окончательный результат и т.д.

Необходимо определить входную и выходную информацию, задать области изменения аргументов, точность вычислений.

2. Входная информация должна быть полной для обеспечения получения всей выходной информации. Входная информация бывает двух видов: постоянная и переменная. Постоянная входная информация сохраняет значение в процессе счета по всему алгоритму. Переменная информация зависит от конкретной частной задачи, решаемой в данный момент.

При разработке алгоритмов универсальность построенного алгоритма зависит от соотношения объемов переменной и постоянной входной информации. При уменьшении переменной информации, как правило, алгоритм становится менее универсальным.

3. Далее выполняется математическая формализация словесно-описательного условия задачи. Цель ее – построить массивы арифметических {A} и логических {P} операторов. В массив {P} входят все условия, которые отражают закономерности алгоритмизуемого процесса. Массивы {A} и {P} являются тем материалом, из которого строится вычислительная схема алгоритма.

В процессе разработки алгоритмов для их анализа, сокращения числа элементов, удобства программирования и других целей используется много способов описания алгоритмов. Среди них наибольшее распространение получили схемы алгоритмов, операторные схемы, различные таблицы. Названные средства описания имеют определенные достоинства и недостатки. Одни из них более удобны на этапе разработки алгоритма, другие – на этапе анализа, третьи – на этапе минимизации алгоритма, четвертые – на этапе общения человека с ЭВМ и т.д.

4. Совокупность данных, которые в соответствии с условиями задачи должны быть получены в результате ее решения, составляют выходную информацию.

Для иллюстрации применения средств записи алгоритмов приведем простейший пример. Пусть задана последовательность X1, X2, ..., Xi, ..., Xn из n произвольных действительных чисел и требуется подсчитать в этой последовательности количество чисел M для случая $Xi \geq a$ и количество чисел B для случая $Xi < a$.

Очевидно, что i может принимать только целые значения 1, 2, ..., n. Отдельные операторы алгоритма будем обозначать числами 1, 2, 3, ... Они будут играть роль меток (номеров) операторов. Тогда решение задачи можно описать в виде следующего алгоритма:

1. B : = 0, M : = 0
2. i : = 1 1
3. Если $Xi < a$, то перейти к 4, иначе к 6
4. Добавить к B единицу (т.е. B : = B + 1)
5. Перейти к 7
6. M : = M + 1
7. Если $i<n$, то перейти к 8, иначе закончить вычисления (10)

8. i : = i +1
9. Перейти к 3
10. Конец

Знак «: =» понимается как символ операции присваивания величине, стоящей сле ва от него, значения величины, стоящей справа.

Эта последовательность операций и представляет собой запись алгоритма для че ловека, знающего алгебру и правила выполнения арифметических действий. Однако ней присутствуют «необычные» операторы 4, 6 и 8, которые бессмысленны для нема шинной математики, поскольку в ней такие равенства, как B = B+1 и другие, аналогич ные ему, невозможны. Суть такой записи в том, что старое значение B увеличивается н единицу и новое значение присваивается снова величине с именем B. Это равносильн операции, когда результат заносится в заданную клетку таблицы, а вычисления прово дятся на черновике. После вычисления нового результата на черновике старый резуль тат в этой клетке таблицы стирается и на его месте записывается новый и т.д. Такая за пись порождена удобством для отображения процессов вычислений на машине, где роль таблицы выполняет память машины. В этом случае после вычислений машина на правляет новый результат на заранее выбранный адрес памяти, что приводит к автома тическому стиранию старого результата и запоминанию нового. Воспользуемся данным алгоритмом для иллюстрации различных общепринятых средств его записи. В принятой нами терминологии операторы 1, 2, 4, 6, 8 отнесем к арифметическим (группа A), а 3, 5, 7, 9 – к логическим (группа P). При необходимости подчеркнуть принадлежность оператора одной из этих групп будем записывать A1, A2, A4, A6, A8 и соответственно P3, P5, P7, P9.

Мы остановимся более детально на операторной форме представления алгорит мов и их схем, так как предварительная подготовка задачи для программирования край не редко обходится без укрупненной разбивки на операторы и описания связей между ними на языке схем. По мере усложнения задач и характера вычислительного процесса, увеличения количества операций, необходимых для их решения, трудности алгоритми зации и программирования резко возрастают. Естественно, что возник вопрос о расчле нении вычислительного процесса на такие простые части, чтобы каждую из них про граммировать отдельно, а затем, соединив частные программы, получить всю програм му в целом.

Сложность операторов определяется характером решаемой задачи. Чтобы после объединения операторов получилось описание алгоритмического процесса, необходимо каким-то образом описать взаимосвязь операторов. Обозначим все операторы буквами с индексами.

Для удобства записи логических схем алгоритмов операторы обычно располагают в одну строку и руководствуются следующими правилами:

1. Порядковый номер оператора в данной схеме изображается нижним индексом оператора. Нумерация операторов сквозная.

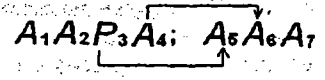
2. Если оператор зависит от параметра, то этот параметр изображается верхним индексом оператора (например, A3, P5 и т.п.).

3. Если знаки двух операторов располагаются в схеме рядом, то оператор, стоящий в схеме слева, передает управление оператору, записанному справа.

4. Если между двумя записанными рядом знаками операторов стоит точка с запя той, то от оператора, записанного слева, нет передачи управления оператору, записан ному справа.

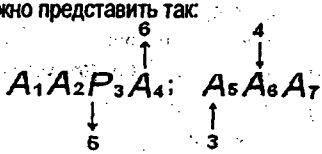
5. Передача управления оператору, записанному не рядом справа, обозначается елкой.

Например, операторная схема может выглядеть так:



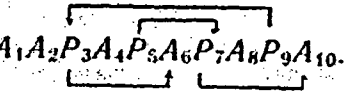
Для удобства записи горизонтальную часть стрелки можно опускать. При этом на-ло стрелки отмечается малой вертикальной стрелкой, направленной от оператора и абженной номером оператора, которому передается управление с указанием номера, редающего управление оператора.

Тогда эту же схему можно представить так:



При начертании схем программ обычно принято стрелки от логических операторов ображать над строкой, если логическое условие истинно, и под строкой в противном учае.

В соответствии с указанными правилами приведенный в этом параграфе алгоритм операторной форме можно записать так:



Во многих случаях бывает удобно группу элементарных операторов обозначать дной буквой. В этом случае придерживаются правила, что управление извне (от операоров, не принадлежащих данной группе) может получать лишь один элементарный ператор группы. Такую группу элементарных операторов называют обобщенным опеатором.

С помощью операторной схемы алгоритма составляется схема решения задачи, оторая дополняется операторами, свойственными машинному вычислительному проессу. Для каждого оператора в порядке его записи затем составляется своя частная рограмма. Совокупность этих частных программ и дает программу решения задачи. Использование операторной схемы алгоритма при программировании рассчитано на олучение следующих преимуществ:

1) облегчается работа по составлению программы;

2) уменьшается количество ошибок при программировании;

3) наличие операторной схемы алгоритма облегчает проверку готовых программ;

4) появляется возможность возобновлять работу над программой после длительного перерыва или поручать продолжение начатой работы другому лицу.

Среди всех задач выделяют расчетные, алгоритмы которых легко описываются с помощью математических формул, и логические задачи.

К логическим задачам относятся преимущественно многочисленные задачи управления и планирования. При их решении на ЭВМ схемы алгоритмов приобретают важное значение, так как они являются пока почти единственным формальным аппаратом, отображающим динамику сложных процессов.

Схема алгоритма, с одной стороны, служит как формальный язык для зап держания и логических связей задачи, а с другой – она сравнительно легко пре ется в программу для ЭВМ.

Схемы алгоритмов – функционально ориентированные графические изобра с помощью которых, используя текст и специальные символы, описывают посл тельность шагов процесса во времени, связи рассматриваемой системы с внешне дой и ветвление процесса. Краткое текстовое или формальное описание шагов дится внутри символов. Сведения о входных и выходных данных некоторых шаго цесса могут даваться в закодированной форме. Схемы алгоритмов обладают те имуществом, что для их понимания не требуется специальных знаний.

Для удобства чтения схем алгоритмов в ряде стран разработаны стандартны фические обозначения для наиболее часто употребляемых и специфических операт

Поясним понятие схемы алгоритма. Пусть требуется вычислить функцию

$$y = \begin{cases} x^2, & \text{если } x < 0 \\ x^2 + \sqrt{x}, & \text{если } x \geq 0 \end{cases}$$

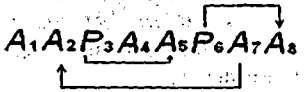
для всех дискретных значений x_i с шагом 0,1 из сегмента [-7,5].

Так как y представлен через аналитические выражения x^2 и $(x^2 + \sqrt{x})$, по котс могут быть произведены вычисления в результате подстановки в них значений x_i, т горитм для вычисления очевиден, остается построить схему алгоритма.

Из условий задачи вытекает необходимость выполнения следующих операций.

1. Ввести исходное значение $x = Xo = -7$. ПКУ2.
2. $M := x^2$. ПКУ3.
3. Проверить $x>=0$. Если да, то ПКУ4; если нет, то ПКУ5.
4. $y=(M+\sqrt{x})$. ПКУ6.
5. $y := M$. ПКУ6.
6. Проверить $x = 5$. Если да, то ПКУ8; если нет, то ПКУ7.
7. $x := x + 0,1$. ПКУ2.
8. Конец

Обозначим эти восемь операций соответственно A_1, A_2, P_3, A_4, A_5, P_6, A_7, A_8 (и еще короче: 1, 2, ..., 8). Если расположить эти символы в строчку и с помощью стре указать связи, то получим операторную схему алгоритма:



Если теперь вместо символов операторов вычертить прямоугольники, в них нап сать содержание действий, выполняемых данными операторами, и показать связям последовательность этих действий, то получим схему алгоритма (рис. 1.2.1).

Таким образом, схема алгоритма есть графическое изображение последовател ности операций, согласно которым получают решение задачи. Часто при разработк блок-схем алгоритмов предварительно проводят синтез граф-схем алгоритмов.

Основные свойства граф-схем алгоритмов следующие:

1. Граф-схема состоит из конечного числа точек, называемых вершинами (узлами и соединяющих их стрелок.

2. В граф-схеме имеются два особых узла: входной, в который не входит ни одн стрелка, и выходной, из которого не выходит ни одна стрелка.

3. Все узлы граф-схем отмечены своей логической переменной или арифметическим оператором.

4. Из каждого узла с символом *P* отходят, как правило, две стрелки, а с символом *A* – только одна стрелка

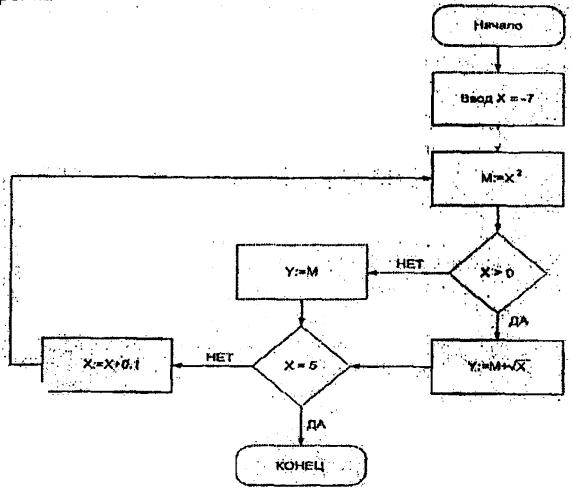


Рисунок 1.2.1 – Исходная схема алгоритма

Если в граф-схеме (рис. 1.2.2) логические переменные и арифметические операторы заменить соответствующими описаниями, то граф-схема превратится в схему алгоритма.

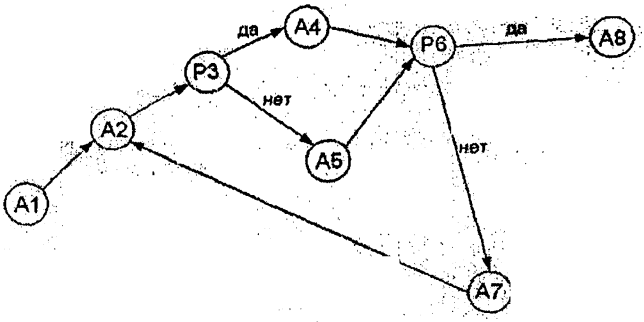


Рисунок 1.2.2 – Граф-схема алгоритма

1.3. Выбор численных методов реализации алгоритмов

Массовое использование цифровых вычислительных машин в решении ряда прикладных задач привело к бурному развитию теории численных методов вычислений. Численный метод – это метод приближенного или точного решения математических задач, основанный на построении конечной последовательности действий над конечным множеством чисел. Дискретная форма представления информации в ЭВМ потребовала

и создания адекватных методов ее обработки. Если по своей природе задача носит д
кретный характер и имеются соответствующие модели вычислений, то в этом слу
метод может практически сразу использоваться для машинной реализации, наприм
метод нахождения наибольшего общего делителя двух целых положительных чис
Проблемы могут лишь возникнуть в связи с задачей представления в памяти оче
больших чисел или же из-за ограничений на время ожидания ответа. Дополнительн
затрат потребовала переориентация методов, использующих непрерывные матема
ческие модели вычислений. Непрерывные модели вычислений стали заменяться ди
кретными. Например, вычисление интеграла методом подстановок свелось к суммир
ванию. Если раньше задача взятия одномерного интеграла на отрезке решалась д
произвольных границ отрезка, то теперь она стала решаться для фиксированных гран
отрезка. В первом случае точность результата определялась на базе погрешностей в
числений по итоговой формуле, а во втором случае она еще стала зависеть и от колич
ства точек деления отрезка, определяющих число элементарных трапеций при сумм
ровании. Таким образом, появилась специфическая задача автоматического выбо
числа точек деления отрезка, чтобы избежать больших погрешностей метода.

Эти проблемы породили задачу анализа устойчивости вычислений по дискретны
алгоритмам.

Во многих вычислительных алгоритмах, разработанных до появления ЭВМ, иссл
довались только вопросы получения погрешностей из-за аппроксимации непрерывны
методов (моделей) вычислений дискретными операциями. Погрешности представлени
данных и накопления систематических ошибок при большом числе простейших опер
ций над приближенными числами не изучались.

Такого рода трудности стали обнаруживаться, когда теоретически сходящиес
процессы в некоторых случаях на ЭВМ не дали достоверных результатов. Поэтом
применяя ЭВМ для решения математических задач, необходимо учитывать ее особен
ности как вычислительного инструмента (в первую очередь это касается персональны
ЭВМ и программируемых микрокалькуляторов с небольшой разрядностью для пред
ставления чисел):

1) количество цифр в изображении чисел, над которыми производятся действия,
погрешности представления их;

2) большую скорость операций над числами, хранящимися в оперативной памяти;

3) сравнительно малую скорость ввода исходных данных и программ, а также вы
вода результатов;

4) сравнительно малую скорость обмена числами между оперативной и внешней
памятью;

5) ограниченную емкость оперативной памяти при практически безграничной емко
сти накопителя;

6) возможность случайных сбоев в работе машин.

Выбирая численный метод, нельзя забывать о необходимости проверки правиль
ности вычислений при решении задачи на машине. Способ контроля путем повторных
расчетов не всегда эффективен. Более экономным и эффективным может оказаться
контроль путем проверки каких-либо заранее известных соотношений между вычисляе
мыми величинами (например, $\sin^2 x + \cos^2 x = 1$).

Некоторые численные методы практически не требуют контроля правильности вы
числений. К этому классу, в частности, относятся сходящиеся итерационные методы. Их

тоинством является то обстоятельство, что получение ошибочного результата при ой из итераций не приводит к ухудшению окончательного результата вычислений, а ь увеличивается количество итераций, которые должна выполнять машина. При бы-ой сходимости итерационного процесса затраты машинного времени будут сравни-ьно небольшими. Приведем пример такого итерационного процесса. Пусть необхо-мо вычислить $y = \sqrt{x}$ с точностью до 0,01.

На ЭВМ для извлечения квадратного корня из x многократно применяется следую-я процедура вычисления:

$$y_{i+1}=0,5\ (y_i+x: y_i)$$

Всего (i+1) раз, пока не выполнится неравенство $|y_{i+1} - y_i| <= e$, тогда y полагают рав-м y_{i+1}. За y_0 обычно можно выбрать x.

В нашем примере $y_{i+1} = 0,5(y_i + 2:y_i)$, $|y_{i+1} - y_i| <0,01$ и $y_0 = x = 2$.

Применим эти соотношения для иллюстрации процесса вычислений без ошибок и с ной ошибкой.

Процесс вычисления y без сбоя с точностью до 0,01.

1) $y_1 = 0,5\ (2 + 2:2) \approx 1,5$, $|1,5 - 2| > 0,01$

2) $y_2 = 0,5\ (1,5 + 2:1,5)=1,67$, $|1,67 - 1,5| > 0,01$.

3) $y_3 = 0,5\ (1,67 + 2:1,67) \approx 1,43$, $|1,43 - 1,67| >0,01$.

4) $y_4 = 0,5\ (1,43 + 2:1,43) \approx 1,41$, $|1,41 - 1,43| > 0,01$.

5) $y_5 = 0,5\ (1,41 + 2:1,41) = 1,41$, $|1,41 - 1,41| < 0,01$.

Ответ: $y \approx 1,41$.

Процесс вычисления y с одним сбоем (пусть на шаге 3 вместо 1,43 было получено =0,43, и с этого шага продолжим вычисления):

4) $y_4 = 0,5\ (0,43 + 2:0,43) = 2,54$, $|0,43 - 2,54| > 0,01$.

5) $y_5 = 0,5\ (2,54 + 2:2,54) = 1,66$, $|2,54 - 1,66| > 0,01$.

6) $y_6 = 0,5\ (1,66 + 2:1,66) \approx 1,44$, $|1,44 - 1,66| > 0,01$.

7) $y_7 = 0,5\ (1,44 + 2:1,44) \approx 1,40$, $|1,40 - 1,44| > 0,01$.

8) $y_8 = 0,5\ (1,40 + 2:1,40) \approx 1,41$, $|1,41 - 1,40| = 0,01$.

Ответ: $y = 1,41$.

Выбирая численный метод, следует учитывать особенности подготовки задачи для ешения ее на машине:

1) необходимость сведения ее к выполнению последовательности арифметических ействий (практически можно использовать и другие элементарные действия, для кото-ых уже составлены стандартные программы);

2) трудоемкость процесса программирования;

3) необходимость отладки программы на машине.

По условиям задачи всегда требуется знать и точность решения. Точность реше-ия обычно задают значением максимально допустимой погрешности. В связи с этим еобходимо выбирать, а иногда и специально разрабатывать соответствующий числен-ый метод.

Во всякой ЭВМ числа, представленные в режиме с плавающей запятой, обычно имеют небольшую погрешность. ЭВМ оперирует с довольно большим количеством цифр, и в результате вычислений накапливается арифметическая погрешность, которая скла-дывается с погрешностью метода. В этом случае бывает необходимо оценивать общую погрешность, которая не должна превышать максимально допустимую погрешность.

В целях экономии времени и количества ячеек в памяти полезно выбирать а ритмы с небольшой связностью.

Связностью алгоритма называется количество данных, которые нужно накапливат запоминающих устройствах машины для перехода от одного этапа вычислений к друго

Если связность алгоритма велика, то возникает необходимость переноса промеж точных результатов из оперативной памяти во внешнюю. Последнее обстоятельст влечет за собой резкое увеличение времени, расходуемого на решение задачи. Иног удается путем незначительных изменений значительно уменьшить связность алгорит

Рассмотрим пример.

Пусть требуется вычислить многочлен:

$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_i x + a_0$.

Если сначала вычислить все одночлены полинома, то для перехода от этапа ч числения величин $y_i = a_i x_i$ к этапу вычисления y требуется запомнить числа y_0, y_1 ..., что приведет к связности алгоритма, равной (n+1).

Но эти вычисления можно упростить по следующим формулам:

$T_i = T_{i-1} x$, $y_i = y_{i-1} + a_i T_i$, $T_0 = 1$, $y_0 = a_0$.

После выполнения всех вычислений получим результат $y = y_{n+1}$.

В этой схеме вычислений по сравнению с предыдущей требуется помнить при п реходе к следующему этапу только 2 числа, т.е. связность равна 2.

Почти все решения задач связаны с вычислением некоторых функций и, в часта сти, функций одной переменной f (x).

Функция $y = f(x)$ может задаваться многими способами, но машинные вычислени функций опираются на стандартные подпрограммы и следующие способы их предста ления.

1.Функция f (x) задается в явном виде с помощью аналитической формулы, соде жащей конечное число основных операций (арифметических, а также операций типа x^n, $\ln x$, $\sin x$, e^x и т.п.).

2.Функция задается конечной системой многочленов.

3.Функция задается таблицами, графиками.

При численном решении задач значение функции вычисляется всегда с заданно точностью. Поэтому многие способы задания функций, теоретически требующие выпол нения бесконечного числа операций, на практике сводятся к способам с конечным чис лом операций. Например, задание функции в виде суммы членов сходящегося степен ного ряда можно отнести к первому случаю, т. к. в памяти машины хранится конечное число первых членов ряда и отбрасываются остальные. Точнее, программа вычисления функции с заданной точностью реализована на базе необходимого числа членов ряда.

Выбор способа задания функции и соответственно вычисления ее значений бази руются на двух основных факторах:

1)экономия количества ячеек памяти, необходимого для осуществления выбирае мого способа вычислений;

2)экономия машинного времени, расходуемого на вычисление всех значений функций.

Если имеется ряд способов, не требующих использования внешней памяти, то прибегают к тому из них, который экономнее относительно временных затрат. Лишь в случае небольших временных затрат можно применять легко программируемые алго ритмы, требующие использования внешней памяти. Пусть функция задана явно с помо

ью формулы, содержащей конечное число операций и описанной различными элемен-
ірными функциями (|x|, x^n, lnx, sinx и т.п.). Программы для вычисления элементарных
ункций имеют все современные ЭВМ. Поэтому на основе стандартных программ мож-
) получить программы для вычисления функции. Если оказывается, что вычисления
ительны и громоздки, то часто прибегают к описанию функции многочленами, табли-
ами и т.п.

Возможна и обратная ситуация, когда большие таблицы заменяются несколькими
остыми функциями, чтобы ускорить время выборки данных и сэкономить количество
еек памяти.

Интервал *(a, b)* изменения независимой переменной *x* разбивают на несколько по-
интервалов, на каждом из которых выбирают нужный способ задания функции.

Рассмотрим реализацию способа задания функции на одном из подинтервалов пу-
ем подбора соответствующего многочлена.

Степень многочлена подбирается так, чтобы выполнялись условия:

1) степень должна быть возможно более низкой;

2) значение многочлена должно отличаться от соответствующего значения функ-
ии на величину, меньшую, чем некоторое малое число *E*, характеризующее допусти-
ую погрешность получаемых значений функции.

Если *Pi (x)* – многочлен, то на соответствующем подинтервале должно выполнять-
я неравенство $|f(x) - Pi(x)| < E$.

В этом случае в память вместо большой таблицы значений функции или большой
ю объему программы для вычисления ее значений вводят небольшую таблицу коэф-
рициентов многочленов и программу их вычисления. По значению *x* выбирается соот-
етствующий многочлен и по стандартным программам вычисляется его значение.

1.4. Ассоциативные исчисления и нормальный алгоритм Маркова

Изученные ранее свойства алгоритмов эмпирические. Выводы о них сделаны на
основании опыта. Но сами по себе свойства не могут являться основой для точной ма-
тематической формулировки понятия алгоритма. В связи с этим перейдем к более стро-
гому описанию алгоритмов.

Удобными для этой цели оказались понятия и средства, накопленные в процессах
преобразования различных слов как цепочек любых символов, что фактически исполь-
зовалось нами при решении задачи поиска пути в лабиринте. На абстрактном уровне
вычислительная машина всегда выполняет операции по переработке исходного текста
(набор букв и чисел, задающих условие задачи) в некоторый заключительный текст
(описание результата решения задачи). В машине буквы и цифры задаются в виде по-
следовательностей нулей и единиц, поэтому решение задачи фактически сводится к пе-
реработке исходной конечной последовательности из нулей и единиц (условие задачи) в
заключительную последовательность из нулей и единиц (результат). Понятие цепочки
символов (слова) определяется через алфавит.

Назовем алфавитом любую конечную систему различных символов. Символы, со-
ставляющие алфавит, будем называть буквами.

Например, {a, 3, ?, *} – алфавит, a, 3, ?, * – буквы.

Любая конечная последовательность букв некоторого алфавита называется сло-
вом в этом алфавите. Например, в алфавите *A={a, b, c)* словами будут последователь-
ности *ab, ac, abac, bbbb* и т.п.

Рассмотрим два слова H и M в некотором алфавите *A*. Если *H* является частью *M*, то говорят, что *H* входит в *M*. Например, *H=ac; M=bbacab.*

Опишем процесс преобразования слов. Зададим в некотором алфавите конечную систему подстановок *H, M, ..., C, T*, где *H, M, ..., T* – слова этого алфавита.

Любую подстановку вида *H – M* можно применить к некоторому слову *K* этого алфавита следующим способом: если в слове имеется одно или несколько вхождений слова *H*, то любое из этих вхождений может быть заменено словом *M*, и, наоборот, если имеется вхождение *M*, то его можно заменить словом *H*.

Например, *H=ab, M = bcb, K=abcbcbab.* Заменив в слове *K* слово *H* на *M*, можно получить такие слова: *bcbcbcbab* или *abcbcbbcb*, и, наоборот, заменив *M* на *H*, получим *aabcbab* или *abcabab*.

Подстановка *ab – bcb* недопустима к слову *bacb*, т. к. ни *ab*, ни *bcb* не входит в это слово. К полученным с помощью допустимых подстановок словам можно снова применить допустимые подстановки и т.д.

Совокупность всех слов в данном алфавите вместе с системой допустимых подстановок называется ассоциативным исчислением. Чтобы задать ассоциативное исчисление, достаточно задать алфавит и систему подстановок.

Слова P_1 и P_2 в некотором ассоциативном исчислении называются смежными, если одно из них может быть преобразовано в другое однократным применением допустимой подстановки.

Последовательность слов *P, P_1, P_2, M* называется дедуктивной цепочкой, ведущей от слова *P к M*, если каждые из двух рядом стоящих слов этой цепочки смежные.

Слова *P* и *M* называются эквивалентными, если существует дедуктивная цепочка от *P* к *M* и обратно.

Пример.

{a, b, c, d, e} – алфавит,

ac – ca; abac – abacc
ad – da; eca – ae
bc – cb; eda – be
bd – db; edb – be

} – подстановки

Слова *abcde* и *acbde* – смежные (подстановка *bc – cb).* Слова *abcde* и *cadbe* эквивалентны.

Ассоциативному исчислению можно поставить в соответствие бесконечный лабиринт, приняв каждое слово данного алфавита за площадку и соединив смежные площадки (слова) ребрами. Так как число слов бесконечно, то и лабиринт бесконечен.

Если слова *P и M* эквивалентны, то в построенном лабиринте это означает, что площадка, соответствующая *M*, достижима с площадки *P*.

Иногда рассматривается специальный вид ассоциативного исчисления, которое задается алфавитом и системой ориентированных подстановок типа *H->M.* Стрелка означает, что разрешается производить подстановку лишь слева направо. Это исчисление соответствует бесконечному лабиринту, в котором разрешается движение только в одном направлении

Для каждого ассоциативного исчисления своя специальная проблема слов: для любых двух слов требуется узнать, эквивалентны они или нет.

Это та же проблема достижимости, которая была рассмотрена в примере с лабиринтом, но лабиринт теперь стал бесконечным. Поэтому метод поиска, пригодный для

чного лабиринта, становится непригодным из-за невозможности в конечное время ледовать лабиринт. Многие конструкторские и другие задачи сводятся к такой же леме (конструкция – слово).

Зная алгоритм поиска в конечном лабиринте, его можно применить лишь к ограни-ой проблеме слов, когда требуется установить, можно ли одно из заданных слов бразовать в другое применением допустимых подстановок не более k раз.

В этом случае проблему можно решать так: построить все смежные слова с исход-, затем для каждого из полученных слов снова построить все смежные слова и т.д., о k раз. Отсюда следует, что логическая задача о поиске пути в лабиринте может ь сформулирована на языке ассоциативного исчисления.

Вообще любой процесс вывода формул, математические выкладки и преобразова- также являются дедуктивными цепочками в выбранном подходящим образом ассо-тивном исчислении. Алгоритмические процессы также могут трактоваться как ассо-тивное исчисление.

Естественно предположить, что построение ассоциативных исчислений может ть универсальным методом для задания детерминированного процесса «переработ- исходных данных, т.е. для задания алгоритма. Для этой цели необходимо уточнить нятие алгоритма.

Алгоритмом в алфавите A называется всякое общепонятное точное предписание, ределяющее потенциально осуществимый процесс над словами из A, допускающий бое слово в качестве исходного и последовательно определяющий новые слова в м алфавите.

Будем считать, что алгоритм в алфавите A задается в виде некоторой системы до-тимых подстановок, дополненной общепонятным точным предписанием о том, в каком рядке и как нужно применять допустимые подстановки и когда наступает остановка.

Приведем пример.

Алфавит	Система подстановок B
$A = \{a, b, c\}$	$cb - cc$
	$cca - ab$
	$ab - bca$

Условие дополняется указанием о способе применения подстановок. Для произ-льного слова P разыскать первую подстановку, левая часть которой входит в P. Если кой подстановки нет, то процесс прекратить. В противном случае берется первая из йденных подстановок и делается замена ее правой части вместо первого вхождения левой части в слово P. Затем полученное слово P_1 играет роль P и т.д.

Итак, схема подстановок вместе с указанием, как ими пользоваться, определяет лгоритм в алфавите A.

Рассмотрим применение системы B на конкретном примере для слов babaac и cacabc:

babaac→bbcaaac→остановка

bcacabc→bcacbcac→bcacccac→bcacabc

Процесс бесконечен, т.е. остановка не наступит.

Для того чтобы формально уточнить понятие алгоритма, советский ученый А.А. Мар-ов ввел понятие нормального алгоритма. Опишем его.

Задается алфавит и схема подстановок B. Для произвольного слова P просматри-аются формулы подстановок в том порядке, в каком они заданы в схеме B, и разыски-

вается формула с левой частью, входящей в P. Если такой формулы нет, то процес рывается. В противном случае берется первая из таких формул и делается подстан ее правой части вместо первого вхождения ее левой части в P, что дает слово P_1, торым поступают аналогично. Обрывается процесс в двух случаях: во-первых, когда лучим такое слово P_i, что ни одна из левых частей подстановок не будет входить в во-вторых, когда при получении P_i нам придется применять последнюю формулу.

Различные нормальные алгоритмы отличаются друг от друга лишь алфавитам системами допустимых подстановок. Приведем пример нормального алгоритма, оп вающего процесс сложения чисел:

Алфавит	Система подстановок в
A{1, +}	1+ →+1
	+1→ 1
	1→ 1

Слово P: 11 + 11 + 111.

Переработаем слово P с помощью алгоритма Маркова, отмечая знаком (-----) бираемую подстановку.

$P = 11 + 11 + 111$ $\quad P_5 = + 1 + 111\ 111$

$P_1 = 1 + 111 + 111$ $\quad P_6 = + + 1111111$

$P_{2=} + 1111 + 111$ $\quad P_{7=} + 1111111$

$P_3 = + 111 + 1111$ $\quad P_8 = 1111111$

$P_4 = + 11 + 11111$ $\quad P_9 = 1111111$

Мы видим, что этот алгоритм суммирует количество единиц. Следует обрат особое внимание на элементарность и однозначность шагов, выполняемых при реа зации алгоритма. Естественно, что представления любого алгоритма в такой фор достаточно, чтобы поручить его реализацию некоторому автомату. Нормальный ал ритм Маркова можно рассматривать как стандартную форму для задания любого ал ритма. Такая форма на практике используется лишь в теоретических построениях. процессе построения теории численных алгоритмов выяснилось, что любой логическ алгоритм можно простыми методами свести к численному. Таким образом, теория чи ленных алгоритмов (она тождественна понятию «теория вычислимых функций») ста универсальным аппаратом для исследования алгоритмических проблем.

Теорема. Любой алгоритм можно свести к вычислению значений некоторой цел численной функции при целочисленных значениях аргументов.

Докажем теорему.

Включим все условия задачи, доступные для переработки данным алгоритмом A пронумерованную неотрицательными целыми числами последовательность

$A_0, A_1, A_2, ..., A_n, ...$.

Аналогично записи возможных решений также включим в пронумерованную посл довательность $B_0, B_1, B_2, ..., B_m, ...$.

После проведения нумерации очевидно, что любой алгоритм, перерабатывающи запись условий A_n в запись решений B_m, можно свести к вычислению значений некото

ии числовои функции $m = \phi(n)$, так как после введения нумерации можно иметь дело ишь с соответствующими номерами записей условий и решений, а не с самими запи-ми. Теперь можно говорить об алгоритме, перерабатывающем номер записи условия номер записи решения, который будет численным алгоритмом.

Очевидно, что при наличии алгоритма, решающего исходную задачу, имеется и ал-ритм вычисления значений соответствующей функции.

Действительно, чтобы найти значение

$\phi(n)$ при $n = n^*$ (звездочка означает конкретное значение n), можно по n^* восстано-ить запись условий задачи, а затем по имеющемуся алгоритму найти запись решения и) ней определить номер $m = m^*$, т.е. $\phi(n^*) = m^*$.

Наоборот, *если* есть алгоритм вычисления функции ф(n), то имеется и алгоритм ешения исходной задачи, так как по записи условий задачи можно найти соответст-ующий ей номер n', затем вычислить $m^* = \phi(n^*)$ и по m^* определить запись решения.

Один из методов такой нумерации был предложен австрийским математиком Гё-елем. Любое целое неотрицательное число n можно представить в форме

$n = 2^{a1}*3^{a2}*5^{a3}*\ldots* P_{m-1}^{am}$, где $P_0 = 2$, $P_1=3$, ..., P_{m-1}, т.е. P_m – m-е простое число.

В силу теоремы о единственности разложения любого числа на простые множите-и следует, что каждому числу n однозначно соответствует набор $A=\{a1, a2, \ldots, am\}$, и, зоборот, каждому набору A однозначно соответствует число n.

Например, если n = 60, имеем $60 = 2^2*3^1*5^1$ т.е. $a_1 = 2$; $a_2=1$; $a_3=1$.

Если n = 98, то $98=2^1*3^0*5^0*7^2$, имеем $a_1 = 1$; $a_2= 0$; $a_3 = 0$; $a_4 = 2$.

С помощью этого способа (геделизации) можно нумеровать любые упорядоченные оследовательности, состоящие из m чисел.

Приведем несколько примеров:

1. Каждой паре чисел a_1 и a_2, для которой мы ищем ее наибольший общий дели-ель q, может быть поставлен в соответствие геделевский номер этой пары

$n ==2^{a1} *3^{a2}$. Алгоритм Евклида сведется к вычислению функции $q = \phi(n)$.

2. Пусть требуется перенумеровать все слова в некотором алфавите A. Это легко делать, поставив в соответствие каждой букве алфавита какое-либо число. Тогда каж-ому слову в алфавите A будет соответствовать последовательность чисел. Проводя атем обычным способом геделизацию, получим геделевский номер этой последова-ельности.

Таким образом, не только арифметические алгоритмы сводятся к вычислению зна-ений целочисленных функций. Любой нормальный алгоритм Маркова с помощью геде-изации может быть также сведен к вычислению значений целочисленных функций. По-тому алгоритм вычисления целочисленных функций можно считать универсальной ормой алгоритма. Естественно, что эти утверждения верны для случаев, когда всех словий задачи может быть бесконечно много, но множество это все же счетное.

1.5. Машина Поста и программирование на ней

Рассматривая различные подходы к строгому определению алгоритма, мы убеди-ись, что во всех случаях результат использования алгоритма не зависит от того, кто его рименяет. Более того, действия человека по выполнению строгих однозначных пред-исаний напоминают действия машины. Исходя из свойств алгоритмов, легко сформу-ировать требования к машине:

1) во-первых, она должна быть полностью детерминированной и действовать в со ответствии с заданной системой правил;

2) во-вторых, она должна допускать ввод начальных данных;

3) в-третьих, заданная система правил работы машины и класс решаемых зада должны быть согласованы так, чтобы всегда можно было прочесть результат работ машины.

Выдающийся американский математик Эмиль Пост одновременно с английски математиком А. Тьюрингом в 1936 г. предложил уточненную трактовку понятия алгори ма на основе своей машины, которую позднее стали называть машиной Поста (в отли чие от Тьюринга, он термин «машина» не использовал). Машина Поста менее популяр на, хотя она значительно проще машины Тьюринга. С ее помощью легче решать задач получения первых навыков по составлению программ для ЭВМ.

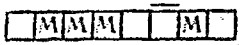
Абстрактная машина Поста состоит из бесконечной ленты, разделенной на равны секции, а также считывающей и записывающей головки.

В каждой секции ленты может быть либо ничего не записано (такая секция называ ется пустой), либо записана метка «М» (тогда секция называется отмеченной):

Информация о заполнении метками секций ленты и пустых секций характеризуе состояние ленты.

Состояние ленты может меняться в процессе работы машины.

Головка (знак «—» над секцией) может передвигаться вдоль ленты влево и вправо В неподвижном состоянии она находится над одной секцией ленты:



Говорят, что головка обозревает эту секцию. Информация о заполнении ленты местонахождении головки характеризует состояние машины Поста.

За единицу времени (такт) головка может сдвинуться на одну секцию влево, вправ или остаться на месте.

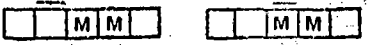
Работа машины Поста заключается в том, что головка передвигается вдоль ленты и печатает или стирает метки в соответствии с заданной программой (инструкцией), ко торая состоит из отдельных команд, а также распознает, имеется ли метка в секции.

Команда машины Поста имеет следующую структуру: *xBy*, где *x* – порядковый но мер команды, *B* характеризует действие, выполняемое головкой, а *y* указывает номе (адрес) следующей команды, которая подлежит выполнению. Машина Поста имее шесть команд, которые представлены ниже в таблице.

Таблица 1.5.1 – Система команд машины Поста

№ п/п	Вид команды	Условная запись	Описание команды
1	Движение вправо	*x->y*	Головка сдвигается на одну секцию вправо
2	Движение влево	*x<-y*	Головка сдвигается на одну секцию влево
3	Печатание метки	*x M y*	В секцию под головкой впечатывается метка М
4	Стирание метки	*x C y*	В секции под головкой стирается метка М
5	Условная передача управления	$X < \begin{matrix} y_1 \\ y_2 \end{matrix}$	При отсутствии метки в секции под головкой передается управление команде y_1, а в противном случае – команде y_2
6	Стоп	*X* СТОП *X*	Остановка машины

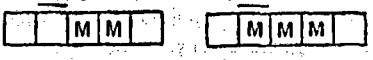
Поясним, как происходит реализация каждой команды с отображением соответствующих ситуаций на ленте. Состояние головки и ленты будем показывать схематически рядом с кодом команды до начала ее выполнения и правее после выполнения. Тогда таблица команд может быть прочитана так:

1. $x \rightarrow y$ 

(головка сдвигается на одну секцию вправо);

2. $x \leftarrow y$ 

(головка сдвигается на одну секцию влево);

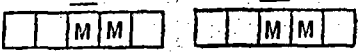
3. x М y 

(головка печатает метку в секцию под ней);

4. x С y 

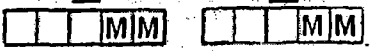
(головка стирает метку в секции под ней);

5. $x < {y_1 \atop y_2}$ y_1: 

y_2: 

Если головка находится над пустой секцией, то управление передается команде по адресу *y1* и головка остается над этой же секцией.

Если же головка находится над секцией с меткой, то управление передается команде по адресу y2 и головка остается над этой же секцией. Смысл этой команды сводится к выбору следующего адреса для продолжения выполнения программы.

6. x **стоп** x 

(головка остается на той же позиции, и машина останавливается).

Покажем реализацию некоторых типичных элементов программ для машины Поста с иллюстрациями.

Пусть задано исходное состояние головки () и требуется на этой пустой ленте написать две метки: одну в секцию под головкой и вторую справа от нее. Это можно сделать по следующей программе, где справа от команды показан результат ее реализации.

1 М 2 

2 → 3 

3 М 4 

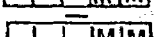
4 стоп 4 

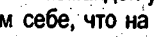
Пусть головка находится над левым концом начала сплошной последовательности из четырех меток () и требуется стереть две метки подряд слева направо. Реализация этого процесса будет следующей:

1 С 2 

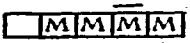
2 → 3 

3 С 4 

4 стоп 4 

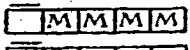
Покажем, как воспользоваться командой условного перехода для организации циклического процесса. Представим себе, что на ленте имеется запись из четырех меток подряд, и головка находится над самой крайней меткой справа (). Требуется перевести головку влево до первой пустой позиции.

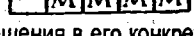
Если решать задачу без использования команды условного перехода, то программа будет следующей:

1 ← 2 

2 ← 3 

3 ← 4 

4 ← 5 

5 стоп 5 

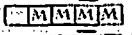
Недостаток этого решения в его конкретности, так как, зная количество меток, можно решить по данной программе только эту задачу. Используя же команду условного перехода, можно решать более общую задачу для любого количества меток: переместить головку до первой пустой секции. Программа будет иметь следующий вид:

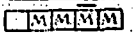
1 ← 2

2 < 3, 1

3 стоп 3

Ее уже не удается представить в виде однозначного отображения в картинках после каждой команды. Такую программу можно «показать» только в процессе ее выполнения для конкретного случая:

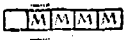
1 ← 2 

2 < 1 

(поскольку под головкой есть метка, то управление передается снова команде 1)

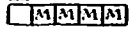
1 ← 2 

2 < 1 

1 ← 2 

2 < 1 

1 ← 2 

2 > 3 

(поскольку под головкой находится пустая секция, то управление передается команде 3)

3 стоп 3 

На этом примере мы видим, что команда условного перехода является одним из основных средств организации типичных циклических процессов, например, нахождения первой метки справа или слева от головки, расположенной над пустой секцией, нахож-

ния слева или справа от головки пустой секции, если она расположена над секцией с ткой и т.д. Этим средством мы будем широко пользоваться при решении различных дач на машине Поста, поскольку таким образом получают на ленте запись произволь- о количества меток. Программой машины Поста называется конечный непустой спи- ее команд, обладающий следующими свойствами:

1) на i-м месте в списке стоит команда с номером i;

2) номер *y* совпадает с номером некоторой (другой или той же) команды списка, *y* обязательно упоминается среди номеров команд с левой стороны от действия *B*.

Примеры.

1)1->2-

2M 3 – программа машины Поста 3 стоп 3 (в соответствии с определением);

2) 2->3

3M 1 – не является программой для 1 стоп 1 машины Поста, так как нарушено рвое условие;

3) 1->2 – не является программой для 2M 5 машины Поста, так как нарушено вто- е условие (5 отсутствует среди левых номеров программы).

3 стоп 3

Можно условиться, что в начальном состоянии головка всегда находится против рвой пустой секции левее последней левой метки на ленте, если не оговаривается обо, где она находится. *Таким образом, начальное состояние машины полностью оп-* делено состоянием ленты. Программа является той инструкцией, на основании кото- й работает машина.

Машина приводится в начальное состояние и приступает к выполнению програм- ы, начиная с первой команды. Каждая команда выполняется за один такт. После вы- лнения текущей команды выполняется команда, адрес которой указан индексом *y*, или оисходит остановка машины.

Причиной остановки машины может служить команда «стоп» или выполнение дей- вия, которое приводит к записи метки в секцию с уже имеющейся меткой или стиранию писи в пустой секции.

При выполнении программы могут встретиться три случая:

а) машина дойдет до невыполнимой команды, и произойдет ее безрезультатная становка;

б) машина при выполнении программы дойдет до команды «стоп», и произойдет езультативная остановка, при которой программа считается выполненной;

в) машина при выполнении программы не дойдет ни до одной из команд, указан- ых в а) и б), и никогда не остановится.

Приведем примеры различных программ, иллюстрирующих случаи а), б) и в) для дного и того же исходного состояния машины Поста:

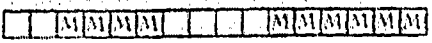
а) 1 → 2 б) 1 → 2 в) 1 → 2

3 M 4
4 стоп 4

3 стоп 3

3 ← 1
4 стоп 4

Остановимся на записи чисел на машине Поста и выполнении операций над ними. Число *к* представляется на машине Поста путем записи подряд *к+1* метки. Между двумя числами делается интервал как минимум из одной пустой секции на ленте. Например, на ленте запись чисел 3 и 5 будет выглядеть так:



На машине Поста можно выполнять разнообразные вычисления, но мы ограничимся лишь простейшими из них для формирования первичных навыков составления программ.

Основная математическая задача при работе человека на вычислительной машине в принципе одинакова для реальных и «абстрактных» машин и сводится к составлению для машины программы, приводящей к заданной цели.

Общими для реальной и абстрактной машин являются следующие моменты:

1) можно составлять различные программы, приводящие к одной и той же цели;

2) если расширяется класс исходных данных, то задача построения программы, как правило, усложняется;

3) при построении сложных программ можно использовать в качестве готовых строительных блоков составленные ранее программы для задач, носящих более частный характер;

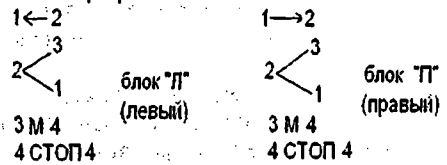
4) при составлении программы надо учитывать структуру исходных данных и их расположение в памяти машины;

5) минимизация программ по числу команд.

Приведем несколько программ для сложения чисел на машине Поста.

Составим программу для прибавления единицы к произвольному числу. Предположим, что на ленте записано только одно число, и головка машины Поста находится над одной из секций, в которой расположена метка, принадлежащая этому числу:

В случае поставленной задачи можно переместить головку влево (блок «Л») или вправо (блок «П») до первой пустой секции, а потом в нее записать метку, и задача будет решена. Составим такие программы:



В целях проверки правильности составленных программ для машины Поста можно, выбрав какую-то общую ситуацию для постановки задачи, промоделировать, начиная с первой команды, действие программы. Например, работу блока «Л» для случая, когда головка находится над второй меткой числа, можно записать так: 1Л2 1Л2 3М4. Такая запись позволяет легко смоделировать действия машины и проверить конечный результат, глядя на исходную позицию головки и запись числа. В самом деле, запись 1Л2 отражает, что головка подвинулась влево на одну секцию и произошла передача управления команде 2; по команде 2 читается метка М под головкой, и поэтому передается управление команде 1 (запись 21); по команде 1 снова выполняется движение влево и передается управление команде 2 (вторая запись 1Л2); после второго смещения влево головка уже окажется над пустой секцией, и поэтому команда 2 передаст управление

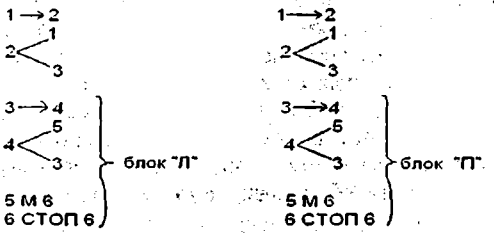
команде 3 (запись 23); по команде 3 произойдет запись метки в пустую секцию и передача управления команде 4 (запись 3М4), и наконец по команде 4 машина остановится. Такое детальное рассмотрение работы программы на достаточно общих (или в крайнем случае типичных) ситуациях называется процессом отладки программы, так как он позволяет обнаруживать ошибки как программирования, так и алгоритма, положенного в основу программы.

Представим себе, что в команде 2 вместо адреса 3 был бы случайно записан адрес 4, тогда, выполнив программу, машина остановилась бы, и к заданному числу метка не была бы дописана. Конечно, такая ошибка была бы обнаружена при моделировании работы программы на машине Поста.

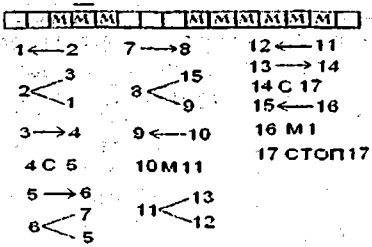
Несколько усложним задачу по прибавлению единицы к числу, расположив головку слева от числа на любое количество пустых секций. Тогда программа будет выглядеть так:

1→2
2<1
 <3
, а далее к ней можно подсоединить составленную ранее программу блок «Л» или блок «П», прибавив к каждой из отсылок и номеров команд число 2.

Дело в том, что написанный нами текст программы из двух команд позволяет решить задачу нахождения первой метки числа на ленте, а дальше мы можем использовать готовый результат, механически увеличив все номера в предыдущей программе на два:

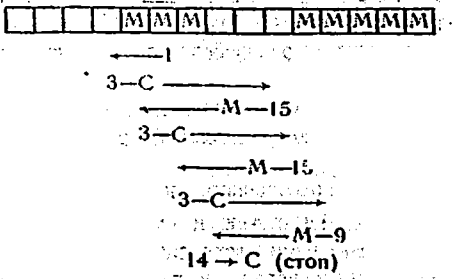


Этот пример имеет важное значение для иллюстрации принципов «автоматического» написания программ на ЭВМ из готовых подпрограмм. Суть процесса в том, что чисто механически, без анализа предыдущих программ, получается программа с новыми свойствами. Естественно, что в такой ситуации возможен перенос ошибок, допущенных другими авторами. Поэтому обычно используют программы, которые прошли многократную проверку на практике. Приведем программу для сложения целых неотрицательных чисел a и b на машине Поста, когда головка находится над числом a, а число b находится на расстоянии k секций от a справа.



Проведем отладку этой программы для сложения чисел 2 и 4, находящихся друг от друга на расстоянии 3 секций, и пусть головка находится над второй секцией числа 2. Рекомендованная нами выше схема записи текста будет выглядеть так: 1Л21Л23П4С5П6.

Очевидно, что в этом случае трудно ориентироваться в движении головки вправо, влево, стирании и написании меток. Поэтому более практична будет такая схема, когда движение головки в одну сторону отмечается стрелкой, включая записи или стирания (они помечаются буквой С или М под ячейкой ленты), пишется в отдельной строке слева направо (или справа налево) и в соответствии с координатой ее движения, а также пишется номер команды начала строки. Убедимся в этом, записывая строчки сверху вниз с переходом на новую строку после смены направления движения головки:



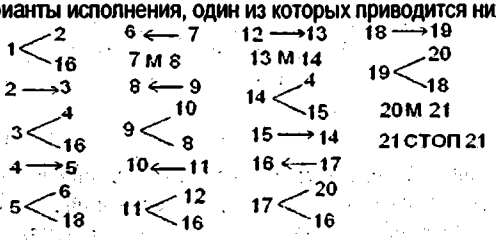
Начало стрелки содержит номер команды и находится под секцией, над которой находится головка.

Из этой отладочной схемы легко усматривается и алгоритм сложения двух любых чисел на машине Поста; первое число постепенно придвигается ко второму до их слияния, а потом стирается одна метка, и машина останавливается. Стирание метки обязательно, так как тогда бы результат был на единицу больше. Мы рекомендуем пользоваться описанной схемой для отладки программ на машине Поста.

Можно решить и еще более сложную задачу по прибавлению единицы к числу, когда головка находится над пустой секцией, но мы не знаем, слева или справа от нее расположено число.

Тогда, двигая головку поочередно вправо и влево и отмечая метками степень удаления от исходного положения головки, мы можем найти число, а затем уже известным нам путем прибавить к нему единицу.

Принцип решения задачи сводится к следующему: проверяется, находится ли головка над одной из меток числа и, если да, то такая программа у нас уже имеется, иначе проверяется, пуста ли секция справа от головки и следующая за ней. Если обе пусты, то делается возврат головки на один шаг и ставится метка, а затем такая же операция выполняется слева, и по отмеченной дорожке головка возвращается вправо, и снова проверяются две соседние секции и т.д. до тех пор, пока головка не натолкнется на число, и тогда можно будет применить программу по прибавлению единицы, когда метка находится над числом. Текст этой программы (без стирания вспомогательных меток) допускает различные варианты исполнения, один из которых приводится ниже:



Его рассмотрение представляет значительный интерес с точки зрения отладки этой более сложной, по сравнению с предыдущими программы. Чтобы убедиться в работоспособности программы, надо рассмотреть следующие случаи:

1) головка в исходном положении находится над числом;

2) головка находится в исходном положении рядом с числом слева (справа);

3) головка в исходном положении удалена от числа на две и более секции слева (справа).

Рассмотреть эти случаи предлагается читателю самостоятельно. Можно рассмотреть задачу составления программ умножения на фиксированное число, сдвига заданного числа на *k* разрядов (секций) и построения копий, т.е. такого же числа на другом участке ленты.

Проведем некоторый сравнительный анализ машины Поста и ЭВМ.

Как в ЭВМ, так и в машине Поста имеются неделимые носители информации (секция-разряд), которые могут быть заполненными или незаполненными. Машина Поста и ЭВМ имеют ограниченный набор элементарных действий-команд, каждая из которых выполняется за один шаг. Обе машины работают на основе программы.

Остановимся на некоторых отличиях машины Поста и ЭВМ. В машине Поста информация располагается линейно и читается подряд, а в ЭВМ можно читать информацию по адресу, что сокращает число шагов при работе ЭВМ. Для машины Поста не оговаривалось, как идет выполнение команд и перемещение головки. В частности, это может делать человек. В ЭВМ все команды выполняются автоматически, программа и исходные данные вводятся до начала ее работы. ЭВМ обладает еще одним важным свойством – возможностью изменить команды в процессе работы.

Машину Поста и ЭВМ можно трактовать как машины, обладающие запоминающими устройствами конечной емкости в каждый данный момент, но неограниченно растущими при необходимости, что позволяет реализовывать любые алгоритмы. Поэтому машину Поста можно рассматривать как упрощенную модель ЭВМ.

1.6. Конечные автоматы и роботы

Развитие вычислительной техники привело к дальнейшему усложнению задач, решаемых с ее помощью. На первом этапе эффективное использование ЭВМ проявилось в автоматизации умственной деятельности человека, но вместе с миниатюризацией элементов ЭВМ появилась возможность сочетания автоматизации производственных процессов с одновременной автоматизацией управления ими непосредственно на объекте. С этого момента начались интенсивные исследования и эксперименты, ставящие своей целью создание машин, обладающих не только мускулами, но и интеллектом. Такие машины, выполняющие некоторые функции человека, стали называться роботами.

Робот – способная действовать целенаправленно система управления и переработки информации, оборудованная датчиками восприятия информации о внешней среде и исполнительными механизмами.

От других систем переработки информации робот отличается антропоморфизмом, т.е. способностью реагировать на те же внешние сигналы, что и человек, и выполнять пространственные движения, подобно человеку.

Робот содержит три основных элемента: блок восприятия (датчики, воспринимающие информацию различной физической природы: свет, звук, магнитное поле и т.д.); блок исполнительного механизма (в зависимости от назначения робота и среды он мо-

жет иметь манипуляторы (руки), педипуляторы (ноги), колесные механизмы и т.п.); бл управления, осуществляющий целенаправленное поведение робота на основе сложно системы распознавания образов.

Многое из того, что человек делает с легкостью, для машин оказывается трудны Очень нелегко поручить роботу даже простую для человека задачу: сложить из ста дартных элементов детского конструктора, например, домик. Есть свои определенны проблемы и в создании механических частей роботов-манипуляторов. Создание кон рукции манипулятора является самостоятельной проблемой. Люди спокойно берут в р ки многие хрупкие предметы, например куриное яйцо, и не ломают их: человек

подсознательно оценивает малые деформации предмета и его прочность, а д робота – это сложная задача.

Ценность роботов для человека определяется тем, что они могут выполнять дейс вия в опасных и недоступных для человека средах или выполнять утомительную д него работу: исследование дна океанов и иных планет; тушение пожаров; операци внутри организма и т.п. Важно и то, что, достигнув однажды нужного качества выполн ния процесса, можно его многократно повторить.

За последние годы роботы нашли чрезвычайно широкое применение в промышле ности, позволив существенно повысить производительность труда, высвободить рабочи занятых монотонным и тяжелым трудом. Они стали настолько привычными участника производственного процесса, что по аналогии с широко распространенными в США н званиями для инженерного и административного персонала «белые воротнички», д рабочих – «синие воротнички», для роботов появился термин «стальные воротнички».

В настоящее время большинство находящихся в эксплуатации роботов и робото технических систем используется в машиностроении для сварки, механической обрабо ки, подачи и перемещения деталей, при измерениях и сборке.

Промышленным роботом называется программируемый многофункциональны манипулятор, предназначенный для перемещения материалов, деталей, инструмент или специализированных устройств по переменным программируемым траекториям выполнения других задач.

Используемые сейчас роботы в основном выполняют узкий круг задач. Это объяс няется как трудностями в создании механических элементов робота, так и его электрон ных и программных средств управления. В основном это относится к задачам по расп знаванию образов и принятию решений, выбору траекторий движения и т.п.

Ведь интеллектуальный робот должен обладать способностью распознавать об становку и автоматически вырабатывать решения о своих дальнейших действиях дл выполнения нужных технологических операций в неопределенной или меняющейся си туации. Еще более интеллектуальным должен быть робот для ведения домашнего хо зяйства и имитации сложных механических действий человека. Зарубежные специали сты ожидают его появления на рынке по относительно приемлемой цене на уровне стоимости легкового автомобиля.

Одной из трудных проблем реализации проекта таких роботов будет обеспечени контакта хозяина с роботом на уровне команд, подаваемых на естественном языке. Во обще эта задача является частной по отношению к проблеме общения человека с ЭВМ на ограниченном подмножестве естественного языка и будет нами рассмотрена ниже при обсуждении возможностей создания высокоинтеллектуальных систем.

Применительно к промышленным роботам мы рассмотрим лишь вопросы управле ния их рабочими органами. Абстрагируясь от конструктивных особенностей промыш

нных роботов, мы остановимся на математических аспектах управления ими и рассотрим вопросы программирования их действий на основе конечных автоматов. Работа ЭВМ, робота и всякого другого реального устройства рассматривается во времени. ремя обычно считается непрерывным и предполагается изменяющимся от настоящего будущему. Однако при рассмотрении ряда устройств удобно вводить воображаемое скретное время.

Пусть полуось времени (0, ∞) каким-то образом разбита на бесконечное число резков произвольной длины. Границы отрезков для простоты запишем как ряд целых отрицательных чисел 0, 1, 2, ..., i, ..., полагая, что воображаемое дискретное время инимает только эти целочисленные значения. Моменты времени 0, 1,2, ..., i назовем ктами.

Текущий такт, соответствующий настоящему моменту времени, обозначим через i, предшествующие ему и последующие такты – соответственно через (I - 1, i - 2, ...) и -1, i + 2, ...).

Будем рассматривать функционирование в дискретном времени динамических сис-м, описываемых координатами, каждая из которых изменяется на конечном множест-. Всякая динамическая система получает ряд внешних возмущений: звуковые, элек-ические, световые и т.п. сигналы. Будем считать, что их число также конечно и каждое возмущений задано на конечном множестве.

Динамические системы, удовлетворяющие вышеуказанным требованиям, назовем нечными динамическими системами.

Рассмотрим множества {C} = {C_1, C_2, ..., C_k} и {P} = {P_1 P_2, ..., P_r}, состоящие из сооттственно *k* и r символов. Набор C назовем состояниями динамической системы, а набор *P* – входами динамической системы.

Для полного описания динамической системы необходимо задать закон ее «движения», т.е. указать, как определяется состояние системы в каждый такт. Ограничимся ссмотрением одной из простейших систем.

Конечная динамическая система называется конечным автоматом, если состояние истемы в каждый такт однозначно определяется состоянием в предыдущий и входом в ссматриваемый такт.

В соответствии с определением конечного автомата состояние. C^i на любом такте i днозначно определяется состоянием C^{i-1} в предыдущий такт (i-1) и входом P^i в расматриваемый такт i, т.е. $C^i = F(C^{i-1}, P^i)$,где моменты времени, к которым относятся символы C и P, обозначены верхним индексом. Очевидно, эта формула определяет собой екуррентное соотношение, так как при известных P^1 и C^0 можно отыскать C^1, приняв *1*. Далее по C^1 и P^2 отыскать C^2 и т.д. конечных множества символов {C} и {P} и рекуррентное соотношение $C^i = P(C^{i-1}, P^i)$, связывающее переменные C и P^i.

Зафиксируем переменную P, положив ее равной некоторому $P_c \in \{P\}$. Конечный автомат, описываемый таким рекуррентным соотношением $C^i = F(C^{i-1}, P_c)$, назовем автономным.

Фиксируя разные символы из множества [P], получим r автономных автоматов. Задания *r автономных автоматов достаточно, чтобы описать поведение конечного автомата*. Автомат будем считать полностью заданным, если известны {P}, {C} и $F(C^{i-1}, P_c)$. Функцию F можно задавать различными способами. Приведем несколько наиболее потребительных из них.

Чаще всего задание различных функций F осуществляется с помощью таблиц.

Пусть множество входов {P} состоит из r символов. Тогда таблица функций пер ходов состояний строится следующим образом: вычерчивается прямоугольник, соде жащий (r+1)* (к+1) клеток. В первую клетку в верхнем левом углу прямоугольника со ветственно записываются в верхней и нижней частях ее наименования координат P1, C Вправо от нее по строке поклеточно вписываются все возможные значения P, а столбцу вниз – все возможные значения C. На пересечении столбца P и строки C вп сываются соответствующие значения F (C^{i-1} Pi) при $P^i = P$ и $C^{i-1} = C_m$. Для конкретизац характера таких функций будем задавать их таблицами, являющимися доступным и люстрируемым материалом.

В итоге получим таблицу, которую обычно называют основной таблицей конечно автомата (табл. 1.6.1).

Таблица 1.6.1 – Основная таблица конечного автомата

C^{i-1} \ P^i	P_1	P_2	...	P_r
C_1	C_K	C_1		C_5
C_2	C_3	C_2		C_4
...	...	...		...
C_R	C_3	C_3		C_6

Размещение данных в таблице таково, что пользование ею не вызывает затруднени

Пусть $P^i = P_2$, а $C^{i-1} = C_K$. Тогда искомое значение функции F (C_K, P_2) отыскивается п пересечении строки Ck и столбца P_2: $F=C_3$.

Можно заметить, что каждый из столбцов P_1, P_2, ..., Pr этой таблицы является о новной таблицей автономного автомата. Такие автоматы можно задавать с помощь направленных графов так, чтобы наблюдалось взаимно-однозначное соответствие ве шин графа и символов из алфавита C.

Переходу автономного автомата из состояния Ci в Cj будет соответствовать н правленная стрелка (рис. 1.6.1).

Такая совокупность кружков и стрелок (граф) полностью опишет функцию F при о ответствующем фиксированном P.

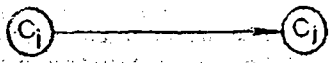


Рисунок 1.6.1 – Задание конечного автомата автономными автоматами

Покажем на примере небольшой таблицы, как ее описать на языке графов. Пус таблица 1.6.2. задает конечный автомат A с двумя входами и четырьмя состояниями.

Таблица 1.6.2 – Основная таблица автомата A

C_{i-1} \ P_i	P_1	P_2
C_1	C_2	C_3
C_2	C_1	C_2
C_3	C_4	C_3
C_4	C_2	C_1

Чтобы различить, для какого фиксированного входа рассматривается автономны автомат, над каждым из графов поставим наименования соответствующих входов.

Исходная таблица может быть представлена двумя графами A (P_1) и A (P_2), как эт сделано на рисунке 1.6.2.

Можно заметить, что от каждого из кружков на рисунке 1.6.3. отходит только по од-ой стрелке. Этот факт следует из определения детерминированного конечного автомата.

Для большей компактности таблицу 1.6.2. можно представить одним объединен-ым графом, который называется диаграммой состояний конечного автомата. Как видно з рисунка 1.6.3, на объединенном графе над каждой стрелкой записывается тот вход P, оторый вызывает переход из исходного состояния в то, к которому направлена стрелка. иаграмма состояний автомата весьма наглядна. Допустим, необходимо установить, ка-им будет внутреннее состояние автомата A на третьем такте при входной последова-ельности {P} = {P_1, P_1, P_2} и начальном состоянии $C^0 = C_3$. Решение задачи отыскивает-я простым обходом от начального состояния $C^0 = C_3$ по стрелке

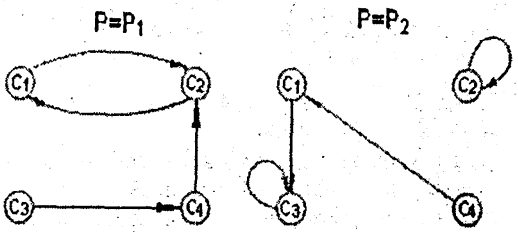


Рисунок 1.6.2 – Задание автомата диаграммой

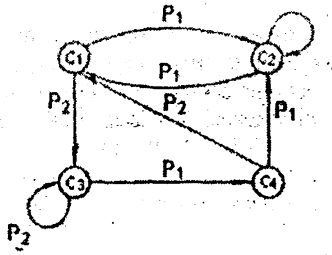


Рисунок 1.6.3 – Объединенный граф автомата

P_1 к следующему состоянию C_4, от него снова по стрелке P_1 к состоянию C_2, а от C_2 по трелке P_2 опять возвращаемся в состояние C_2, оно и будет искомым.

В ряде случаев желательно выполнять подобные операции более формальным пу-ем, чтобы использовать вычислительные средства для отыскивания нужного результа-а. Для этой цели удобно матричное задание конечного автомата.

Графу P_c автономного автомата можно поставить во взаимно однозначное соот-етствие квадратную матрицу, состоящую из нулей и единиц. Пронумеруем все возмож-ые состояния C цифрами от 1 до *к*. Составим матрицу $к * к$, такую, что в строке i на есте j будем записывать 1, если существует стрелка от i к j на графе A, а в противном лучае 0. Обозначим эту матрицу через $A(P_c) = (a_{ij}(P_c))$.

Очевидно, сумма $(a_{ij}(P_c)) = 1$, так как по определению автономного детерминиро-анного автомата от каждого C должна отходить одна и только одна стрелка, и, естест-енно, в строке *m* матрицы $A(P_c)$ найдется один элемент $a_{mi}(P_c) = 1$, а остальные будут авны нулю.

При матричном задании конечного автомата начальное состояние C^0 также задает-я матрицей $A(P^0)$, состоящей из одной строки, в которой записываются $(к-1)$ нулей, а а месте *j* с номером, соответствующим номеру C^0, – единица.

Пусть заданы входная последовательность $\{P\} = \{P^1, P^2, \ldots P^i\}$ и матрицей $A\ (P^\circ)$ начальное состояние C^0. Тогда состояние автомата на такте i устанавливается как результат вычисления произведения:

$$\prod_{i=0} A\ (P^i) = A_i\ (P^i),$$

где $A_i\ (P^i)$ - матрица, состоящая из одной строки с единицей на месте, соответствующем номеру искомого состояния.

Проиллюстрируем вышесказанное на примере автомата *A*. Графам P_1 и P_2 будут соответствовать матрицы

$$A\ (P_1) = \begin{pmatrix} 0100 \\ 1000 \\ 0001 \\ 0100 \end{pmatrix} \quad \text{и} \quad A\ (P_2) = \begin{pmatrix} 0010 \\ 0100 \\ 0010 \\ 1000 \end{pmatrix}$$

а начальному состоянию $C^\circ = C_3$ – матрица $A\ (P^\circ)$ =(0010). Тогда, используя формулу, получим искомый результат: $A_3\ (P^3)$ = (0100).

Система, состоящая из конечного автомата *A*, преобразующего символы алфавита {*P*} в символы алфавита *(C)* в соответствии с функцией $C^i = F(C^{i-1}, P^i)$, и преобразователям {*λ*}, который мгновенно и однозначно ставит в соответствие каждому символу *C* символ из некоторого алфавита {*λ*}, т.е. $\lambda^i = (C^i)$, называется конечным автоматом с выходным преобразователем.

В более общем случае можно считать, что преобразователь имеет два входа: *C* и *P*, тогда конечная динамическая система, получающаяся соединением конечного автомата и выходного преобразователя символов, на вход которого подводятся *C* и *P*, называется последовательностной машиной и описывается двумя соотношениями:

$$C^i = F\ (C^{i-1}, P^i) \text{ и } \lambda^i = (P^i, C^i).$$

Если вычисления выполнять последовательно и фиксировать промежуточные результаты на ленте бумаги, то полученная запись даст ленту последовательного изменения состояний автомата.

При описании функционирования реальных объектов (ЭВМ, роботов, станков и т.п.) часто удается применять конечные автоматы. Последнее связано с тем, что почти всякий процесс человек расчленяет на конечный ряд этапов и фиксирует *только* их, игнорируя переходные процессы. При алгоритмизации задач наблюдается аналогичная картина, когда отдельный алгоритм рассматривается как набор операторов, перерабатывающих входную информацию.

Покажем на примере, как готовится программа для управления роботом на основе модели управляющего устройства в виде конечного автомата. Если конечный автомат задан, то, зафиксировав его исходное состояние и подав входную последовательность символов, мы получим выходную последовательность символов. Каждый член входной последовательности в этом случае может рассматриваться как команда программы. Смена всей входной последовательности на другую аналогична смене программы работы автомата.

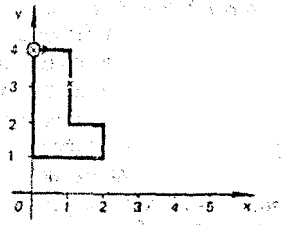
Пусть задан конечный автомат для управления роботом, предназначенным для обхода замкнутых контуров из отрезков прямых целочисленной длины, соединенных так, что они образуют внутренние углы замкнутого многоугольника в 90 или 270 градусов. Автомат служит управляющей системой для робота, который может двигаться за один такт на единичный шаг вперед, либо поворачиваться на 90 градусов по часовой стрелке,

р захватывать деталь, либо отпускать деталь, либо стоять. Иными словами, по ко-манде «двигаться вперед» робот проходит единичный отрезок, а по команде «по-рт» – поворачивается на девяносто градусов по часовой стрелке.

Команда на исполнительные механизмы формируется благодаря анализу входа и треннего состояния конечного автомата. Один и тот же вход может определять раз-команды для исполнительного механизма из-за разных состояний конечного авто-а. Покажем, что конечный автомат *A* (задается конструктором), приведенный в ниже-дующей таблице, достаточен для решения поставленной задачи.

Таблица 1.6.3 – Управляющий автомат А

C \ P	P_1	P_2	$P3$	$P4$
C_1	C_2	C_3	C_4	C_5
C_2	C_2	C_1	C_3	C_5
C_3	C_3	C_1	C_2	C_5
C_4	C_4	C_2	C_3	C_1
C_5	C_5	C_2	C_3	C_1



(x) – захват детали
x – выгрузка детали
O – ориентация робота

Рисунок 1.6.4 – Схема движения робота

Для понимания последующих записей приведем некоторые соглашения по интер-етации команд автомата для исполнительных органов робота. Состояние C_1 будет со-ветствовать состоянию покоя робота, C_4 – прямолинейному движению, C_3 – повороту, – захвату детали, C_5 – выгрузке детали.

Представим себе, что нам задан контур (рис. 1.6.4) и требуется, захватив деталь в нальной точке с координатами $(X_0; Y_0)$, перенести ее в точку с координатами $(X_1; Y_1)$ и м выгрузить, а затем снова вернуться в точку $(X_0; Y_0)$. Ориентация робота в направле-и возможного движения по прямой указана стрелкой на его кожухе.

В нашем случае $(X_0; Y_0)$=(0; 4) и $(X_1; Y_1)$ = (1; 3). Если робот, сориентированный в правлении OX, находится в состоянии (C_1) в точке (0; 4) и готов к движению по ямой от точки (0; 4) к точке (1; 4), то нижеследующая программа обеспечит *выполне-ие* задачи (для наглядности будем писать в скобках соответствующие символы состоя-я, на которое шло входное воздействие символов P_i в виде цепочки смены состояний):

$P_3\ P_2\ P_3\ P_3\ P_4\ P_2\ P_3\ P_1\ P_3\ P_3\ P_3\ P_3\ P_1\ P_3\ P_3\ P_1,\ P_1,\ P_2\ (C_1\ C_4\ C_2\ C_3\ C_2\ C_5\ C_2\ C_3\ C_3\ C_2\ P_3$ $_3\ C_2\ C_3\ C_2\ C_2\ C_3\ C_2\ C_2\ C_2\ C_1)$

Чтение записи происходит на базе таблицы автомата *A* и рисунка контура. В таб-це мы находим, что при входном воздействии P3 на состояние *C1* автомат переходит состояние C_4 (захват детали), затем по воздействию P_2 на C_4 он начинает движение на дин шаг по контуру (состояние C_2) и приходит в точку (1; 4), по воздействию P3 на C_4 беспечивается переход в состояние C_3 (поворот на 90° по часовой стрелке), что соот-

ветствует ориентации робота для движения вниз по прямой и т.д. В итоге подачи все сигналов входной последовательности робот, двигаясь по контуру, снова возвратится точку (0; 4). Его состояние покоя C_1 во второй последовательности выделено особо (на символом C_1 нет никакого входного символа Р). Это объясняется тем, что после подач последнего входного сигнала робот будет находиться в состоянии покоя.

Естественно, что можно начать обход контура и в другую сторону (сначала идти п оси *У*), но тогда изменится лишь цепочка входных воздействий. Можно пойти еще дальше и убедиться, что задача решается и для контуров другой конфигурации из избранного нами класса. Таким образом, на этом небольшом примере можно проиллюстрировать главное преимущество гибких автоматизированных производств с использованием роботов: для избранного класса задач меняются только управляющие программы, а оборудование остается тем же. Не составляет труда сделать и ряд других интерпретаций возможных функций робота, например: сварка деталей по контуру или какой-то линии, шлифовка заусенцев, покраска поверхностей, сверление отверстий в заданных точках и т.п.

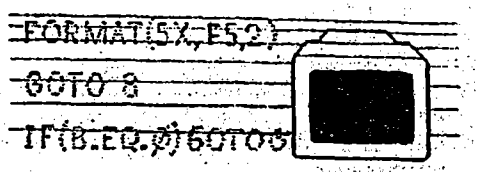
При объяснении процесса движения робота по контуру нам пришлось прибегнуть ряду соглашений, касающихся описания состояний робота, его входов, пошагового принципа перемещения и поворота и т.д. Аналогичные типы соглашений применялись нами при описании программ для машины Поста. Похожее явление встречается и при использовании математической символики для записи различных формул и при доказательстве теорем. Эти символические записи могут читать люди или машины, которые в состоянии интерпретировать их смысл. Понятно, что последовательность символов не может быть любой, часто имеются ограничения на порядок записи символов в зависимости от порядка их следования. Таким образом, мы невольно пользуемся искусственными языками для точного описания различных процессов и объектов. Языки такого типа используются и при описании алгоритмов. Остановимся ниже на особенностях их построения и использования.

Литература

[1, 9, 20, 21, 26, 27, 32, 36, 38, 39, 40]

Контрольные вопросы

1. Чем отличается автомат от других типов устройств для выполнения процессов?
2. Назовите отличительные особенности арифметических и логических операторов.
3. Зачем разработано и используется много средств для описания алгоритмов?
4. В чем заключается особенность численных методов реализации алгоритмов?
5. Укажите на особенности использования ассоциативных исчислений в процессах доказательств и общего описания функционирования ЭВМ.
6. Чем принципиально отличается нормальный алгоритм Маркова от других алгоритмов?
7. Какие основные особенности функционирования ЭВМ можно продемонстрировать на машине Поста?
8. Чем отличается робот от вычислительного устройства любой сложности?
9. Почему модель конечного автомата может использоваться для управления дискретными процессами?
10. Как роботы и конечные автоматы могут использоваться в организации гибких производственных систем?



Глава 2. ЭЛЕМЕНТЫ ТЕОРИИ ФОРМАЛЬНЫХ ЯЗЫКОВ

2.1. Синтаксис и семантика формальных языков

Интерес к применению обычного разговорного языка в диалоговом общении чело-а с ЭВМ и к подготовке программ объясняется целым рядом причин. Исходя из прак-и общения между людьми, пользователи ЭВМ, конечно, хотели бы говорить с ней на ычном языке: подавать устные команды голосом, давать читать текст или в крайнем чае делать набор этого текста на клавиатуре.

Во многих языках программирования эта тенденция проявилась в том, что в них ючаются слова из естественного языка, чтобы облегчить написание программ. Одна-полного совпадения этих языков с естественным не удается достигнуть, и поэтому жам приходится обучаться.

Профессиональные программисты проходят такое обучение, но вместе с ростом можностей машин в различных прикладных областях растет и число людей, которые ели бы пользоваться ЭВМ в своей работе, не проникая в программистские тонкости. йчас резко возросло число таких программ для непрофессиональных пользователей елаются попытки ввести диалоговые средства для пользователя на естественном ыке. Одним из простейших средств такого типа является управление работой про-ммм на уровне «меню», т.е. машина дает рецепты человеку, а он выбирает нужный из (для своего случая. Но все же есть и ряд непреодолимых трудностей по использова-о естественных языков на ЭВМ.

Многие годы в математике и других точных науках широко использовались естест-ные языки (русский, английский и др.)

Однако вместе с необходимостью однозначно описывать различные процессы для следующей реализации на ЭВМ вскрылся ряд недостатков в построении естествен-х языков, и пришлось прибегнуть к созданию специальных формальных языков.

Математические языки отличаются от естественных более простым строением, по-ляющим достигать в их описании предельной четкости и однозначности, зато они ьно уступают естественным языкам в широте сферы применимости.

В естественных языках различают структуру предложений (их форму) и содержа-е (их смысл). Форме конкретного предложения не всегда соответствует единственное ержание. Например, фраза «он встретил ее на поляне с цветами» не является опре-ленной (она была с цветами, поляна была с цветами или он был с цветами).

Фраза «предмет имеет синий цвет» обладает расплывчатым смыслом, так как раз-е люди из множества допустимых голубых и синих предметов выберут разное их ко-чество по признаку «синий».

Грамматические правила естественного языка могут зависеть от смысла получае-ix с их помощью предложений. Например, правильно будет «я вижу стол» (а не стола) я вижу студента» (а не студент). Чтобы правильно строить такие предложения, нужно ать, говорится ли об одушевленном или неодушевленном предмете, а это составляет

часть смысла предложения. Зависимость формы грамматических правил от смысла предложений приводит к ряду затруднений, если мы, например, желаем осуществлять на ЭВМ перевод с одного языка на другой, так как ЭВМ не может понимать смысл. Естественный язык допускает и противоречивые конструкции: «Высказывание, которое я произношу, ложно» (оно не может быть без противоречия ни истинным, ни ложным).

Следовательно, естественные языки противоречивы, неоднозначны и неточны. Наиболее простой путь для преодоления этих затруднений состоит в использовании некоторого подмножества естественного языка, фразы которого однозначны и в смысловом отношении не зависят ни от каких внешних для выбранного подмножества языка условий. Смысл каждой фразы такого подмножества определяется только ее формой.

Для описания формального языка нужен другой язык. Описываемый язык называют языком-объектом в отличие от так называемого метаязыка (мета – вне, за пределом), применяемого для описания языка-объекта. Говоря о языке, в дальнейшем, чтобы избегать путаницы, мы будем четко различать метаязыки (внешние языки) и язык-объект.

Завершенную конструкцию естественного языка обычно называют предложением. Сохраним этот термин и для формальных языков. Внешний язык должен обладать возможностями для описания, как структуры, так и смысла предложений языка-объекта. Часто его разделяют на два языка, один из которых предназначен для описания структуры предложений языка-объекта, а другой – для описания смысла. Первый язык при этом называют метасинтаксическим, а второй – метасемантическим. Систему правил, определяющих структуру предложений языка-объекта, называют его синтаксисом; соответствие между предложениями языка-объекта и их значениями – его семантикой.

При описании структуры предложений формального языка обычно указывают его алфавиты букв и связей и приводят правила для построения предложений (в простейшем случае связей следования алфавит связей опускается). Для определенности будем считать, что каждое правило называет некоторую операцию, которую можно применить в процессе конструирования предложений формального языка, и указывает допустимые исходные данные для этой операции.

Следовательно, под синтаксисом формального языка понимается некоторый перечень операций, для каждой из которых известна область ее определения. Кроме того, предполагается, что синтаксис содержит формулировку некоторого условия, которое выполняется для законченных конструкций формального языка и не выполняется для конструкций, не являющихся его предложениями.

В качестве наиболее известного примера формальных языков можно назвать коды. Внешний язык такого формального языка состоит из двух подъязыков, первый из которых является множеством осмысленных предложений, относящихся к интересующей нас области, а второй содержит описания двух правил, называемых соответственно правилом кодирования и декодирования. Синтаксис формального языка состоит из одного правила кодирования, которое при применении к предложению первого подъязыка дает предложение формального языка.

Правило декодирования определяет обратную операцию по отношению к операции кодирования (которая должна допускать обратную операцию). Вообще говоря, правило декодирования задает семантику кода. Чтобы код был формальным языком, его внешний язык тоже должен быть формальным языком. Например, число семь арабскими цифрами можно следующим образом закодировать в различных системах счисления: 111_2; 21_3; 13_4; 12_5; 11_6; 10_7; 7_8; 7_9 (нижний индекс указывает систему счисления: двоич-

ную, троичную и т.д.). Правило декодирования тогда сведется к алгоритму перевода из кодированной записи в десятичную ($111_2=1*2^2+1*2^1+1*2^0=7$; $21_3=2*3^1+1*3^0=7$; ...).

В большинстве описаний различных разработок по алгоритмическим языкам и языкам описаний различных объектов широко используется язык, предложенный Бэкусом. Этот *метасинтаксический язык состоит из* конечного числа предложений, называемых металингвистическими формулами Бэкуса. При их построении используются два универсальных метасимвола: «:: = » и «|», первый можно читать как «по определению есть», второй – как «или». Остальные метасимволы выбираются произвольно разработчиком формального языка, который является также и разработчиком внешнего языка.

Метасимволы являются произвольными фразами естественного языка, заключенными в угловые скобки (<...>). Будем называть такие метасимволы составными. Также в формулах Бэкуса используются символы формального языка, которые легко узнать по тому, что они отличны от «:: = » и «|» и стоят вне угловых скобок. В левой части формулы Бэкуса должен стоять составной метасимвол, за ним следует знак «:: = », после которого записывается правая часть формулы. Она может быть строкой либо пустой, либо состоящей из конечного числа составных метасимволов и символов формального языка, либо конечной последовательностью таких строк, разделенных знаками «|». По определению две формулы Бэкуса, имеющие одинаковые левые и различные правые части, обозначают то же самое, что формула, которая получится, если к правой части любой из первых формул приписать символ «|», а за ним правую часть другой из них.

Начало и конец формулы Бэкуса ничем не обозначены, поэтому начинать формулу удобно, отступив от начала, а кончать, не доходя до конца строки. При переносе со строки на строку никаких дополнительных знаков ставить не следует. Смысл формулы Бэкуса легко понять, читая ее на естественном языке (при этом угловые скобки не произносятся), пустая строка в правой части формулы читается как «пусто».

Пример. Формальный язык для представления четных натуральных чисел с помощью формулы Бэкуса может быть описан так:

<ненулевая цифра> :: = 1|2|3|4|5|6|7|8|9|
<цифра> :: =0| <ненулевая цифра>
<четная цифра> :: = 0| 2| 4| 6| 8|
<начало> :: = <ненулевая цифра> |
<начало> <цифра>
<четное число> :: = <четная цифра>| <начало> <четная цифра>

Читая эти строки на естественном языке, убеждаемся в однозначности восприятия определяемых в них понятий: четко *сказано, что (ненулевая* цифра) есть по определению 1, или 2, или 3, или 4, или 5, или 6, или 7, или 8, или 9, т.е. фактически понятие задано прямым перечислением составляющих его объектов: (начало) есть по определению (ненулевая цифра) или (*начало*) (цифра), т.е. здесь указывается, что (начало) всегда будет числом, которое начинается не с нулевой цифры. Например, 2, 25 и 20 подпадают под определение *понятия (начало), так как 2* – ненулевая цифра в первом случае, а во втором и третьем случаях 2 играет роль начала, а другие цифры могут быть любыми. Анализируя аналогичным образом понятие (четное число), видим, что оно может состоять *из* одной четной цифры или начала (любого числа) с *окончанием на четную цифру. Действительно,* это правило позволяет порождать четные числа из любого количества цифр, так как, приписывая к началу справа еще одну цифру, мы снова получаем начало, к которому можно приписать четную цифру.

Если *для формального языка известно* точное и однозначное правило, преобразующее его предложение в предложения некоторого другого языка, обладающего семантикой, то тем самым задается семантика нашего формального языка. Семантика называется формальной, если указанное правило *является алгоритмом.* Теория формальных грамматик занимает в математической лингвистике центральное *место, так как она обеспечивает* возможность вести переработку смыслов в тексты и обратно.

Формальная грамматика – система правил, описывающая множество конечных последовательностей символов. Конечные последовательности символов (цепочки), входящие в указанное множество, называются предложениями, а само множество – языком, описываемым данной формальной грамматикой.

Различают два типа формальных грамматик: грамматики порождающие – системы правил, позволяющие строить предложения языка, и грамматики распознающие – системы правил, распознающие по любой цепочке, является ли она предложением. Это деление в некоторой степени условно, так как любая распознающая грамматика, по существу, задает способ построения предложений.

Формальные грамматики чаще всего применяют для описания естественных и искусственных (в частности, алгоритмических) языков. Мы остановимся более детально на изучении порождающих грамматик, исходя из следующих соображений:

1.Математическое значение порождающих грамматик определяется тем, что они *представляют собой* одно из средств эффективного задания множеств.

2.Одним из важных направлений теории порождающих грамматик является изучение сложности вывода как по числу шагов вывода (временная сложность), так и по объему используемой памяти (по максимальной промежуточной цепочке вывода).

3.Большая прикладная роль порождающих грамматик проявляется в построении и изучении искусственных языков, в особенности языков программирования, а также и в ряде других случаев (исследование естественных языков, машинный перевод и т.д.).

Порождающей грамматикой называется упорядоченная четверка $\Gamma = \langle V, W, i, R \rangle$ (которая включает систему правил для построения последовательностей символов), где V и W – непересекающиеся конечные множества, называемые соответственно основным и вспомогательным алфавитом (словарем). V обычно называют словарем терминальных, а W – словарем вспомогательных (нетерминальных) символов; i является элементом W и называется начальным символом (или выделенным вспомогательным символом); R – набор правил вывода, или синтаксических правил, каждое из которых имеет вид $\varphi \to \phi$, где φ и ϕ – цепочки, состоящие из основных и вспомогательных символов.

Основные символы обычно интерпретируются как слова языка, вспомогательные – как названия классов слов и словосочетаний, начальный символ – как символ предложения.

Синтаксические правила описывают связи между частями предложения. Применение правила $\varphi \to \phi$ к цепочке, имеющей вид $\alpha\varphi\beta$, означает преобразование ее в цепочку $\alpha\,\phi\,\beta$ (здесь α и β – цепочки, одна из которых или даже обе могут быть пустыми).

Вывод в данной порождающей грамматике есть последовательность цепочек, в которой любая цепочка, кроме первой, получается из предыдущей применением какого-либо правила вывода.

Цепочка основных символов, выводимая из начального символа, называется предложением, а множество всех предложений – языком, порождаемым данной грамматикой.

Приведем пример.

Пусть *V = {a, b, c}, W = {A, B}, I = A,*
R = {A->aAB, A->Bc, B->b}.

Одним из выводов будет последовательность: A, *aAb, aBcB, abcb.* Цепочка *abcb* – едложение. (Другой пример вывода: A, *Bc, bc.)*

Основные классы порождающих грамматик выделяются в зависимости от ограни-ний, налагаемых на вид синтаксических правил.

В бесконтекстной или контекстно-свободной грамматике (КС – грамматике) исполь-ется правило вывода A-> ф, где *A* – вспомогательный символ и ф – непустая цепочка. ыки, порождаемые такими грамматиками, называются бесконтекстными языками.

В грамматике непосредственно составляющих (НС – грамматике) или контекстной амматике синтаксические правила имеют вид vAw->vфw). В НС – грамматике каждый г вывода состоит в замене одного вхождения символа *A* вхождением цепочки ф, при-м замена обусловлена наличием контекстов V и ф. В НС – грамматиках на каждом ша-вывода заменяется только один символ, поэтому в них с каждым выводом предложе-я ассоциируется так называемое дерево вывода (рис. 2.1.1), строящееся следующим разом.

Корень дерева соответствует начальному символу. Каждому символу цепочки, на торую заменяется начальный символ на первом шаге вывода, ставится в соответствие ел дерева, и к нему проводится ветвь от корня. Для тех из полученных узлов, которые мечены вспомогательными символами, строится аналогичная конструкция и т.д.

Например, если грамматика содержит следующие правила: *I->AAB; A->a ; B->C;* >c (основные символы – *a, b, c; A, B, C* – вспомогательные символы, I – начальный мвол), то вывод (I, *AAB, aAB, aaB, aaC, aac)* можно представить в виде иерархической руктуры (дерева) так, что 1 будет корнем; A, A, B – тремя вершинами второго уровня; a, C – тремя вершинами третьего уровня; c-вершиной четвертого уровня.

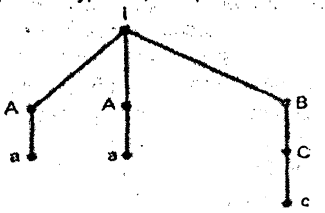


Рисунок 2.1.1 – Дерево вывода

Иногда в определение грамматики еще включают алфавит связей между буквами ми словами, а также выделяют операции и формулы, которые используются в построе-ии цепочки, являющейся предложением.

В теоретических исследованиях разных авторов имеется сильное взаимное про-икновение теории автоматов и математической лингвистики, одним из важных объек-ов которой являются порождающие грамматики. В некотором смысле к классу порож-ающих автоматов можно отнести и любой вычислитель (человека, машину), который в роцессе вычисления пишет цифры или другие знаки; эти знаки можно рассматривать ак элементы, которые он порождает в процессе вычисления.

Математический вывод всегда осуществляется средствами заданной формальной истемы. Сначала происходит кодирование входной информации в языке данной фор-альной системы (или машины), а затем эти коды перерабатываются в соответствии с равилами функционирования рассматриваемой системы (или машины). Процессы вы-

вода в порождающих грамматиках аналогичны преобразованиям слов в ассоциативных исчислениях.

2.2. Использование формальных языков для поиска информации

Одним из самых распространенных и важных применений формальных языков являются задачи автоматизированного информационного поиска. Развитие научно-технического прогресса сопровождается резким увеличением потока информации.

В США на поиск научно-технической информации тратится более миллиарда долларов в год. На дублирование разработок из-за неполной информированности тратится около 10% всех средств, расходуемых на *новые разработки.* Поэтому *поиск* информации превратился в крупную научную проблему.

Задачи обработки информации, *библиографического поиска и анализа фактографических* данных часто объединяют под общим названием информационно-логических *задач. Для них характерны логическая обработка больших* объемов информации и ее хранение. Информация часто представляется не только в количественной, *но и в качественной форме. В* первую очередь такого рода задачи используются при применении различных автоматических систем обработки информации, при поиске конструкторско-технологической информации в системах автоматизированного проектирования, в оперативном управлении предприятиями.

Применение автоматических библиографических систем имеет большое значение *для развития всех областей науки и техники.* Число публикаций и разработок по всем отраслям знаний *настолько велико, что специалисты не в* состоянии следить за ними даже в своих узких областях. Из-за трудностей поиска иногда легче произвести исследования или разработки заново, чем найти их описание.

Фактографические системы поиска информации представляют собой новый уровень автоматизации умственных процессов. Они должны позволить не только находить соответствующую литературу, но и выполнять логический анализ и сопоставление содержания различных статей и книг, производить обобщение сведений и т.д.

Библиографические дескрипторные системы накопления, хранения, обработки и поиска информации условно делятся на два типа:

1) дескрипторные системы без грамматики;

2) дескрипторные системы с грамматикой.

Рассмотрим сущность дескрипторного метода поиска. Содержание документа в общих чертах может быть представлено набором характерных для данного текста слов. Их называют ключевыми словами для данного текста. Для построения поисковой системы из всего многообразия ключевых слов, выбранных из ряда характерных для данной тематики текстов, составляется стандартный набор терминов без синонимов со строго фиксированными значениями. Эти термины называются дескрипторами, а полный набор таких терминов называется словарем дескрипторов. Для каждого документа составляется свой набор дескрипторов, называемый поисковым образом документа.

Массив дескрипторных наборов документов может быть построен двумя различными способами: прямым и инверсным. При прямом способе в памяти машины последовательно записываются номера документов и за каждым из них указываются дескрипторы или их коды. Процесс поиска заключается в последовательном сравнении для каждого документа всех дескрипторов запроса с дескрипторами документа и выделении тех документов, для которых выполняется критерий соответствия.

При инверсном способе за основные позиции принимают не документы, а дескрипторы. Для каждого дескриптора записываются все номера документов, которые имеют в своих поисковых образах этот дескриптор. Поиск нужных документов осуществляется в словаре дескрипторов путем обращения к тем дескрипторам, которые имеются в запросе, и путем выбора всех номеров документов этих дескрипторов. Отобранными считаются те документы, номера которых оказались общими для всех дескрипторов, указанных в запросе.

Например, пусть требуется найти литературу по алгоритмическим языкам для программирования экономических задач в заданном массиве документов:

Таблица 2.2.1 – Инверсный массив

Дескрипторы / Наименование	Коды	Номера документов
Вычислительная машина	101	0031. 0034. 0038. 0045
Логическая обработка	102	0012. 0026. 0031. 0046
Алгоритмический язык	105	0003. 0022. 0027. 0031
Экономика	205	0031. 0168. 1342
Оптимизация	337	0512. 1314

Наш запрос можно записать следующим образом: 101, 105, 205

Шаги поиска:

1) 0031, 0034. 0038. 0045 (дескриптор 101);

2) 0003, 0022. 0027. 0031 (дескриптор 105);

3) 0031, 0168, 1342 (дескриптор 205);

4) 0031 – ответ.

Ответ получается как отыскание общей части для всех шагов поиска.

Иногда дескрипторы запросов делятся на категории, указывающие их важность.

В простейшем способе дескрипторного поиска дескрипторы, относящиеся к документам, и дескрипторы, входящие в состав запроса, представляют собой простые наборы, не связанные какой-либо грамматикой, в частности, порядок их расположения безразличен. Однако такой способ приводит либо к выдаче излишних документов, не относящихся к интересующему нас вопросу, либо к пропуску нужных документов.

Например, слова «банк, данные, электронный» могут встретиться в тексте о банке, ведущем финансовые операции и запирающемся на электронный замок, а также в тексте об электронном банке данных научной информации.

Более эффективными являются дескрипторные поисковые системы с грамматикой, в которых дескрипторы, относящиеся к документам, и дескрипторы, входящие в состав запросов, снабжаются дополнительными символами, указывающими на семантическую роль этих дескрипторов (объект процесса, процесс, причина).

Иногда роль дескрипторов определяется их положением в наборе или связями между отдельными дескрипторами. В этом случае дескрипторные системы приближаются по своему принципу действия к фактографическим информационным системам с грамматикой. Например, для приведенного выше запроса можно потребовать, чтобы слова электронный банк дан ных» шли в заданном порядке и неразрывно в одном предложении. Для фактографических информационных систем с грамматикой строится специальный информационный язык, позволяющий записывать фактические сведения, относящиеся к тем или иным областям знаний.

Он обычно состоит из следующих четырех основных частей;

1) набора базисных терминов, обозначающих основные предметы данной области знаний (например, для вычислительной техники двоичный разряд, машинное слово и т.п.);

2) набора смысловых отношений между предметами (примеры отношений: один предмет является элементом класса, представляющего другой предмет; предмет является субъектом процесса; предмет является объектом процесса и т.п.);

3) набора формальных правил (синтаксис), позволяющих из основных терминов и отношений языка строить более сложные термины и отношения, а также осуществлять переход от информационного языка к естественному;

4) системы кодирования понятий, отношений и синтаксических правил языка, позволяющей осуществлять преобразование и хранение информации, представленной на этом языке, с помощью ЭВМ.

Сложные фактографические системы обычно предназначены для выдачи фактического материала на различные вопросы определенного круга и характера.

Простейшей системой может быть такая, которая осуществляет проверку вводимых в машину утверждений и дает 3 ответа: «да», «нет», «неизвестно».

В качестве иллюстрации одного из возможных специальных информационных языков для поиска конструкторско-технологической информации на основе порождающей грамматики Г приведем пример.

Пусть *V* – множество понятий (терминологические словосочетания, характеристики объектов), *Г* – порождающая грамматика.

Г=<*V,W,I,R*>, где *V* – основной словарь, *W* – вспомогательный словарь, *I* (*I* ∈ *W*) – начальный символ, R – множество правил подстановки, которые представим в форме Бэкуса с использованием знаков (и), (или), ¬ (нет): <описание объекта на информационном языке> :: = <логическое произведение сообщений>

<логическое произведение сообщений> :: = <сообщение> | <сообщение> ^ <логическое произведение сообщений>

<сообщение> :: = <логическая сумма фраз>

<логическая сумма фраз> :: = <фраза> | <фраза> V <логическая сумма фраз> < фраза > :: = < понятие >| ¬ < понятие > | <характеристика> | ¬ <характеристика>

<характеристика >::==< понятие > < понятие> | <понятие> <знак отношения> <значение числовой характеристики>

< знак отношения >::=>|<|> = |< = | = <значение числовой характеристики> :: = <положительное число> | <отрицательное число>

Язык, порождаемый грамматикой Г, будем называть информационным языком, а элементы множества *W* – сообщениями на информационном языке. Для осуществления поиска нужных сведений в этом случае сообщения на языке взаимодействия переводятся на информационный язык (кодирование) и после получения ответа на информационном языке проводится его декодирование. Сложность поиска для пользователя в этом случае сводится к умению описать требуемый объект, используя его характерные признаки в их логической взаимосвязи.

Из последних разработок представляет значительный интерес система фактографического типа на научно-производственном объединении «Пластполимер». В ней хранятся сведения практически о всех полимерных материалах. Проводя диалог с ЭВМ, можно получить необходимые сведения о любом из девяти тысяч марок полимерных материалов и их известных свойствах. Аналогичная информация имеется в ЭВМ и о свойствах деталей, выпускаемых из полимеров.

Особенность автоматизированной системы состоит в том, что она не только дает положительный или отрицательный ответ, но, учитывая предъявляемые к материалу требования, рекомендует, какой полимер выбрать, а в случае отсутствия возможности

ать точный ответ предлагает рекомендацию о материалах с близкими свойствами по ношению к заданным.

Развитие поисковых систем такого типа в будущем окажет значительное влияние а развитие научно-технического прогресса и поднятие культурного уровня специалистов в различных областях деятельности. Уже современные автоматизированные хранилища позволяют в малых объемах сосредоточить колоссальные объемы информации. Например, с помощью лазерной технологии порядка тысячи страниц текста и чертежей можно записать на специальной пластинке, которая по размерам примерно равна этикетке спичечного коробка. Телевидение, спутниковые и другие средства связи позволяют обеспечить передачу информации на большие расстояния.

В настоящее время страны выработали международные коммуникативные форматы, позволяющие передавать информацию по каналам связи. Например, информацией на машинных носителях в области изобретательской деятельности сейчас обмениваются ряд стран. Аналогичные обмены ведутся и в области реферативной информации по публикуемым источникам.

Таким образом, использование стандартных форм описания документов позволяет упростить решение такой важнейшей задачи, как международный обмен информацией во всех областях деятельности человека. Большие перспективы открываются и в области автоматического перевода с разных языков на язык страны-потребителя информации.

В технических областях знаний запись информации на практически однозначно воспринимаемых машиной подмножествах естественного языка позволяет уже сегодня получать машинные переводы, пригодные для понимания затрагиваемого вопроса специалистом. Поэтому информационная система, снабженная соответствующими программами для перевода с наиболее употребительных языков, позволит специалистам, минуя языковой барьер, знакомиться с последней информацией в мире в конкретных областях деятельности.

2.3. Общие подходы к построению алгоритмических языков и трансляторов

Алгоритмический язык является средством точного формулирования вычислительных процессов для их последующей реализации на вычислительных машинах.

Можно назвать три основных круга задач и процессов, для описания которых созданы алгоритмические языки:

1) алгоритмы численного анализа (расчеты по формулам и т.п.);

2) процессы обработки данных в экономических расчетах (обработка таблиц);

3) переработка символьной информации (обработка текстов, аналитические выкладки, моделирование интеллекта и т.п.).

Алгоритмический язык сам по себе может лишь использоваться как средство для публикации алгоритмов, удобное для однозначного восприятия человеком. Эффективность алгоритмического языка возрастет во много раз, если на ЭВМ имеются соответствующие средства трансляции (перевода) алгоритма на язык ЭВМ.

Разработка системы программирования, базирующейся на алгоритмическом языке, довольно трудоемка и состоит из следующих этапов:

1) отбора изобразительных средств языка;

2) описания языка;

3) разработки транслятора.

Охарактеризуем кратко каждый из этапов.

Отбор изобразительных средств языка связан прежде всего с анализом классо задач, подлежащих решению. Основная цель анализа – выделить в рассматриваемы задачах устойчивые и наиболее часто встречающиеся структуры объектов языка и дей ствия над ними, которые после их формализации и введения соответствующей симво лики объявляются элементарными объектами и базисными операциями языка.

Например, в арифметических вычислениях это числа, скобки, операции сложени вычитания, умножения, деления и правила выполнения действий при наличии скобок.

В языке должны быть правила, позволяющие из элементарных операций и объе тов закономерно строить более сложные. Для операций таковыми являются в основно правила образования линейных структур с использованием разного рода скобок и вве дением функциональных обозначений.

Алгоритмы численного анализа находятся в лучшем положении, так как для ни уже существовала сложившаяся веками математическая символика. Кроме того, приме нение ЭВМ началось с автоматизации арифметических вычислений. Поэтому для ни были созданы первые в мире системы автоматизации программирования и первый ал горитмический язык, ставший международным стандартом, – АЛГОЛ-60.

Базисными операциями в языках для алгоритмов численного анализа являютс арифметические действия над числами и их отношения, а также элементарные функци математического анализа. Элементарными объектами являются числа и числовые пе ременные, которые могут объединяться в линейные массивы (векторы, матрицы) с ука занием компонентов массива при помощи индексов-координат.

Основными средствами описания сложных структур в языках являются: объедине ние объектов в линейные последовательности, образование скобочных структур функциональные обозначения.

Одна из главных проблем в языках обработки экономической и бухгалтерской ин формации – представление данных, носящих смешанный текстово-числовой характер соблюдением табличной структуры и двумерной соподчиненности и характерных дл экономической и коммерческой документации. Задачам этого класса свойственна негл бокая переработка больших массивов информации. Поэтому класс базисных операци не выходит за пределы арифметических действий.

В то же время большое место занимают операции поиска и обработки данных, также описания формата и структуры входной информации.

Проблема обработки символьной информации возникла только с появлением вы числительных машин. Поэтому авторы первых алгоритмических языков из рассматри ваемого класса вводили тот или иной набор операций и объектов как продукт некоторо умозрительного и априорного представления о закономерностях обработки символьно информации.

Отбор изобразительных средств языка – противоречивый и сложный процесс, ко торый требует учета последующих стадий разработки системы программирования. Со став и грамматический строй алгоритмического языка всегда являются компромиссо между стремлением к богатству и разнообразию операций и объектов и стремлением его простоте и лаконичности, которые позволяют обеспечить создание транслятора упростить его работу.

На втором этапе составляется описание языка. Предполагается, что любой язы представляет собой некоторое множество осмысленных текстов. Описать язык – значи задать правило, которое определяет, какие тексты могут быть текстами языка (синта

сис), и установить, какие процессы или объекты могут описываться синтаксически допустимыми текстами языка (семантика).

В настоящее время общепринятым методом описания синтаксиса алгоритмических языков стал метод порождающих грамматик для языков с фразовой структурой. Первым языком программирования, для описания которого был применен этот метод, является АЛГОЛ-60.

Язык с фразовой структурой, задаваемый порождающей грамматикой, имеет три компонента:

1) набор основных символов (алфавит языка: буквы, цифры, разделители, знаки операций и т.д.);

2) набор имен или символов для обозначения «частей речи» языка;

3) набор математических формул, указывающих, как данная часть речи может быть сконструирована из других частей путем сочетания последних в некотором заданном порядке.

Среди частей речи есть элементарные, которые не сводятся ни к каким другим и выражаются только через основные символы, и сложные, представляющие наиболее всеобъемлющие единицы языка. Они получаются путем свободного сочетания значений элементарных частей речи, которые входят в металингвистическую формулу, задающую часть речи. Множество значений сложной части речи и есть множество синтаксически допустимых текстов языка. Особенностью языков с фразовой структурой является рекуррентное строение многих металингвистических формул, когда часть речи может, в частности, определяться сама через себя, что дает возможность создавать периодические и вложенные структуры.

Достоинство определений через металингвистические формулы заключается в их краткости и недвусмысленности. Авторы АЛГОЛа изложили понятия языка с помощью металингвистических формул, предпослав изложению эпиграф: «То, что вообще может быть сказано, может быть сказано ясно, а о чем невозможно говорить, о том следует молчать». Металингвистическими формулами очень удобно пользоваться для справок.

С помощью металингвистических формул можно давать определение лишь объектам, которые представляют собой конечные последовательности некоторых символов. Ошибки в программе или в элементах программ (в выражениях, операторах и пр.), обусловленные несоответствием программы или элементов программы металингвистическим формулам, называются синтаксическими ошибками. Программы или какие-либо элементы программы, не содержащие синтаксических ошибок, называются синтаксически правильными. Синтаксическая правильность программы не означает, что в программе вообще нет ошибок. Например, средствами синтаксического контроля нельзя обнаружить ошибку, допущенную автором, если синтаксически фраза не противоречит правилам ее написания. Это можно сравнить с известным примером, когда перестановка запятой на другое место решает судьбу человека: «Помиловать, нельзя повесить», хотя автору текста хотелось отдать другой приказ: «Помиловать нельзя, повесить».

На третьем этапе разрабатываются трансляторы с алгоритмического языка, представляющие особые программы, которые, воспринимая текст задачи на алгоритмическом языке, осуществляют его обработку и обеспечивают выполнение на машине предписанного текстом алгоритма. Проблема состоит не только в том, чтобы найти алгоритмы выполнения отдельных функций транслятора, но и в том, чтобы отобрать нужные алгоритмы и объединить их в единую систему. Как правило, трансляторами также проводится синтаксический контроль программ, заданных на алгоритмических языках.

Алгоритмические языки вместе с улучшением конструкций вычислительных машин и усложнением производственных задач непрерывно совершенствуются. Часть из них отмирает, рождаются новые языки. Например, причиной рождения новых языков послужило появление многопроцессорных машин, позволяющих одновременно выполнять независимые цепочки операций в одном алгоритме.

Трудности в освоении ранее созданных программ и проблема доверия к их правильности привели к необходимости создания языков, которые хорошо описывают структуру программ в смысле их простого чтения человеком (ПАСКАЛЬ). Управление производственными процессами в реальном масштабе времени потребовало развития соответствующих языков (АДА). Естественно, что новые черты в рождающихся языках требуют совершенствования и средств трансляции, а также методики их использования.

Алгоритмы, описанные на различных формальных языках, как уже отмечалось выше, не могут непосредственно выполняться на цифровой электронной вычислительной машине, так как они, как правило, не представлены на языке ее команд. Языки описания алгоритмов обычно называют языками более высокого уровня по отношению к машинному описанию алгоритмов на языке команд, хотя тот и другой языки являются формальными. Запись алгоритма на машинном языке сразу очень трудоемка из-за слишком элементарных шагов его описания, предназначенного для выполнения ЭВМ. На языке высокого уровня программа, как правило, не привязана к конкретной марке ЭВМ, занимает меньше места при записи и легче обозрима человеком из-за структурированности программы на уровне крупных блоков и операторов, а на проблемно-ориентированных языках она легко воспринимается специалистом, который не знает в деталях особенности ее машинной реализации. Поэтому существуют программы (трансляторы), которые выполняют функции посредника (переводчика) для обеспечения выполнения алгоритма на ЭВМ.

Строго говоря, транслятор – программа, предназначенная для перевода (трансляции) описаний алгоритмов с одного формального языка на другой.

Первый язык называется входным языком, а второй – выходным.

Работа трансляторов протекает в зависимости от принятой схемы получения машинной программы. Часто удобно использовать ступенчатую схему трансляции.

В схеме непосредственной трансляции выходным языком служит язык команд конкретной ЭВМ, на которой осуществляется трансляция. В практике используются трансляторы компилирующего и интерпретирующего типов.

Компилирующий транслятор обычно более сложен по конструкции, так как он сначала получает машинный вариант программы, затем она может выполняться сразу или спустя некоторое время. Эти трансляторы содержат развитые средства оптимизации программ и более развитую структурную схему для диагностики ошибок синтаксического типа в алгоритмах на языках высокого уровня.

В трансляторах интерпретирующего типа процесс перевода программы на машинный язык совмещается с выполнением получаемой выходной программы. Эти трансляторы более удобны для работы в диалоговом режиме. Поэтому они получили широкое распространение на современных персональных микро-ЭВМ. Пошаговое выполнение алгоритма позволяет быстрее убедиться в его *работоспособности и помогает* сразу скорректировать обнаруженные ошибки. Пошаговый принцип работы иногда используется и в компилирующих трансляторах.

Процесс трансляции можно разбить на несколько составных частей: синтаксический анализ и контроль текста на входном языке (обнаружение конструкций, записанных

тклонениями от правил их записи); анализ описания данных и распределение памяти я *них и для программы;* получение и оптимизация текста программы, выдача текста транслированной программы. Часть из этих функций в некоторых трансляторах может сутствовать, например оптимизация.

По характеру работы близки к трансляторам и различные конверторы и эмуляторы. можно рассматривать как трансляторы с более узким выполнением функций. Программы эмуляции используются для интерпретации команд одной машины на другой. К услугам прибегают при моделировании работы проектируемых ЭВМ или же при необходимости сохранить возможность выполнения программ для ЭВМ старых моделей на вых ЭВМ.

2.4. Операционные системы

Операционная система (ОС) – комплекс программ, осуществляющих управление числительным процессом и реализующих наиболее общие алгоритмы обработки информации на данной ЭВМ. Из приведенного определения трудно выделить функциональные границы этого комплекса программ. Он может быть по функциональному составу мощнее на больших ЭВМ и слабее на малых. Программы операционной системы полняют всегда две главные функции: обеспечение работы самой ЭВМ и организация имодействия человека с ЭВМ. Структура современных развитых операционных систем больших ЭВМ очень сложна, а тексты ее программ могут состоять из миллионов волов информации. Главными в обеспечении работы ЭВМ считаются управляющие ограммы, которые являются программным продолжением электронного устройства авления. Они используются для реализации функции управления заданиями, распределения памяти, управления обменом данными, для фиксации сбоев в работе, для ения протокола вычислительного процесса и т.п.

Основной управляемой единицей ОС является задание. С точки зрения пользования ЭВМ, результат его обработки является конечным продуктом. Задания не зависят т от друга. Каждый пользователь для управления своим заданием записывает для специальные управляющие предложения, характеризующие его требования по выполнению каждого шага задания.

Особая роль среди управляющих программ отводится супервизору, который всегда одится в оперативной памяти. Супервизор ведет текущее обеспечение выполняемых гов заданий, ведает такими ресурсами машины, как память, управляет процессами мена с внешними устройствами.

Планирование распределения ресурсов осуществляется на основе анализа потока даний. Информация о ресурсах для данной задачи описывается пользователем ЭВМ имерный объем памяти, необходимые устройства ввода – вывода и т.п.). В соответствии с описанием процесса выполнения отдельной программы ее необходимые части менты) для каждого шага используются супервизором как единицы планирования загрузки в оперативную память. Супервизор также вызывает сегменты с внешних опителей для их выполнения и обеспечивает сохранение связи между отдельными иентами. Супервизор обрабатывает сигналы прерываний, поступающие от аппаратной части ЭВМ. Они формируются в процессе обработки заданий или же контрольной аратурой ЭВМ.

Управление данными представляет собой обобщение понятия системы управления вводом – выводом, которая организует ввод – вывод при выполнении вычислительной системой всех возможных заданий. Диагностика сбоев и неисправностей машины

сообщается оператору и он принимает соответствующие меры по устранению неисправностей или же продолжает решение оставшейся части задач, которые не используют вышедшей из строя аппаратуры. Весь ход вычислительного процесса на ЭВМ протоколируется автоматически.

Одной из наиболее сложных функций ОС является поддержание режима диалога, когда пользователи работают за экранными пультами (дисплеями) в непосредственной близости от ЭВМ или же на значительном удалении от нее. В этом случае каждый пользователь получает квант времени, в течение которого решается его задача. По истечении этого времени решение задачи прерывается, а состояние ЭВМ запоминается на уровне информации, необходимой для продолжения решения задачи. За один проход ЭВМ продвигает решение задачи каждого пользователя или же повышает его приоритет в очереди, а затем цикл повторяется. При удачном подборе числа дисплеев и задач, согласующихся с быстродействием процессора, у пользователя создается «иллюзия», что он один работает за пультом машины. Схемы организации такой работы могут идти и на приоритетной основе, когда для более важных задач кванты времени выделяются чаще, или же кванты времени имеют большую длительность. Еще сложнее обеспечить работу программ, которые используются для управления некоторыми процессами в реальном масштабе времени. Когда результат их работы должен быть известен не позже наперед заданного момента времени.

Кроме того, ОС для конкретной ЭВМ с учетом состава ее оборудования и предполагаемого характера решаемых задач может с помощью специальных программ генерироваться в наиболее рациональном варианте с точки зрения затрат различных ресурсов на обслуживание пользователей. Таким образом, очевидно, что ОС является фактически довольно сложной автоматизированной системой специального назначения с рядом программ, имитирующих деятельность человека по эксплуатации и разработке программ. Создание таких ОС с элементами искусственного интеллекта за относительно короткий срок с момента появления машин, которые обслуживают одновременно нескольких пользователей, объясняется тем, что их создатели-программисты должны были алгоритмизировать свой собственный труд и им не потребовалось привлекать знания из других областей деятельности человека.

Вместе с тем сложность современных ОС на больших ЭВМ возросла до такой степени, что воспользоваться всеми возможностями, которые они могут предоставить, не всегда в состоянии даже опытный программист. Поэтому операционные системы для персональных ЭВМ и больших ЭВМ для пользователей-непрофессионалов в программировании стали снабжать элементами «дружественного» интерфейса, направленного на облегчение диалога человека и машины. ЭВМ взяла на себя функции по обучению и консультированию пользователя при решении им конкретных задач: на базе тестовых примеров советовать пользователю, какой алгоритмический язык ему выбрать; на какие конструкции операторов и специальных средств следует ориентироваться; какими подсказками ЭВМ воспользоваться на базе ограниченного выбора действий («меню») и т.д.

Операционные системы с каждым днем продолжают совершенствоваться, и степень их доступности для пользователей будет повышаться. Обобщение материалов по существующим ОС и перспективным разработкам ОС идет пока медленно. Это связано с тем, что большинство учебников по ОС тесно привязано к конкретным маркам машин, но постепенно общие закономерности начинают выкристаллизовываться и находить отражение в литературе по системному программированию. Следует отметить, что долгие годы работа по обеспечению «дружественного» интерфейса системными программи-

стами-профессионалами считалась черновой и неинтересной, так как она была направлена лишь на облегчение освоения ОС пользователем и не касалась основных функций ОС.

Сейчас время внесло свои коррективы в этот подход. Постепенно количественный рост числа системных программистов и вообще чистых программистов замедлился. С каждым днем стало возрастать число людей-непрограммистов, желающих использовать вычислительную технику для решения задач в своей области. Количественное преобладание таких пользователей ЭВМ проблему создания «дружественного» интерфейса для них сделало главной: стали появляться специальные технические и программные средства для облегчения контакта непрофессионала с ЭВМ. Работа со многими программными средствами стала сводиться к управлению движением маркера на экране дисплея в нужных областях, к указке «световым» пером нужного объекта или его элемента, звуковому вводу информации на основе ограниченного словаря естественного языка и т.д.

Сейчас все яснее намечается тенденция к сведению интерфейса человека с ЭВМ к привычной для него схеме диалога с другими людьми. Естественно, как отмечает академик А. П. Ершов, придется пока считаться с трудностями проблемы общения и на первых порах пользоваться в «разговоре» с ЭВМ языком «деловой прозы», т.е. в соответствии с ориентацией на решение определенных задач будет и соответствующее языковое обеспечение данной группы специалистов. Таким образом, ОС, по-видимому, не сможет впитать в себя все виды программ, обеспечивающих диалог в специальных областях, но должна постепенно впитывать в себя элементы, пригодные для всех областей деятельности.

Идеи, развитые при создании и совершенствовании операционных систем, оказались весьма плодотворными в автоматизации процессов разработки алгоритмов и решения различных классов задач.

2.5. Базы данных и знаний, СУБД

Большинство сложных производственных автоматизированных систем различного назначения требует не только использования специфических процессов для подготовки исходных данных для решения отдельных задач, но и обмена данными с выполняемыми одновременно другими задачами, а при создании интеллектуальных систем и подбора методов для разрешения создавшихся ситуаций. Это привело к развитию систем, близких по духу к операционным, но имеющим целевое назначение – автоматизацию подготовки, обработки, хранения, защиты и актуализации данных, включая такие интеллектуальные процессы, как выбор методов, и оценку путей решения текущей задачи.

Эта ситуация в своем разрешении начала свой путь с формирования понятий базы данных и знаний. Идея базы данных имеет в своей основе подход в решении ряда задач, когда исходные данные и другие промежуточные результаты хранятся отдельно от исполняемых программ и с помощью заранее описанных механизмов управления извлекаются из базы данных по стандартным схемам, используемым каждой задачей независимо от вычислительного процесса на стадии подготовки исходного материала для текущего этапа его реализации. Базу данных часто определяют как совокупность взаимосвязанной информации, организованной по определенным правилам. База данных обычно определена по схеме, не зависящей от программ, которые к ней обращаются. Чтобы легче искать данные и их обрабатывать, обычно используют системы управления базами данных (СУБД), которые ориентированы на поддержку выполнения различных запросов и ведения БД. На практике шире других используются реляционные СУБД, когда данные отображаются в табличной форме. В каждой СУБД имеется специальный

язык запросов, обеспечивающий эффективный доступ к различным элементам (или их группам) в базе данных.

В отличие от базы данных, в базе знаний располагаются проверенные и накопленные человечеством истины, факты, принципы и другие аналогичные объекты. Обычно их источником являются документы, книги, статьи и т.п. В базе знаний (БЗ) располагаются в соответствии с принятой системой классификации объекты познания, представляющие собой совокупность знаний. Каждый из объектов представляет набор элементов знаний. Благодаря использованию концептуальных связей объекты объединяются, образуя базу знаний. Каждая база знаний включает в себя набор сведений, правила и механизмы логического вывода.

Такие базы знаний являются необходимым компонентом при решении задач искусственного интеллекта, в частности, при создании экспертных систем, так как они позволяют получать знания, которые непосредственно не создаются в БД, а определяются с помощью правил вывода.

В экспертных системах знания специалистов являются источником формирования баз знаний, а правила вывода могут использоваться из универсальных систем или их стандартных оболочек.

На основе баз данных и знаний в различных автоматизированных системах используются банки данных. Вместе с принятой к использованию СУБД они являются его ядром. Автоматизированные системы управления, спроектированные на фундаменте концепций банков данных, позволяют обеспечить многоаспектный доступ к совокупности взаимосвязанных данных, интеграцию и централизацию управления данными, устранение их избыточности и защиту, возможность телепроцессорной обработки. По своей сути банк данных является информационной системой. Формирование и ведение банков данных связаны с большими затратами, которые окупаются при большом трафике (потоке запросов). В ряде разработок применяется реляционная (табличная) модель данных, которая поддерживается соответствующими СУБД. Опора на эту модель позволяет легче организовать процесс обмена и между различными банками данных. Популярность такого подхода основывается на использовании таблиц, как структуры для описания объектов реляционной модели. Структура таблицы определяется совокупностью столбцов. В каждой строке таблицы содержится по одному значению в соответствующем столбце. Столбец соответствует некоторому элементу данных – атрибуту. Каждый столбец имеет имя соответствующего элемента данных (атрибута). Один или несколько атрибутов, значения которых однозначно идентифицируют строку таблицы, называют ключом таблицы. Он и является исходным элементом различных операций с объектами и для организации получения информации из других таблиц. Табличная форма работы с информацией широко используется в различных сферах деятельности, и поэтому такие банки данных облегчают их применение непрофессионалами.

Рассмотрим специфику представления и использования знаний в интеллектуальных системах.

При обработке на ЭВМ данные постоянно трансформируются, последовательно проходя от входной информации как результата измерений и наблюдений, к данным на материальных носителях информации в виде таблиц, протоколов, справочников; структур данных в виде диаграмм, графиков, функций; в компьютере на языках описания данных и т.д. Более высокой формой организации данных является их представление в виде баз знаний.

Знания связаны с данными, основываются на них, но представляют собой результат мыслительной деятельности человека, обобщают его опыт, полученный в

ходе практической деятельности. *Знания – это выявленные закономерности предметной области.* При обработке на ЭВМ знания также трансформируются: от существующих форм в памяти человека как результат обучения, воспитания, мышления к знаниям, помещенным на материальных носителях – учебниках, инструкциях, методических пособиях, книгах и далее к знаниям, описанным на языках их представления для занесения в компьютерные базы знаний.

Знания могут быть классифицированы на *первичные* о видимых взаимосвязях между *отдельными* событиями и фактами в предметной области и *глубинные* – абстракции, аналогии, схемы, отображающие структуру и процессы в предметной области.

Часто знания разделяют на *процедурные* и *декларативные.*

Исторически первичными были процедурные знания, т.е. знания, растворенные в алгоритмах. Для их изменения требовалось изменять программы.

Рассмотрим, например, фрагмент программы на алгоритмическом языке Паскаль.

```
Pi:= 3.14;
R:= 20;
S:= Pi * R * R;
WRITELN ('Площадь круга S =', S).
```

Первые два оператора представляют собой данные, третий оператор – знание. Оно является результатом интеллектуальной деятельности древних геометров и представляет собой закон, выражающий площадь круга через его радиус.

Существуют десятки способов представления декларативных знаний для различных предметных областей. Большинство из них может получаться на основе следующих классов моделей: продукционных; фреймовых; семантических сетей.

Продукционная модель состоит из трех основных компонентов. Первый из них – это база правил типа ЕСЛИ (условие), ТО (действие). ЕСЛИ холодно, ТО надеть шубу; ЕСЛИ идет дождь, ТО взять зонтик и т.п.

Вторым компонентом является рабочая память, в которой хранятся исходные данные к задаче и выводы, полученные в ходе работы системы.

Третий компонент – механизм логического вывода, использующий правила в соответствии с содержимым рабочей памяти. Рассмотрим конкретный пример. В базе правил экспертной системы имеются два правила.

Правило 1: ЕСЛИ «намерение – отдых» и «дорога ухабистая», ТО «использовать джип».

Правило 2: ЕСЛИ «место отдыха – горы», ТО «дорога ухабистая».

Допустим, что в рабочую память поступили исходные данные: «намерение – отдых»; «место отдыха – горы».

Механизм вывода начинает сопоставлять образцы из условных частей правил с образцами, хранимыми в рабочей памяти. Если образцы из условной части имеются в рабочей памяти, то условная часть считается истинной, в противном случае – ложной.

В данном примере при рассмотрении правила 1 оказывается, что образец «намерение – отдых» имеется в рабочей памяти, а образец «дорога ухабистая» отсутствует, поэтому условная часть правила 1 считается ложной. При рассмотрении правила 2 выясняется, что его условная часть истинна. Механизм вывода выполняет заключительную часть этого правила, и образец «дорога ухабистая» заносится в рабочую память. Правило 2 при этом выбывает из числа кандидатов на рассмотрение.

Снова рассматривается правило 1, условная часть которого теперь становится истинной, и содержимое рабочей памяти пополняется образцом «использовать джип». В итоге правил, которые можно было бы применять, не остается и система останавливается.

В рассмотренном примере приведен прямой вывод – от данных к поиску цели. Однако применяют и обратный вывод – от цели для ее подтверждения к данным. Продемонстрируем этот способ на нашем примере. Допустим, что наряду с исходными данными «намерения – отдых»; «место отдыха – горы» имеется цель «использовать джип».

Согласно правилу 1 для достижения этой цели требуется выполнение условия «дорога ухабистая», поэтому условие становится новой целью. При рассмотрении правила 2 оказывается, что условная часть этого правила в данный момент истинна, поэтому рабочая память пополняется образцом «дорога ухабистая». При повторном рассмотрении правила 1 подтверждается цель «использовать джип».

При обратном выводе система останавливается в двух случаях: либо достигается первоначальная цель, либо кончаются правила. При прямом выводе система останавливается только тогда, когда кончаются правила, либо при появлении в рабочей памяти специально предусмотренного образца, например, «использовать джип».

В приведенном примере на каждом этапе прямого вывода можно было использовать только одно правило. В общем же случае на каждом этапе вывода таких правил несколько, и тут возникает проблема выбора. Например, введем в рассмотрение еще одно правило.

Правило 3: ЕСЛИ «намерение – отдых», ТО «нужна скорость».

Кроме того, введем условие останова системы – появление в рабочей памяти образца «использовать джип».

Теперь на первом этапе прямого вывода появляется возможность применять либо правило 2, либо правило 3. Если сначала применить правило 2, то на следующем этапе можно будет применять правило 1 и правило 3. Если на этом этапе применить правило 1, то выполнится условие останова системы, но если прежде применить правило 3, то потребуется еще один этап вывода. Этот пример показывает, что выбор применяемого правила оказывает прямое влияние на эффективность вывода. В реальной системе, где имеется множество правил, появляется проблема их оптимального выбора.

Если на каждом этапе логического вывода существует множество применимых правил, то это множество носит название *конфликтного набора,* а выбор одного из них называется *разрешением конфликта.*

Аналогичная ситуация возникает и при обратном выводе. Например, дополним предыдущий пример еще одним правилом.

Правило 4: ЕСЛИ «место отдыха – пляж», ТО «дорога ухабистая».

Если на основании этого условия подтверждается цель «использовать джип», то для достижения первоначальной цели достаточно применить только одно правило 1; однако, чтобы подтвердить новую цель «дорога ухабистая», открывается возможность применения правила 1, нужно использовать либо правило 2, либо правило 4. Если сначала применить правило 2, то это будет самый удачный выбор, поскольку сразу же можно применить и правило 1. С другой стороны, если попытаться применить правило 2, то, поскольку образца «место отдыха – пляж», который является условием правила 4, в рабочей памяти не существует и, кроме того, не существует правила, подтверждающего его, данный выбор является неудачным. И лишь со второго захода, применяя правило 2, можно подтвердить цель "дорога ухабистая".

Следует обратить внимание на то, что при обратном выводе правило 3, которое не оказывает прямого влияния на достижение цели, не принималось в расчет с самого начала. Таким образом, для обратных выводов характерна тенденция исключения из рассмотрения правил, не имеющих прямого отношения к заданной цели, что позволяет повысить эффективность вывода.

Продукционная модель – это наиболее часто используемый способ представления наний в современных экспертных системах. Основными преимуществами продукционой модели являются наглядность, высокая модульность, легкость внесения изменений дополнений, простота механизма логического вывода.

Следующей эффективной моделью представления знаний является фреймовая одель. Фреймовая модель использовалась первоначально в психологии и философии ля описания понятия абстрактного образа. Например, слово «автомобиль» вызывает у лушающих образ устройства, способного перемещаться, имеющего четыре колеса, сон для шофера и пассажиров, двигатель, руль. Приведенное описание абстрактного браза «автомобиль» является минимальным и из него ничего нельзя убрать без потери го сущности.

Фрейм – это модель абстрактного образа, минимально возможное описание сущости какого-либо объекта, явления, события, ситуации, процесса. Фрейм состоит из мени и отдельных единиц, называемых слотами. Он имеет однородную структуру:

ИМЯ ФРЕЙМА Имя 1-го слота: значение 1-го слота Имя 2-го слота: значение 2-го пота Имя N-го слота: значение *N*-го слота.

В качестве значения слота может выступать имя другого фрейма. Таким образом реймы объединяются в сеть. Свойства фреймов наследуются сверху вниз, т.е. от выестоящих к нижестоящим через АКО-связи (начальные буквы английских слов «A Kind f», что можно перевести как «это»). Слот с именем АКО указывает на имя фрейма боее высокого уровня иерархии.

Например, на рис. 2.5.1. фрейм «Студент» имеет ссылки на вышестоящие фреймы: Человек» и «Млекопитающее». Поэтому на вопрос: «Может ли студент мыслить?» – твет будет положительным, так как этим свойством обладает вышестоящий фрейм Человек».

Если одно и то же свойство указывается в нескольких, связанных между собой, реймах, то приоритет отдается нижестоящему фрейму. Так, возраст фрейма «Студент» не наследуется из вышестоящих фреймов.

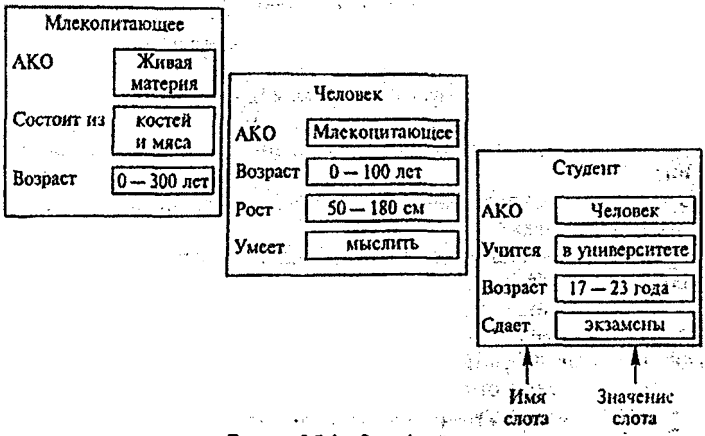


Рисунок 2.5.1 – Сеть фреймов

Основным преимуществом фреймов как способа представления знаний является аглядность и гибкость в употреблении. Кроме того, фреймовая структура согласуется с

современными представлениями о хранении информации в памяти человека.

В основе семантических сетей для представления знаний лежит идея о том, чт любые знания можно представить в виде совокупности *понятий* (объектов) и *отноше ний* (связей). Семантическая сеть представляет собой ориентированный граф, верш нами которого являются понятия, а дугами – отношения между ними. Сам термин «с мантическая» означает смысловая.

Пример семантической сети приведен на рис. 2.5.2.

Основным преимуществом этой модели является наглядность представления зн ний, а также соответствие современным представлениям об организации долговреме ной памяти человека. Недостаток – сложность поиска вывода, а также сложность ко ректировки, т.е. удаления и дополнения сети новыми знаниями.

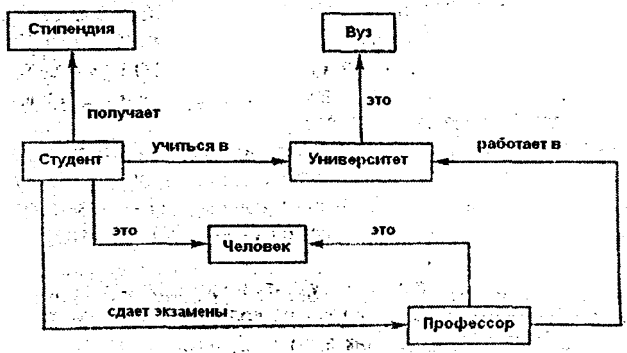


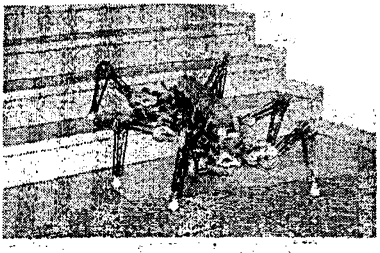
Рисунок 2.5.2 – Семантическая сеть

Литература

[13, 21, 23, 26, 27, 33, 34, 35]

Контрольные вопросы

1. Каковы причины использования формальных языков в вычислительной технике
2. Чем отличаются понятия синтаксис и семантика формального языка?
3. В чем заключается преимущество нотации Бэкуса при ее использовании при описании различных формальных и алгоритмических языков?
4. Зачем используются формальные языки для поиска и смысловой обработки информации?
5. В чем заключаются особенности использования дескрипторных систем с грамматикой и без грамматики?
6. Какие основные этапы необходимо выполнить при разработке алгоритмическог языка и его транслятора?
7. В чем заключается принципиальное отличие трансляторов компилирующего и интерпретирующего типа?
8. Какие основные функции выполняет операционная система ЭВМ?
9. В чем заключается отличие базы данных от базы знаний?
10. Каковы основные функции систем управления базой данных (СУБД)?
11. В чем отличие банка данных от базы данных?
12. Чем отличаются процедурные и декларативные знания?
13. Назовите особенности продукционных моделей вывода.
14. Чем отличаются семантические представления знаний?



Глава 3. СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В РЕШЕНИИ ПРИКЛАДНЫХ ЗАДАЧ

3.1. Проблема создания искусственного интеллекта

В течение многих столетий человек в основном занимался проблемами автоматизации физического труда. Появление современной вычислительной техники, необходимейшего компонента автоматизации умственного труда, послужило могучим толчком к систематическим исследованиям по автоматизации решения творческих задач.

Под влиянием работ А. Тьюринга,Дж. Неймана, Н. Винера, А. Колмогорова, В. М. Глушкова и других в 50-х годах начались интенсивные исследования по вопросам создания искусственного интеллекта.

В первых исследованиях в основном решался принципиальный вопрос о возможности создания технических устройств, обладающих творческими способностями. Вскоре был получен и ряд обнадеживающих результатов прикладного плана при попытках моделирования творческой деятельности человека по управлению различными объектами, проектированию, распознаванию образов, доказательству теорем, игре в шахматы и шашки, сочинению стихов, музыки и т.п.

Развитие робототехники в последние годы привело и к еще более сложным проблемам, связанным с организацией поведения роботов. Эта проблема не возникала, пока специалисты по искусственному интеллекту имели дело с обычной ЭВМ, которая имитировала некоторые виды интеллектуальной деятельности, но не обладала прямым контактом с внешней средой. Появление роботов кардинально изменило ситуацию. Роботы, в отличие от ЭВМ, перемещаются в реальной физической среде, воздействуют с помощью эффекторов на ее объекты, получают от среды с помощью рецепторов различную информацию. Поведение таких систем заинтересовало конструкторов роботов с точки зрения приемлемости поведения подвижной системы искусственного интеллекта. В области искусственного интеллекта началась работа по формальному представлению систем процедур, обеспечивающих заданное нормативное поведение при заданных знаниях о внешней среде и целях функционирования системы. Это привело к развитию идей, связанных с нахождением эффективных систем представления знаний и планирования целесообразной деятельности. В остальных аспектах продвижение работ в области робототехники существенно опирается на исследования в области машинного интеллекта.

Принципиальная возможность реализации различных алгоритмизируемых процессов на вычислительной машине была доказана еще английским математиком А. Тьюрингом. А. Тьюринг развивал функциональный подход к оценке деятельности автоматов. Такой подход нашел сторонников при решении вопросов о возможности машинного мышления.

Если определить мышление как нечто свойственное только человеку, то ответ н вопрос о возможности машинного мышления будет отрицательным. Если же стать н позиции оценки «деятельности» машины по конечным результатам, то ответ будет п ложительным, что хорошо иллюстрируется следующим опытом (рис. 3.1.1).

Имеются три комнаты. В первой комнате находится человек (экспериментатор), в второй – машина, в третьей – человек. Экспериментатор точно не знает, в какой комнат находится человек, а в какой – машина. Ему предлагается задать конечную серию зада

Рисунок 3.1.1 – Схема эксперимента Тьюринга

(вопросов) и определить по ответам, в какой комнате принимаются решения человеко а в какой – машиной. Естественно признать, что машина может «мыслить», если эксп риментатор однозначно не может указать, где находится машина. Простейшими прим рами для эксперимента могут послужить решение шахматных этюдов или задач, прое тирование режущего инструмента и т.д.

Работы по машинному интеллекту ведутся в двух направлениях:

1) моделирование деятельности мозга на ЭВМ;

2) поиск алгоритмов для решения различных творческих задач.

Исследователи, придерживающиеся первого направления, идут по двум путям. одной стороны, они стремятся детально изучить функции и свойства элементарно ячейки нервной системы – нейрона, а с другой – научиться «собирать» из этих яче сложные конструкции по типу человеческого мозга. На этом пути пока не удалось до тигнуть существенных в прикладном отношении результатов, так как имеются трудно как с описанием самой элементарной ячейки, так и с поиском закономерностей объед нения их в сети. Имитировать же эффективно мозг с помощью случайно скоммутирова ной сети маловероятно. Другой путь связан с моделированием работы мозга на уров информационных процессов, добиваясь того, чтобы функции модели были по возмо ности неотличимы от функции моделируемого объекта.

Второе направление представлено серией работ по искусственному интеллект представляющих как теоретический, так и практический интерес. Среди них можно вы делить ряд результатов, полученных с помощью методов ситуационного управления эвристического программирования.

Методы ситуационного управления, предложенные Д. А. Поспеловым и Ю. А. Кл ковым, ориентированы на те случаи, когда формальное описание модели объекта мет дами классической математики нецелесообразно или же невозможно. Взамен традиц онных подходов в ситуационном управлении строится семиотическая модель объекта протекающих в нем процессов. Описание модели базируется на естественном язык Для процессов проектирования и управления такой подход весьма перспективен.

В этих случаях указанный метод позволяет, используя опыт специалиста-человек построить модель описания объекта и создать схему принятия необходимых решений.

В целом ситуационное управление дает с точки зрения языка описания единоо разный процесс как для построения модели объекта, так и для управления им. Одна решение задач этим методом из-за отсутствия специального математического обесп

ения достаточно трудоемко и качество получаемых решений определяется квалифика-ией специалистов, привлекаемых для формирования моделей объекта и управления им.

Мы детальнее остановимся на той ветви второго направления работ по искусст-енному интеллекту, которое широко известно как эвристическое программирование, и становимся главным образом на его прикладных аспектах, так как в недрах эвристиче-кого программирования зародились идеи создания универсальной программы для ре-ения любых задач.

По вопросам создания искусственного интеллекта насчитывается сотни тысяч пуб-икаций. И тем не менее при решении конкретных прикладных проблем не всегда уда-тся эффективно использовать эту информацию. Обычно статьи содержат общие реко-ендации по построению программ и не имеют описаний конкретных приемов, исполь-ованных данным автором.

Современное эвристическое программирование занимается исследованием типо-ых приемов, полезных в процессах решения проблем. Эвристика строится на основе аблюдений за тем, как люди решают задачи, что есть общего в решении самых различ-ых проблем.

До начала решения задачи надо быть уверенным, что она поставлена. В современ-ой эвристике рассматриваются лишь некоторые методы решения задач, но не иссле-уются вопросы, связанные с постановкой проблемы. Методы эвристического програм-ирования в основном направлены на ограничение области поиска решения за счет ис-ользования опыта решения данного класса задач или выдвижения некоторых гипотез.

Прежде чем приступить к решению, надо обладать методом распознавания удов-етворительного ответа. Иными словами, за конечное число шагов требуется устано-ить, является ли полученный ответ удовлетворительным. С этой точки зрения, крайне елательно иметь эффективный алгоритм, гарантирующий получение решения, если но существует в пределах разумных затрат времени.

Анализ эффективности эвристических программ показывает, что для решения ложных задач необходимо иметь в составе программ стандартные процедуры для существления поиска, распознавания, обучения, планирования и индуктивного вывода. оиск заключается в проверке пригодности всех предлагаемых решений. Однако подоб-ый путь не представляет практического интереса из-за больших затрат времени. бычно почти в каждой интересующей нас задаче можно оценивать некоторые предва-ительные решения и делать следующие попытки в более перспективном направлении.

Процедура распознавания образов позволяет не исследовать все возможности для ганизации поиска, чтобы сразу применить к конкретной проблеме наиболее эффек-ивный метод. В связи с этим возникает задача классификации методов и распознава-ия соответствующих им ситуаций. Наиболее простой путь в этом отношении – сравне-ие неизвестного объекта с рядом эталонов. На практике оказывается, что число этало-ов довольно велико. Поэтому пытаются истолковать образ как систему в некотором мысле подобных объектов. Возникает задача выявления хорошей системы признаков, писывающих образы.

При наличии такой системы подпрограмма распознавания подвергает исходные анные последовательным испытаниям на присутствие различных признаков. После че-решается вопрос о принадлежности исследуемого объекта к одному из образов в со-ветствии с весом выявленных признаков. Поэтому основной задачей многих эвристи-еских программ является отыскание хорошей системы признаков, инвариантных отно-ительно различных преобразований.

Обучение позволяет при решении новых задач использовать методы, оказавши эффективными при решении аналогичных проблем. Такую эвристику обычно реализ на основе моделей, повышающих ее приоритет при успешном применении. Любая чающая система должна использовать результаты прошлого опыта как основу для лее общих предположений. Простейшим способом обобщения внутри множества знаков является построение «типичного» члена данного множества (т.е. усреднен Все же возможности системы с простым повышением приоритета ограничены ее з симостью от «учителя» (тренировочной последовательностью задач). Один из п преодоления этой трудности состоит в разработке процедуры обобщения дейст «учителя». Тогда при решении задач машина будет сама повышать приоритет эври в процессе работы.

Процедура планирования служит для выбора перспективной цепочки подзада ходе решения проблемы приходится сталкиваться с множеством взаимосвязанных задач. Выбор подзадачи должен основываться на относительных оценках трудноС которые встречаются при ее исследовании, и на оценках, характеризующих важн подзадачи для решения всей проблемы.

При решении сложных задач применение таких «пошаговых» эвристик не дает зультатов. Машина должна обладать способностью к анализу структуры проблемы в лом, т.е. способностью планирования. На практике всякая возможность планирова оказывается полезной, если задача достаточно трудна. Лишь от схем, которые акти продолжают анализ для выработки набора цепочек, можно ожидать, что они смогут ваивать решение задач все возрастающей сложности.

Желательно сообщить машине способность к индуктивному мышлению, т.е. о печить ее методами, которые можно использовать для построения общих утвержден событиях, не встречавшихся ранее. Однако может оказаться, что не существует си мы индуктивного вывода, которая работала бы во всевозможных средах. Все же сч ется, если задана среда и критерий успеха, то эта проблема для машины является то технической. Отметим, что в настоящее время не имеется пока программ с разви индуктивным мышлением.

Все пять видов процедур в той или иной степени были использованы впервы США в 60-х годах Ньюэллом, Шоу и Саймоном в программе «Общий решатель зад предназначенной для решения широкого класса задач (доказательство теорем, инте рование с использованием табличных интегралов, игровые задачи и др.). Они осозн необходимость развития новой теории, опирающейся на имитацию процедур уни сального характера, с помощью которых порождались бы конкретные процедуры, правленные на решение определенных задач – задача интеллектуального типа.

В качестве основной ими была взята процедура, имитирующая поиск по лабир возможных альтернатив. Авторы программы выдвинули тезис, что решение любой дачи человеком состоит в поиске пути, приводящего от начальной площадки лабири соответствующей исходному описанию ситуаций, к некоторой конечной площадке, с ветствующей решению этой задачи. На каждом шаге поиска пути используются лок ные критерии успеха, дающие возможность выбора очередного коридора лабири Авторы вначале были уверены, что с помощью найденной ими процедуры удается шить все интеллектуальные задачи. Первыми успешными экспериментами в этом правлении были доказательство теорем исчисления высказываний и игровые прог мы. Но проверка универсальности процедуры поиска по лабиринту на шахматной

грамме показала ее авторам, что, основываясь лишь на ней, практически невозможно решить сколько-нибудь серьезную задачу.

В прикладном плане, по-видимому, в настоящее время будут иметь успех методы, сочетающие точный и приближенный путь решения задач, т.е. такие методы, где сокращение объема вычислений по точному методу будет вестись за счет использования эффективных эвристик, учитывающих особенности данного класса задач.

Естественно, что вопросы автоматизации творческих процессов занимают одно из центральных мест в современной кибернетике. Среди них наиболее важным является вопрос о теоретических и практических границах, в пределах которых возможна автоматизация этих процессов.

С практической точки зрения, границы автоматизации определяются состоянием теории и соответствующей области деятельности и возможностями современных вычислительных машин.

Творчеством обычно называется целенаправленная деятельность человека, создающая новые материальные и духовные ценности, обладающие общественной значимостью. Такое определение затруднительно применить к результатам, полученным на ЭВМ, если считать оригинальными только те из них, которые являются новыми для человечества в целом.

Дискуссия ученых о том, какие системы считать интеллектуальными, не привела к формированию единой точки зрения. Ряд ученых уходит от прямого определения и относит к таковым системы, играющие в шахматы и шашки, сочиняющие стихи и музыку, распознающие речь и образы, выполняющие автоматический перевод, доказывающие теоремы и др. Однако все существующие конструктивные определения распадаются на две группы: в первой группе перечисляются свойства интеллектуальной системы, а во второй – минимальный уровень выполнения некоторого списка функций.

А. Эндрю искусственный интеллект определяет как область исследований, направленных на то, чтобы заставить машины выполнять трудные для них функции, которые способны сегодня выполнять только люди. Такое определение является «скользящим», т.е. меняется со временем, так как функции человека по мере познания мира постоянно расширяются.

Не останавливаясь детально на философских аспектах этой проблемы, будем считать, что процесс получения любого результата всегда расчленяется на несколько этапов, каждый из которых может полностью или частично выполняться вычислительной машиной. Если аналогичные результаты, полученные человеком на каком-то этапе или на нескольких автоматизированных этапах, квалифицируются как творческие, то будем и машинный результат относить к разряду творческих. При невозможности автоматизировать весь процесс, можно остановиться на автоматизации лишь его рутинных этапов. Однако во всех случаях машиной будет получаться либо творческий результат, либо использование машины поможет человеку прийти к этому результату. На наш взгляд, в практике не существует резкой границы при классификации задач на творческие и рутинные. Всегда разумно говорить о сложности той или иной конкретной задачи и необходимом уровне творчества для ее решения, т.е. считать, что творчество носит многоуровневый характер. Термин «машина» становится более расплывчатым, так как многие процессы могут выполняться программно и аппаратурно. Во многих случаях даже удобно под «машиной» понимать программу для реализации процесса. Возникает ряд новых задач, связанных с рациональным разбиением алгоритма на части, выполняемые человеком и машиной, а также с вопросами эффективного обмена информацией между че-

ловеком и машиной. От активного вмешательства человека в процесс выполнения машиной программ во многом определяется качество результата и скорость его получения.

Поэтому в ближайшем будущем следует ориентироваться на участие человека в системах проектирования, управления и других в качестве активного звена с достаточно большой нагрузкой в части оценки текущей ситуации на определенном этапе при решении задачи и принятия различных логических решений. Использование специалистами на своих рабочих местах удаленных от ЭВМ терминалов или персональных ЭВМ позволит использовать машину в качестве основного инструмента для выполнения практически любых творческих работ. Однако эффективность использования этого инструмента будет определяться полнотой и качеством системы из отдельных модулей, ориентированных на решение частных вопросов, и наличием проблемно-ориентированного входного языка, удобного для пользователей.

Проблема создания и использования диалоговых систем для достижения решения лучших или эквивалентных по сравнению с получаемыми традиционным путем требует отдельного рассмотрения.

3.2. Диалоговые методы решения задач, сети ЭВМ и экспертные системы

Решение проблемы создания искусственного интеллекта в различных областях деятельности человека, как отмечалось выше, еще долгие годы будет идти по пути вскрытия универсальных механизмов принятия решений, направленных на автоматическое сужение области поиска результатов в соответствии с конкретной исходной ситуацией. Тем не менее человеку приходится на практике решать целый ряд задач, которые требуют использования вычислительных ресурсов. В этом случае становится логичным использование самого человека в качестве звена автоматизированной системы, что порождает проблему обеспечения эффективного общения человека с ЭВМ в диалоговом режиме.

Средства общения могут развиваться в следующих основных направлениях: создание общего программного обеспечения для манипулирования с различными объектами в памяти ЭВМ на подмножествах естественного языка, опираясь на внешние устройства ЭВМ (дисплей, клавиатуру, звуковой ввод информации и т.п.) и специальное программное обеспечение, рассчитанное на использование средств естественного языка и графику для выполнения некоторой части процесса решения конкретной задачи человеком.

Оба направления существенно опираются на программное и техническое обеспечение диалога на уровне простейших операций, но построение программ в области автоматизации сложных процессов немыслимо на базе только элементарных действий, так как решение задачи сводится к большим затратам времени вплоть до утраты разумных границ временных ограничений на получение ответа.

В таких случаях приходится строить программы, включающие языки манипулирования объектами на уровне некоторых законченных операций в специальных областях или для некоторых классов задач на содержательном уровне.

Сложная задача может включать фрагменты из смежных областей для данного специалиста, тогда с его стороны логично обратиться за консультацией к соответствующему специалисту или же к специализированной автоматизированной «экспертной» системе. Решение таких задач в реальном масштабе времени должно опираться на эффективные средства связи как людей, так и ЭВМ.

Диалоговый режим открывает новые возможности и в осуществлении моделирования (вычислительного эксперимента). Некоторый объект можно представить в виде ряда

лементов, формальное описание которых позволяет имитировать их поведение на ВМ. В результате имитации поведения среды можно проверить качество будущей конрукции без ее физической реализации и тем самым экономить средства, затрачиваеые на неудачные экспериментальные образцы. Наличие моделей для различных элеентов позволяет инженеру в диалоговом режиме путем подбора построить модель бъекта из нужного набора элементов, обладающую искомыми свойствами.

Например, при решении аэродинамической задачи о входе космического аппарата атмосферу Земли необходимо рассчитать силовые нагрузки, тепловой режим и т.д. олучить такие данные в лаборатории чрезвычайно сложно, так как необходимо создать ответствующие условия по температурам и скоростям, что практически приведет к оспроизведению самого явления.

Поэтому в лаборатории изучают отдельные элементы явлений, а на их основе троится математическая модель для выполнения вычислительного эксперимента, коорый позволяет сэкономить миллионы рублей.

Использование при решении задач проектирования больших банков данных, в коорых хранятся типовые решения и элементы, поможет обеспечить получение новых онструкций из стандартных элементов, а также быстро учитывать в создаваемых конрукциях новинки, разработанные в других организациях и записанные в общий банк анных.

Создание больших информационно-справочных систем и развитие средств связи озволит любому предприятию передавать заказы на решение не свойственных ему заач специализированной проектной организации или выполнять их на крупных кустовых втоматизированных проектных центрах, обладая лишь средствами связи и необходиой периферийной аппаратурой для получения результатов от ЭВМ.

Одновременное использование при решении сложных задач ряда специалистов в азличных областях знаний позволит применить системный подход к решению проемных вопросов. Отдельно взятый специалист (физик, инженер, математик, врач и .д.) изучает человека или иной сложный объект очень хорошо в пределах своей подгоовки, но на современном уровне знаний принятие решения в одной области или же по дному узкому вопросу требует анализа влияния последствий на весь организм или систему в целом и связанные с ними объекты. Электронная вычислительная машина в таих случаях может стать тем звеном, которое в диалоговом режиме поможет группе спеиалистов, находящихся в разных местах, прийти к эффективному решению.

Возможности машинного эксперимента по сравнению с натурным очень большие. он легко управляем, дешев, позволяет воспроизвести условия, которые трудно создать лаборатории, дает полную информацию о процессах, протекающих в модели.

Например, проблемы загрязнения среды, развития животного и растительного миа, осушения и обводнения, развития городов и поселков разумно решать с участием ногих специалистов и путем широкого машинного эксперимента.

Упомянутые машинные эксперименты приводят к естественной постановке задачи связи отдельных ЭВМ в сети для организации решения сложных проблем.

Каждый из участников решения задачи мог бы изучать свои аспекты проблемы и бмениваться своей информацией с другими участниками, а от них брать информацию ля корректировок своих моделей процессов и их параметров. Аналогичные задачи в олее простой постановке возникают при нехватке информации или ресурсов для решеия узких задач.

Взаимодействие процессов, протекающих в сетях из разнородных ЭВМ, кром процедур обмена информацией, порождает еще массу проблем, связанных с достове ностью передачи данных и их защитой от постороннего вмешательства.

Для рядового пользователя ЭВМ – специалиста только в своей области – сгусто всех этих специфических вопросов постепенно стал непреодолимой стеной на пути и пользования вычислительной техники.

Поэтому проблему общения с ЭВМ на подмножествах естественного языка пр шлось выделить как самостоятельную.

ЭВМ, соединенные в сети и хранящие информацию и тексты различных програм в узлах сети, придают новое качество функциональным возможностям специалиста оперативность в решении сложных задач. Он не заменяется техникой, а лишь его ус лия перемещаются от рутинной работы в более творческие области благодаря перви ной обработке и пересылке информации без его участия.

На схему работы человека существенно должно повлиять внедрение персонал ных микро-ЭВМ, которые допускают как автономное решение задач, так и использов ние их как компонент в локальных сетях ЭВМ. Такие ЭВМ используются в автоматиз ции подготовки данных и как «интеллектуальные» терминалы для пересылки информ ции, и как средства обращения за помощью в решении задач к другим ЭВМ сети.

Такие сети ЭВМ могут использоваться для безбумажной обработки информаци благодаря простым средствам обмена ею в «электронной» форме между людьми, обл дающими персональными ЭВМ. Более того, развиваются идеи о проведении телеко ференций на основе сетей ЭВМ и о перенесении рабочих мест сотрудников из учрежд ний в домашние условия. Таким образом, диалоговые методы решения задач по мер развития научно-технического прогресса получают новые эффективные возможнос для обработки, хранения и обмена информацией. В частности, хорошие результаты ряде специальных областей деятельности человека получили с использованием эк пертных систем, построенных с использованием знаний высококвалифицированны специалистов.

Знания, которыми обладает специалист в какой-либо области, можно разделить формализуемые и плохо формализуемые. Формализуемые знания излагаются в книг и руководствах в виде законов, формул, моделей, алгоритмов. Формализуемые знан характерны для точных наук, таких как математика, физика, химия, астрономия. Нау которые принято называть описательными, обычно оперируют с плохо формализуем ми знаниями. К таким наукам можно отнести, например, зоологию, ботанику, экологи социологию, педагогику, медицину и др.

Существуют неформализуемые знания, которые вообще не попадают в книги и р ководства в связи с их неконкретностью, субъективностью, приблизительностью. Знан этого рода являются результатом многолетних наблюдений, опыта работы, интуици Они обычно представляют собой множество эмпирических и эвристических приемов правил. Такие знания передаются из поколения в поколение в виде определенных нав ков, ноу-хау, секретов ремесла. Есть также знания, которые не могут быть выражены в математическом виде, ни в терминах обычного человеческого языка. Такими знания обладают экстрасенсы, контактеры, шаманы.

Класс задач, относящихся к неформализуемым и плохо формализуемым знания значительно больше класса задач, для которых знания формализуемы. Этим объясн ется особая популярность и широкое практическое применение экспертных систем,

торые открыли возможность применения компьютерных технологий в предметных областях, в которых знания плохо формализуемы.

Экспертные системы – это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие эти знания для консультаций менее квалифицированных пользователей.

При решении различных задач иногда приходится обращаться к другим специалистам или системам, образно говоря, привлекать экспертов. В последнее время интенсивно ведется разработка таких программ, которые обеспечивают экспертизу в различных областях знаний: управлении производством, медицине и т.д.

Экспертная система должна содержать существенную часть знаний эксперта-человека в конкретной области и использовать накопленную информацию подобно человеку. Например, экспертная система в области медицины должна уметь ставить диагноз на основе анализа симптомов пациента. Из такого класса программ наиболее интересна в прикладном плане система МИЦИН, созданная в Стэнфордском университете в США, и ее последующая версия ТЕЙРЕСИАС. Их успех у практикующих медиков в том, что они помогают врачу поставить правильный диагноз на основании рассуждений типа: «Если..., то можно предположить, что...». Программа дает врачу численную оценку вероятности каждого из заключений. За словами «можно предположить», например, стоит вероятностная оценка 0,1, а за словом «вероятно» оценка 0,8 и т.д. Программа также указывает, какие дополнительные наблюдения и тесты могут снизить неопределенность ответа.

Главная особенность этих программ в том, что они способны на языке, близком к естественному, объяснить, как было выработано решение. Программа это делает на основе записи шагов, по которым можно объяснить, почему та или иная возможность была исключена из рассмотрения.

Способность программы давать объяснения сыграла существенную роль в признании системы практикующими медиками. Конечная ответственность за принятый диагноз, конечно, лежит на враче. Естественно, как упоминалось выше, врач, работающий в контакте с вычислительной машиной, будет иметь преимущество, так как в худшем случае он примет свое решение, а в лучшем воспользуется диагнозом машины или примет дополнительное решение изучить глубже на основе точных анализов сложившуюся ситуацию.

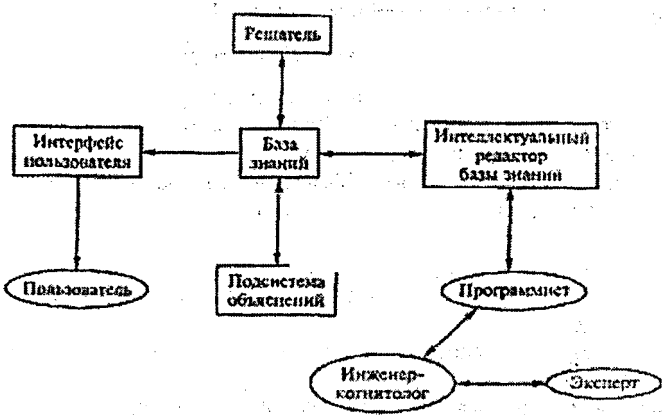


Рисунок 3.2.1 – Типичная блок-схема экспертной системы

Обобщенная блок-схема экспертной системы представлена на рис. 3.2.1.Обычно в ее состав входят следующие взаимосвязанные между собой модули:

база знаний – ядро экспертной системы, совокупность знаний предметной области, записанная на машинном носителе в форме, понятной эксперту и пользователю;

интеллектуальный редактор базы знаний – программа, представляющая инженеру-когнитологу и программисту возможность создавать базу знаний в диалоговом режиме. Она включает в себя системы вложенных меню, шаблонов языка представления знаний, подсказок (help-режим) и других сервисных средств, облегчающих работу с базой знаний;

интерфейс пользователя – комплекс программ, реализующих диалог пользователя с экспертной системой на стадии как ввода информации, так и получения результатов;

решатель (синонимы; дедуктивная машина, блок логического вывода) – программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в базе знаний;

подсистема объяснений – программа, позволяющая пользователю получать ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?». Ответ на вопрос «Как?» – это трассировка всего процесса получения решения с указанием исполняющих фрагментов базы знаний, т.е. всех шагов цепи умозаключений. Ответ на вопрос «Почему?» – ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад.

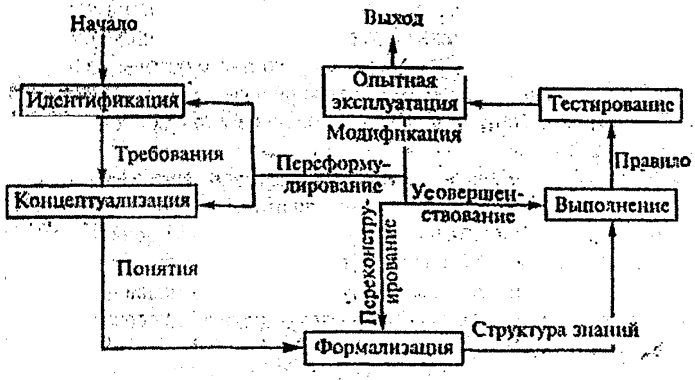


Рисунок 3.2.2 – Технология разработки экспертной системы

В коллектив разработчиков экспертной системы входят как минимум четыре специалиста (или четыре группы специалистов): эксперт, инженер-когнитолог, программист, пользователь. Возглавляет коллектив инженер-когнитолог – ключевая фигура при разработке систем, основанных на знаниях. Обычно это руководитель проекта, в задачу которого входит организация всего процесса создания экспертной системы. С одной стороны, он должен быть специалистом в области искусственного интеллекта, а с другой – разбираться в предметной области, общаться с экспертом, извлекая и формализуя его знания, передавать их программисту, кодирующему и помещающему их в базу знаний экспертной системы.

Экспертная система работает в двух режимах – приобретения знаний и решения задач или консультаций.

В режиме приобретения знаний происходит формирование базы знаний. В режиме решения задач общение с экспертной системой осуществляет конечный пользователь.

Обычно знания, которыми располагает эксперт, различаются степенью надежно-, важности, четкости. В этом случае они снабжаются некоторыми весовыми коэффи-ентами, которые называют коэффициентами доверия. Такие знания обрабатываются ющью алгоритмов нечеткой математики.

В процессе опытной эксплуатации коэффициенты доверия могут подвергаться кор-ктировке. В этом случае говорят, что происходит обучение экспертной системы. Про-сс обучения экспертной системы может производиться автоматически с помощью учающего алгоритма либо путем вмешательства инженера-когнитолога, выполняюще-роль учителя.

В процессе разработки экспертные системы проходят определенные стадии, в ре-льтате которых создаются различные версии, называемые прототилами:

демонстрационный прототип – экспертная система, которая решает часть требуемых дач, демонстрируя жизнеспособность метода инженерии знаний. Работает, имея в базе аний всего 50...100 правил. Время разработки такой экспертной системы – 6...12 мес.;

исследовательский прототип – экспертная система, которая решает все требуемые дачи, но неустойчива в работе и не полностью проверена. База знаний содержит 0...500 правил. Разработка занимает 3...6 мес.;

действующий прототип – надежно решает все задачи, но для решения сложных за-ч может потребоваться много времени и памяти. База знаний содержит 500... 1000 авил. Время разработки – 6...12 мес.;

промышленная экспертная система – обеспечивает высокое качество решения ех задач при минимуме времени и памяти, что достигается переписыванием программ использованием более совершенных инструментальных средств и языков низкого овня. База знаний содержит 1000... 1500 правил. Время разработки – 1...1,5 года;

коммерческая экспертная система – отличается от промышленной тем, что помимо бственного использования она может продаваться различным потребителям. База аний содержит 1500... 3000 правил. Время разработки – 1,5...3 года. Стоимость – 3...5 млн. долларов.

В настоящее время уже сложилась определенная технология разработки экспертных истем, которая состоит из следующих этапов, схематично изображенных на рис. 3.2.2:

1. Идентификация (постановка задачи). На этапе устанавливаются задачи, которые одлежат решению, выявляются цели разработки, требования к экспертной системе, ре-рсы, используемые понятия и их взаимосвязи, определяются методы решения задач. ель этапа – сформулировать задачу, охарактеризовать поддерживающую ее базу зна-ий и таким образом обеспечить начальный импульс для развития базы знаний.

2. Концептуализация. Проводится содержательный анализ проблемной области, вы-ляются используемые понятия и их взаимосвязи, определяются методы решения задач.

3. Формализация. Определяются способы представления всех видов знаний, фор-ализуются основные понятия, определяются способы интерпретации знаний, оценива-тся адекватность целям системы зафиксированных понятий, методов решения, редств представления и манипулирования знаниями.

4. Выполнение. Осуществляется наполнение экспертом базы знаний. Процесс при-бретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, беспечивающую эффективную работу системы, и представление знаний в виде, понят-ом экспертной системе. Из-за эвристического характера знаний их приобретение явля-тся весьма трудоемким.

5. Тестирование. Эксперт и инженер по знаниям в интерактивном режиме, испо зуя диалоговые и объяснительные средства, проверяют компетентность экспертной темы. Процесс тестирования продолжается до тех пор, пока эксперт не решит, что тема достигла требуемого уровня компетентности.

6. Опытная эксплуатация. Проверяется пригодность экспертной системы для нечных пользователей. По результатам этого этапа может потребоваться модифика экспертной системы.

7. Модификация. В ходе создания экспертной системы почти постоянно прои дится ее модификация: переформулирование понятий и требований, переконструир ние представления знаний и усовершенствование прототипа.

Усовершенствование прототипа осуществляется в процессе циклического про дения через этапы выполнения и тестирования для отладки правил и процедур выво,

Переконструирование выбранного ранее способа представления знаний предп гает возврат с этапа тестирования на этап формализации.

Если возникшие проблемы еще более серьезны, то после неудачи на этапе те рования может потребоваться возврат на этап концептуализации и идентификаци этом случае речь идет о переформулировании понятий, используемых в системе, перепроектировании системы заново.

3.3. Нейронные сети и нейрокомпьютеры

Интеллект характеризует способность в процессе мышления к генерировани выбору способа действий, адекватно отражающих решаемую проблему. Естествен интеллект возник и развивается в процессе биологической эволюции с целью адапта к внешней среде. Он присущ в той или иной мере биологическим формам жизни. Ис ственный интеллект характеризует способность к мышлению искусственных систем. опирается на биологические основы естественного интеллекта и пытается в той иной мере моделировать мыслительные процессы живых существ. В результате ра тия систем искусственного интеллекта возник термин "искусственная жизнь" (Artif Life), который отражает различные аспекты поведения и взаимодействия искусствен систем. Нейронная организация и механизмы обучения искусственных систем, проце их самоорганизации развития родились на стыке наук.

Человеческий мозг является самой сложной в обозримом мире структурой, кото является результатом многих миллионов лет эволюции. В отличие от компьютера создавался без жестко определенной схемы, а в результате естественного отбор главной движущей силы эволюции. Способность человеческого мозга к обучению, за минанию сложной информации и логическому мышлению с давних пор привлека нему философов, нейробиологов, психологов и кибернетиков. В прошлом многие, пример, Декарт и др., представляли психику как нечто нематериальное. Сейчас бо шинство исследователей сходятся к мысли, что психику можно объяснить материа стически, как результат взаимодействия клеток мозга. До недавнего времени предста тели различных наук работали изолированно. Так, нейробиология – наука о мозге, чала функционирование мозга на молекулярном уровне. Здесь она использует прин "от молекулы к разуму". Психология долгое время находилась под влиянием бихе ризма и исследовала мозг как "черный ящик", основываясь лишь на наблюдении по дения без учета морфологии мозга. В противовес этому в середине 50-х годов возни когнитивная психология (наука о разуме), которая дала возможность анализировать

хические процессы. кибернетика, используя теорию искусственных неиронных сетеи и опираясь на достижения нейробиологии и когнитивной психологии, исследует искусственные системы, которые позволяют воспроизводить мыслительные процессы биологических существ.

В настоящее время происходит интеграция этих наук. Так, когнитивная нейробиология пытается выяснить, как психические события скоррелированы с электрическими сигналами в мозге, и описать высшие психические функции в терминах точных наук.

Теория искусственных нейронных сетей, моделируя в той или иной степени мыслительные процессы, позволяет глубже понять функционирование мозга. Задача объединения этих наук состоит в том, чтобы описывать высшие психические функции как скоординированную активацию нейронов в коре головного мозга. Это создает потенциальные предпосылки для создания нейрокомпьютеров, которые являются новым этапом в эволюции вычислительной техники.

Головной мозг человека весит от 1.3 до 1.8 кг и содержит триллион клеток, из которых 100 млрд. представлены соединенными в сети нейронами. По порядку величины это сравнимо с числом звезд в Млечном Пути. Нейроны, соединенные разнообразными связями в сеть, определяют интеллект, творческие способности и память человека. Количество нейронов в головном мозге человека больше, чем у всех остальных известных форм жизни. Нейрон представляет собой особый вид клеток, которые обладают электрической активностью. Он получает информацию (рис.3.3.1) при помощи сильно разветвленных отростков, называемых дендритами, и передает информацию вдоль тонкого волокна, которое называется аксоном.

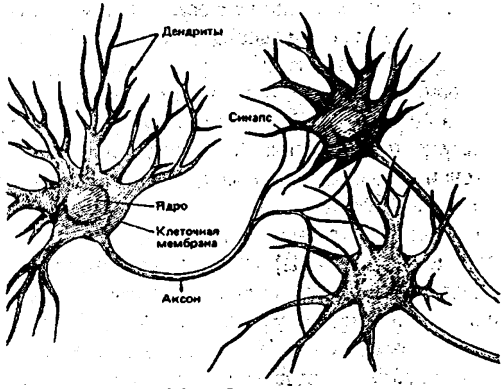


Рисунок 3.3.1 – Биологический нейрон

Аксон имеет множество ответвлений, на конце каждого из которых находится область, называемая синапсом. Посредством синапсов осуществляется связь между различными нейронами. Каждый нейрон может иметь тысячи связей с соседними нейронами. Информация по аксонам передается в виде коротких электрических импульсов, амплитуда которых составляет около 100 мВ, а длительность – 1 мс. На участках контакта между нейронами (синапсы) электрические импульсы превращаются в химические сигналы, которые стимулируют проникновение в клетку нейрона положительных зарядов. Когда достигается некоторое критическое значение потенциала, называемое пороговым, возникает электрический импульс, который как волна распространяется по аксону на

следующий нейрон. Вклад одного синапса в установление соответствующего потенциа ла на выходе нейрона очень маленький. Для возникновения электрического импульс необходимо, чтобы нейрон непрерывно интегрировал множество синаптических входов

При этом такая интеграция является нелинейным преобразованием и не соотве ствует простой операции линейного суммирования. Использование технологии окрас нейронов солями серебра позволяет выявить большое разнообразие типов нейронов коре головного мозга. Так, существуют пирамидальные нейроны, нейроны Таламус нейроны Пуркинье и т.д., всего около 50 типов. Из этого следует, что не все компоненты из которых построен мозг, взаимозаменяемы.

Скорость распространения нервного импульса в аксоне составляет приблизител но 100 м/с, что в миллион раз меньше скорости распространения электрического сигн ла по медной проволоке. Однако параллельная обработка нейронами информации, к торая одновременно распространяется по множеству связей, компенсирует этот недо таток.

Таким образом, в процессе психической деятельности, в коре головного мозга ра пространяются нервные импульсы, которые активизируют соответствующие област нейронов. Совокупность нейронов и связей между ними образуют нейронную сеть (НС от функционирования которой зависят эмоциональные реакции, сознательная деятел ность и память человека. По аналогии с нейросетями создаются и искусственные не ронные системы.

Интерес к нейроинтеллекту возник еще на заре развития вычислительной техник В его основе лежит нейронная организация искусственных систем, которая имеет биол гические предпосылки. Способность биологических систем к обучению, самоорганиз ции и адаптации имеет большое преимущество по сравнению с современными вычисл тельными системами. Достоинством компьютерных систем является большая скорос распространения информации и возможность учета большого объема знаний, накопле ных человечеством в этой области. Разработка искусственных разумных систем, кот рые соединяют преимущества биологических существ и современной вычислительно техники, создает потенциальные предпосылки для перехода к качественно новому эта эволюции в вычислительной технике.

Первые шаги в области искусственных нейронных сетей сделали в 1943 г. В. Ма Калох (W. MCculloch) и В. Питс (W. Pitts). Они показали, что при помощи пороговых не ронных элементов можно реализовать исчисление любых логических функций. В 1949 Дональд Хебб предложил правило обучения, которое стало математической основ для обучения ряда нейронных сетей. В 1957-1962 годах Ф. Розенблатт предложил и и следовал модель нейронной сети, которую он назвал персептроном. Результаты иссл дований он обобщил в книге "Принципы нейродинамики", которая имеет большое знач ние для развития нейронных сетей. В 1959 г. В.Видроу (W. Widrow) и М. Хофф (M. Ho предложили процедуру обучения для линейного адаптивного элемента, который они н звали "ADALINE". Процедура обучения получила название "дельта правило". В 1969 М. Минский (M. Minsky) и С. Пайперт (S. Papert) опубликовали монографию "Персептр ны", в которой осуществили математический анализ персептрона и показали ограничен присущие ему. Выводы их были довольно пессимистичными, и это сыграло негативну роль для дальнейшего развития исследований в области нейронных сетей. Работы в о ласти персептронных сетей были практически приостановлены. В 70-е годы появился р работ в области ассоциативной памяти. Так, Андерсон (Anderson) предложил в 1977

модель линейной ассоциативной памяти. В этом направлении продолжил исследования Т. Кохонен (T. Kohonen), который предложил модель оптимальной линейной ассоциативной памяти. В 1976 г. С. Гроссберг (S. Grossberg) разработал теорию адаптивного резонанса, которая может быть использована для построения ассоциативной памяти.

В 80-е годы происходит значительное усиление интенсивности исследований в области нейронных сетей. Джон Хопфилд (John Hopfield) в 1982 году произвел анализ устойчивости нейронных сетей с обратными связями и предложил использовать их для решения задач оптимизации. Тео Кохонен разработал и исследовал самоорганизующиеся нейронные сети. Ряд авторов (Rumelhart, Hinton, Wiliams) предложил и алгоритм обратного распространения ошибки, который стал мощным средством для обучения многослойных нейронных сетей. В 1987 году под эгидой общества IEEE (Institute of Electrical and Electronic Enginneer's) проводится первая международная конференция в области нейронных сетей.

Большой вклад в развитие теории нейронных сетей внесли российские ученые Галушкин А.И. и Горбань А.Н., а также украинские – Куссуль Э.М. и Ивахненко А.Г.

В настоящее время исследования в области искусственных нейронных сетей ориентированы в основном на создание специализированных систем для решения конкретных задач. Разработано большое количество нейросистем, которые применяются в различных областях: прогнозирование, управление, диагностика в медицине и технике, распознавание образов и т.д. Рынок продуктов в области нейроинтеллекта растет стремительным образом. Происходит постепенное накопление критической массы для создания универсальных нейросистем, способных к различного рода интеллектуальной деятельности. В глобальном масштабе задача состоит в создании искусственного разума, обладающего способностью к воспроизводству и эволюции. Для реализации такого рода задач можно использовать нейрокомпьютер, базовым элементом которого является нейронная сеть.

Основным элементом нейронной сети является формальный нейрон, который осуществляет операцию нелинейного преобразования суммы произведений входных сигналов на весовые коэффициенты:

$$y = F\left(\sum_{i=1}^{n} \omega_i x_i\right) = F(WX),$$

где $X = (x_1, x_2, \ldots, x_n)^T$ – вектор входного сигнала;

$W = (\omega_1, \omega_2, \ldots, \omega_n)$ – весовой вектор; F – оператор нелинейного преобразования.

Схема нейронного элемента изображена на рис. 3.3.2. и состоит из сумматора и блока нелинейного преобразования F. Каждому *i*-му входу нейрона соответствует весовой коэффициент ω_i (синапс), который характеризует силу синаптической связи по аналогии с биологическим нейроном.

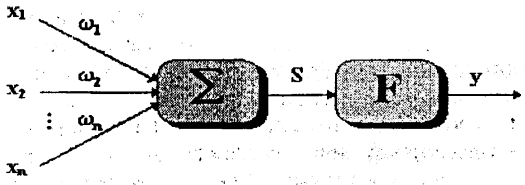


Рисунок 3.3.2 – Нейронный элемент

Сумма произведений входных сигналов на весовые коэффициенты называется *взвешенной суммой.* Она представляет собой скалярное произведение вектора весов на входной вектор:

$$S = \sum_{i=1}^{n} \omega_i x_i = (W, X) = |W| \cdot |X| \cdot \cos\alpha ,$$

где $|W|$, $|X|$ – соответственно длины векторов W и X, $\alpha = W, X$ – угол между векторами W и X.

Длины весового и входного векторов определяются через их координаты:

$$|W| = \sqrt{\omega_1^2 + \omega_1^2 + ... + \omega_n^2} ,$$

$$|X| = \sqrt{x_1^2 + x_1^2 + ... + x_n^2} .$$

Так как для нейронного элемента длина весового вектора после обучения $|W| = const$, то величина взвешенной суммы определяется проекцией входного на весовой вектор:

$$S = |W| \cdot |X| \cdot \cos\alpha = |W| \cdot X_W ,$$

где X_W – проекция вектора X на вектор W.

Если входные векторы являются нормированными, т.е. $|W| = const$, то величина взвешенной суммы будет зависеть только от угла между векторами X и W. Тогда при различных входных сигналах взвешенная сумма будет изменяться по косинусоидальному закону. Максимального значения она будет достигать при коллинеарности входного и весового векторов.

Если сила связи wi является отрицательной, то такая связь называется тормозящей. В противном случае синаптическая связь является усиливающей.

Оператор нелинейного преобразования называется функцией активации нейронного элемента. Вектор входного сигнала называется паттерном входной активности нейронной сети, а вектор выходного сигнала – паттерном выходной активности.

Рассмотренная здесь модель формального нейрона лишь отдаленно напоминает биологический нейрон. Она используется для построения искусственных нейронных сетей, которые являются базовыми элементами нейрокомпьютеров. Для работы нейрокомпьютера выполняется обучение сети, суть которого заключается в подаче набора образов с известной теоретически реакцией сети. Цель ее настройки – решение избранного класса задач. Настройка сети обычно сводится к изменениям весовых коэффициентов wij при многократной подаче образов до тех пор, пока не будет гарантирован предельный уровень ошибки в работе сети.

Теоретически число слоев и может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС. Чем сложнее НС, тем масштабнее задачи, подвластные ей.

Выбор структуры НС осуществляется в соответствии с особенностями и сложностью задачи. Для решения некоторых отдельных типов задач уже существуют оптимальные на сегодняшний день конфигурации. Если же задача не может быть сведена ни к одному из известных типов, разработчику приходится решать сложную проблему синтеза новой конфигурации. При этом он руководствуется несколькими основополагающими принципами: возможности сети возрастают с увеличением числа ячеек сети (нейронов), плотности связей между ними и числом выделенных слоев; введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической ус-

тойчивости сети, сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тромозящих и др.), также способствует усилению мощи НС. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Первые успехи нейросетевого подхода связаны с решением следующих, часто плохо формализованных, задач: прогнозирования, распознавания, классификации.

Задача распознавания образов, по существу, состоит в отнесении входного набора данных, представляющего распознаваемый объект, к одному из заранее известных классов. В частности, распознавание рукописных и печатных символов при оптическом вводе в ЭВМ (например, номеров автомобилей при слежении за их движением, типов клеток крови, речи, где нейросетевые подходы дали хорошие результаты.

Задача кластеризации данных сводится к группировке входных данных по присущей им «близости». Сеть кластеризует данные на заранее неизвестное число кластеров, особенно при сжатии данных.

Задача аппроксимации функций, когда по набору экспериментальных данных {(x1, y1)..., (xn, yn)}, представляющему значения yi неизвестной функции от аргумента xi, требуется найти функцию, аппроксимирующую неизвестную и удовлетворяющую некоторым критериям.

Задача важна для сложных систем управления динамическими объектами.

Задача предсказания по набору {y(t1),...,y(tn)} – предсказать поведение системы в момент tn+1, например, при управлении запасами, при принятии решений.

Задача оптимизации – найти решение NP-проблемы, удовлетворяющее ряду ограничений и оптимизирующее значение целевой функции (например, задача коммивояжера).

Причины развития работ по введению интеллектуальных компонентов в высокопроизводительные и обычные вычислительные сети – это в первую очередь обеспечение гибкости новых структур в динамике.

Ставка на параллелизм вычислений за счет использования жестко связанных компонентов сети (постоянное соединение) и ориентация на узкие классы задач исчерпала себя при переходе к решению более широких классов задач, для которых возникла проблема разработки методов алгоритмизации задач с учетом особенностей системы и организации вычислений с распараллеливанием; трудоемкое создание трансляторов и обучения кадров программистов; проблемы переносимости старого и вновь созданного программного обеспечения на другие машины; появились трудности в организации соединений между отдельными узлами сети с учетом конкретной задачи.

Поэтому взоры исследователей обратились к идее использования в системе элементов искусственного интеллекта. Привлекательной стороной явились два основных момента: возможность адаптации к решению нужного класса задач методами ИИ без больших усилий человека и возможность без разработки алгоритма программ после настройки системы получить алгоритм и программу в скрытой форме в виде соответствующей нейросети для реализации на нейрокомпьютере.

Отметим основные особенности в их работе.

Нейрокомпьютером называют ЭВМ (аналоговую или цифровую), основной операционный блок (центральный процессор), который построен на основе нейронной сети и реализует нейросетевые алгоритмы.

Нейрокомпьютер качественно отличается от действующих классических систем параллельного типа тем, что для решения задач используются не заранее разработанные алгоритмы, а специальная нейронная сеть, обучение которой проводится для решения избранного класса задач на специально подобранных примерах.

В качестве важных направлений развития нейрокомпьютеров выделяют следующие четыре:

1. Решение традиционных задач для ИИ: распознавание образов (зрение, слух, обоняние), классификация (извлечение знаний из данных и т.п.). На этой основе создаются модели искусственных органов человека: искусственный глаз, ухо, нос и др.

2. Решение сложных вычислительных задач (систем линейных уравнений, молекулярное конструирование лекарств, с получением легкого распараллеливания работы алгоритмов на основе НС)

3. Использование нейрокомпьютера как инструмента для моделирования работы структур человеческого мозга.

4. Создание на основе концепции НС принципиально новых систем обработки информации со свойствами адаптации к быстро меняющейся обстановке, возможностями высокоскоростной обработки как аналоговой, так и дискретной информации, возможностью решения оптимизационных задач в реальном времени.

В принципе, все задачи обычно делят на 3 класса: формализуемые (допускающие получение четкого алгоритма их решения); трудноформализуемые (качество алгоритма решения которых трудно оценить или вообще получить достижимое решение); неформализуемые (задачи с неявно заданными функциями и параметрами типа распознавания образов, предсказания, аппроксимации заданий и т.п.).

Нейрокомпьютеры в основном используются для решения задач третьего класса (в распознавании рукописных и печатных символов при оптическом вводе текстов в ЭВМ, речи, в задачах страхования и учета рисков, аппроксимациях неизвестных в явном виде функций и т.п.)

Опишем общую структурную схему абстрактного нейрокомпьютера (рис. 3.3.3).

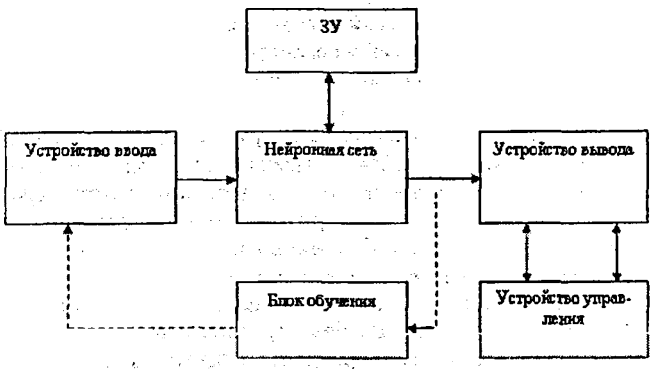


Рисунок 3.3.3 – Структурная схема абстрактного НК

Операционным блоком НК (процессором) является искусственная НС, состоящая из формальных нейронов, соединенных каналами передачи информации. Этот блок в обычном понимании не производит вычислений. Он трансформирует входной сигнал

(образ) в выходной в соответствии с топологией НС и значениями коэффициентов межнейронной связи. В ЗУ хранится не программа решения задачи, а программа изменений коэффициентов связи между нейронами. Устройства ввода и вывода информации выполняют обычные функции. Устройство управления служит для синхронизации работы всех структурных блоков НК при решении конкретной задачи.

В работе НК выделяют два режима: обучения и рабочий. Чтобы НК решал требуемую задачу, его НС должна пройти обучение на эту задачу. Суть обучения заключается в настройке межнейронных связей на совокупность входных образов этой задачи. Установка коэффициентов осуществляется на примерах, сгруппированных в обучающие множества (эталонные наборы).

Настройка и решаемая задача идут по итерационной схеме: при подаче очередного эталонного образа на НС выходной сигнал может отличаться от желаемого. Блок обучения оценивает величину ошибки и корректирует коэффициенты межнейронных связей с целью уменьшения ошибки на следующем шаге. По мере повторения процедуры ошибка уменьшается, и процесс обучения завершается при достижении ошибки, менее заданной величины. Такой тип обучения относят к классу обучения с учителем. В рабочем режиме блок обучения отключается.

3.4. Нейронные сети и их применение

Существуют различные подходы к построению нейросетевых систем управления мобильными роботами. Они основываются на применении разных моделей нейронных сетей и концепций их обучения. Большое значение при этом имеет система реактивного управления, которое осуществляется при движении робота в неизвестном пространстве.

С точки зрения модели обучения, наиболее часто применяются в нейросетевых системах управления методы подкрепляющего обучения и методы обучения с учителем.

В качестве транспортных средств могут использоваться мобильные роботы или автомобиль, которые оснащены сенсорными устройствами для отображения окружающей обстановки. После обучения нейронная сеть на основе информации от сенсорных устройств должна обеспечивать корректное управление движением. Возможность создания таких систем базируется на обобщающей способности нейронных сетей, которая позволяет интегрировать частные данные для определения закономерностей процесса. В результате этого нейронная сеть способна выдавать правильную реакцию на входных данных, которые не входили в обучающую выборку. Общая модель взаимодействия таких систем с внешней средой изображена на рис. 3.4.1.

Самоорганизация здесь происходит в процессе обучения с целью адаптации к внешней среде. Данная схема эквивалентна модели взаимодействия индивидуума с внешней средой.

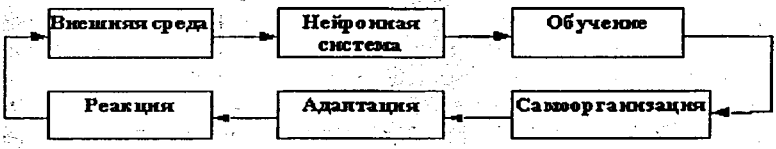


Рисунок 3.4.1 – Взаимодействие нейронной системы с внешней средой

Во многих лабораториях мира разрабатываются системы для управления роботами. Существует большое количество моделей роботов и различных проектов, вплоть до

создания человекоподобных роботов. Нейросетевой подход создает здесь новые во
можности по сравнению с традиционным аппаратом.

Нейронные сети могут эффективно использоваться для управления автомобиле
В этом случае достаточно задать системе управления координаты конечной точки дв
жения – и автомобиль без участия человека будет двигаться к цели. Такая система ра
работана в университете Карнеги-Меллона в рамках ALVINN проекта (Autonomous Lar
Vehicle In a Neural Networks). В ней предполагается, что автомобиль оборудован виде
камерой, которая отображает дорогу с разметкой. Центральным элементом такой си
темы является трехслойная нейронная сеть с прямыми связями (рис. 3.4.2).

Входной слой содержит 30x32 нейронных элемента, на которые подается преобр
зованное от видеокамеры изображение пути. Скрытый слой состоит из пяти, а выходно
слой из 30-ти нейронных элементов. В качестве функции активации используется сигм
идная функция. Активность выходных нейронов характеризует поворот руля в ту ил
иную сторону. Так, если максимальной активностью обладает центральный нейрон,
это означает движение прямо. Когда наибольшую активность имеет крайний левый не
рон, то это соответствует повороту налево на определенное число градусов. Нейронна
сеть обучается при управлении автомобилем оператором. При этом оператор управля
автомобилем при движении со скоростью 9.5 км/ч, моделируя различные ситуации. Из
бражение от видеокамеры использовалось как вход, а текущее направление руля – к
желаемый выход. С целью упрощения получения обучающей выборки используетс
программное вращение изображения от видеокамеры (рис. 3.4.3), и соответствующи
образом меняется реакция нейронной системы.

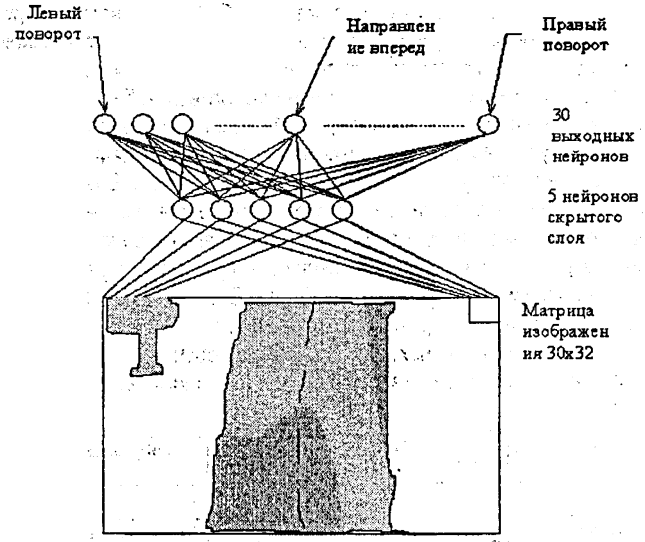


Рисунок 3.4.2 – Отображение изображения дороги на нейронную сеть

В результате была создана обучающая выборка, объем которой составил 12(
тренировочных наборов. Обучение нейронной сети проводилось с использованием тр
станций «Sun-4». Время обучения методом обратного распространения ошибки сост

вило пять минут. После обучения, как показали эксперименты, нейронная сеть может автономно управлять автомобилем. В настоящее время в рамках этого проекта была достигнута скорость движения автомобиля до 70 миль/ч. При этом автомобиль проехал 90 миль к северу от Питтсбурга, что свидетельствует о большом потенциале нейронных сетей для решения различных задач.

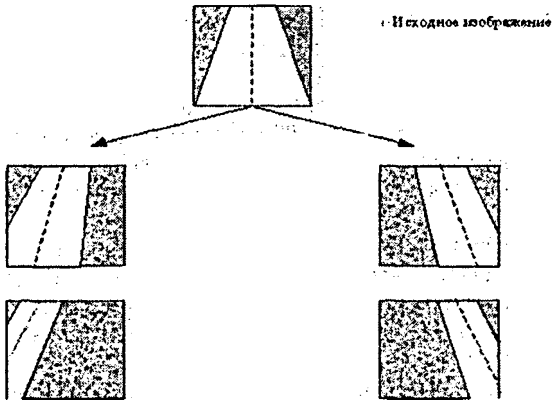


Рисунок 3.4.3 – Вращение исходного изображения от видеокамеры

3.5.Роботы в управлении дорожным движением автотранспорта

Бурный процесс автомобилизации с каждым годом охватывает всё большее число стран, постоянно увеличивается автомобильный парк, количество вовлекаемых в сферу дорожного движения людей. Рост автомобильного парка и объем перевозок ведёт к увеличению интенсивности движения, что в условиях городов с исторически сложившейся застройкой приводит к возникновению транспортной проблемы. Особенно остро она проявляется в узловых пунктах улично-дорожной сети. Здесь увеличиваются транспортный задержки, образуются очереди и заторы, что вызывает снижение скорости сообщения, неоправданный перерасход топлива и повышенное изнашивание узлов и агрегатов транспортных средств. Переменный режим движения, частые остановки и скопления автомобилей на перекрёстках являются причинами повышенного загрязнения воздушного бассейна города продуктами неполного сгорания топлива. Городское население постоянно подвержено воздействию транспортного шума и отработанных газов.

В последние годы за рубежом находят широкое распространение адаптивные системы управления автотранспортом.На Западе разрабатываются системы управления четвёртого поколения, учитывающие дорожную обстановку, интенсивность транспортных потоков, скорость автомобилей, фазы дорожного движения.

Фирма "Tune & Wear" (Великобритания) представила "умный" светофор, включающий мини-камеры, которые оценивают дорожную обстановку и устанавливают периодичность переключения цветов. Устройство идеально подходит для борьбы с дорожными пробками.

В Институте информации и системного анализа г. Мануа (Италия) создан светофор, оснащенный системой анализа загруженности дорог, который очень точно приспосабливается к различным ситуациям в городском дорожном движении, кроме того, он не

За последние десятилетия, особенно в связи с развитием искусственного интеллекта, термин «адаптация» стал все более широко применяться в технике. Под адаптацией в широком смысле слова сейчас понимается процесс изменения свойств системы, позволяющий ей достигнуть определенного, часто оптимального или по крайней мере удовлетворительного функционирования при начальной неопределенности и изменяющихся внешних условиях. В процессе адаптации системы могут изменяться ее восприимчивость к внешним воздействиям, те или иные параметры и структура системы, алгоритм ее функционирования и т.п.

Адаптивное управление на отдельном перекрестке состоит в постоянном нахождении оптимальных для данных средних значений интенсивностей движения длительностей цикла и фаз регулирования, а также в корректировке этих длительностей в соответствии с мгновенными колебаниями в количестве автомобилей, подходящих к перекрестку. Для этого необходима реализация звена обратной связи между параметрами транспортного потока и управляющими воздействиями системы. Параметры транспортного потока (интенсивность, скорость, плотность, длина очереди у перекрестка, наличие транспортных средств с правом приоритетного пропуска и т.д.) фиксируются с помощью детекторов транспорта (ДТ). Полученная информация о состоянии транспортного потока обрабатывается, и полученные результаты используются для управления, а также могут служить основой для вычисления таких характеристик потока, которые нельзя получить непосредственным измерением.

Детекторы транспорта предназначены для обнаружения транспортных средств и определения параметров транспортных потоков. Эти данные необходимы для реализации алгоритмов гибкого регулирования.

Использование автоматизированных систем адаптивного управления автотранспортом на регулируемом перекрёстке позволит решить следующие задачи:

- сократить время нахождения АТС перед светофором;
- сократить количество дорожных пробок;
- снизить потребление топлива;
- снизить количество вредных выбросов в атмосферу;
- сохранить моторесурс автомобиля.

В 80 годы прошлого столетия на основе исследований в области искусственного интеллекта (ИИ) сформировалась новая отрасль индустрии – производство интеллектуальных систем (ИС). Интеллектуальные системы предназначены для выполнения таких практических задач, которые называются интеллектуальными, если они выполняются людьми.

В последнее время в ИИ активно используется *принцип прецедентов*. Прецедент – это описание проблемы или ситуации в совокупности с подробным указанием действий, предпринимаемых в данной ситуации или для решения данной проблемы. Подход, основанный на прецедентах, в целом состоит из следующих компонентов:

- получение подробной информации о текущей проблеме;
- сопоставление (сравнение) этой информации с деталями прецедентов, хранящихся в базе знаний (прецедентов), для выявления аналогичных случаев;
- выбор прецедента, наиболее близкого к текущей проблеме, из базы знаний;
- адаптация выбранного решения к текущей проблеме, если это необходимо;
- проверка корректности каждого вновь полученного решения;
- занесение детальной информации о новом прецеденте в базу знаний.

нуждается в централизованном компьютерном управлении, поэтому обходится гораздо дешевле. Светофор сам меняет фазы зелёного и красного света в зависимости от плотности движения на перекрёстке.

Главное полицейское управление Японии (NPA) сообщило о создании нового типа светофора, который самостоятельно "разруливает" заторы. С помощью специальных датчиков фиксируется интенсивность движения на перекрёстке, и в случае возникновения «пробки» светофор автоматически выбирает наиболее подходящий для её рассасывания режим смены красного света на зелёный. Экспериментальная система подтвердила эффективность данного метода.

Власти Чикаго (США) предложили решение проблемы бесконечных пробок, которая в городе стоит крайне остро. На 2900 городских перекрёстках будут установлены камеры и датчики новой системы регулирования дорожного движения, связанные с «умными» светофорами. Свет автоматически будет переключаться в соответствии с ситуацией на перекрёстке.

В настоящее время в РБ система адаптивного управления автотранспортом не производится. Производится «жесткое регулирование», при котором светофор работает автономно по заложенным в нём режимам переключения цветов. Переход на адаптивное управление позволит сократить количество дорожных пробок, количество вредоносных выбросов в атмосферу, а также снизить потребление топлива.

Существующее сегодня жесткое программное регулирование на перекрёстках города не способно учитывать кратковременные случайные колебания в числе автомобилей, подходящих к перекрёстку. При медленном изменении интенсивностей движения оптимальные длительности цикла и фаз, рассчитанные для условий пикового периода, для остального времени суток оказываются неоптимальными, как правило, слишком большими, приводящими к неоправданным задержкам транспорта. В таких случаях необходимо внедрение программ координации, ориентированное на выделение пиковых периодов. И всё же такая система координированного управления не сможет учитывать случайный характер колебаний в числе автомобилей, подходящих к перекрёстку за одинаковые периоды времени.

Задача улучшения автоматического управления движением на перекрестке состоит в создании технических средств и алгоритмов управления, которые обеспечили бы адаптацию режимов регулирования к изменению условий движения.

До недавнего времени термин «адаптация» относили только к области биологии, понимая под адаптацией приспособление организмов в целом к условиям существования или приспособление отдельных органов к тем или иным достаточно длительным воздействиям.

Процессы адаптации проявляются как в развитии видов живых организмов на протяжении многих поколений (в филогенезе), так и в развитии отдельного живого организма от его зарождения до смерти (в онтогенезе). Мы говорим, например, об адаптации южанина к северному климату, об адаптации растения, пересаженного в другую почву, и т.д. Наиболее наглядным примером адаптации отдельных органов является адаптация органов чувств. Весьма эффективно приспосабливается к условиям освещенности зрительный анализатор. При переходе из темного помещения в ярко освещенное человек вначале ослеплен и не различает окружающих предметов, однако по мере адаптации к свету ослепление проходит, и он начинает отчетливо видеть окружающее. Аналогичное явление наблюдается и при переходе из условий яркого освещения в темное помещение.

На рис. 3.5.1 представлена граф-схема игры, активно использующая принцип пр цедентов. Схема игры представляет собой граф, вершинами которого являются текущ ситуации (позиции), а ребра указывают направления переходов. На графе темным ве шинам соответствуют ситуации-прецеденты, т.е. такие позиции, с которых игра одн значно (по колее) ведёт к цели (выигрышу). Тёмные вершины – это те позиции, котор когда-то уже встречались в играх и даже, может быть, описаны в литературе как класс ческие. Они хранятся в памяти компьютера.

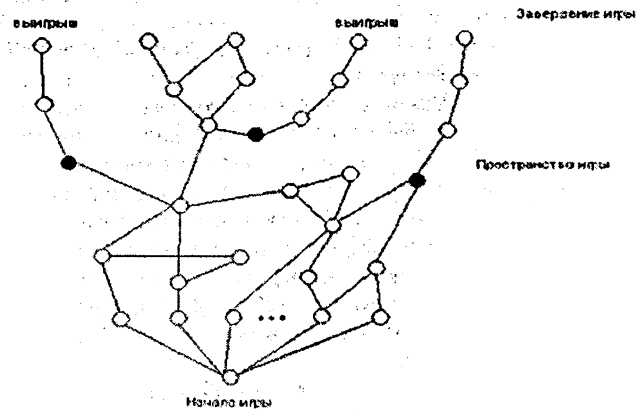


Рисунок 3.5.1 – Граф-схема игры

Аналогичным образом может быть построено управление светофорным объекто (СФО), использующим принцип прецедентов. То есть, на определённые ситуации, част встречающиеся на перекрестке, управление СФО отвечает определённой оптимальнс реакцией, хранящейся в памяти управляющего компьютера. Таким образом, обеспеч вается быстрая реакция на ситуацию без потери времени на просчёт оптимального в рианта управления.

На заре возрождения автотранспорта, когда ещё не было светофоров, на перекр стках управление производилось постовым. Функции постового:

1. Принять зрительный образ авто;
2. Оценить, с каких направлений они поступили;
3. Просчитать оптимальную стратегию их разъезда;
4. Выполнить эту стратегию движением рук и тела.

Человек, по своей сути, является природным оптимизатором. После того, как с п мощью светофора автоматизировали разъезд на перекрёстке, то и интеллект также у рали с перекрёстка.

Сейчас идёт процесс возвращения интеллекта на перекрёсток. С диалектическс точки зрения, совершенно верно, то есть произошло развитие по спирали (новые мат матические методы и технические средства в известном объекте).

Интеллектуальные системы (ИС) общего класса (рис 3.5.2) могут решать задач которые ранее не встречались (представитель – экспертные системы). Специализир ванные ИС выполняют решение фиксированного набора задач, определённого на эта проектирования. Функции интеллектуального светофора:

1. Сбор и обработка информации о АТП на перекрёстке;
2. Выбор оптимального момента смены фазы светофорного цикла;

Рисунок 3.5.2 – Интеллектуальные системы

3. Статистическая обработка информации о АТП в течение суток;
4. Прогнозирование будущих ситуаций;
5. Наличие различных программ регулирования движения:
- на малых потоках;
- на потоках средней интенсивности;
- на насыщенных потоках;
- при наличии пробок.

6. Использование аналитических методов расчёта режимов оптимальности либо работа на прецедентах.

Дальнейшим развитием в управлении транспортными потоками является создание интеллектуальной транспортной сети (магистрали). Оптимальная работа по отдельности каждого СФО не гарантирует оптимальности всей потокопроводящей сети города (принцип теории систем). Каждый СФО должен работать не на себя, а на систему. Отсюда в него вводится дополнительная функция: общение. Только оперативная информация о потоках на магистрали позволит оптимально управлять СФО.

Интеллектуальное управление на отдельном перекрестке состоит в постоянном нахождении оптимальных для данных средних значений интенсивностей движения длительностей цикла и фаз регулирования, а также в корректировке этих длительностей в соответствии с мгновенными колебаниями в количестве автомобилей, подходящих к перекрестку. Для этого необходима реализация звена обратной связи между параметрами транспортного потока и управляющими воздействиями системы.

3.6. Интеллектуальные системы в диагностике

Количество и сложность технических систем, поставленных на службу человеку в самых различных областях его деятельности, достигли чрезвычайно высокого уровня. Важность решаемых задач, большие потери от простоев из-за различного рода неисправностей и возможность аварий с тяжелыми последствиями вызывают необходимость в безотказной работе этих систем. Несмотря на успехи, достигнутые в области повышения надежности, полной безотказности в работе технических систем достичь невозможно. Это объясняется естественным старением и износом элементов систем, а также и тем, что увеличение сложности систем обычно сопровождается уменьшением надежности. Поэтому особо важное значение приобретают различные аспекты проблемы восстановления работоспособности систем. Успехи в этом направлении должны привести к значительному экономическому эффекту. Достаточно сказать, что, по подсчетам американских инженеров, «...затраты на ремонт аппаратуры военного назначения примерно в 000 раз превышают ее первоначальную стоимость».

Для восстановления работоспособности технической системы необходимо прежде всего определить ее состояние, т.е. найти те элементы, ненормальное функционирование которых привело к потере работоспособности. Решение этой задачи (задачи диагностики) связано с большими затратами времени и средств и требует привлечения обслуживающего персонала высокой квалификации. Обработкой большого объема статистической информации было установлено, например, что «...около 80% времени ремонта расходуется на отыскание и устранение неисправностей». Поэтому трудно преувеличить роль исследований, направленных на изыскание эффективных и экономичных методов и средств диагностики технических систем.

Во многих случаях логическим завершением такого рода исследований должны служить автоматически действующие устройства для определения состояния диагносцируемых систем. Однако построение диагносцирующих автоматов оправдано далеко не всегда, решающим фактором здесь является экономическая целесообразность. Отсюда следует, что одной из важнейших задач является разработка способов расчета эффективности автоматической и ручной диагностики. Следует отметить, что при определении состояния диагносцируемой системы человеком-оператором необходимость в теоретических исследованиях не отпадает. Действительно, оператор должен пользоваться научно обоснованной программой поведения – четкой и подробной последовательностью заранее предписанных действий. Только при этом условии возможно сокращение времени и затрат на восстановление исследуемой системы и обслуживание персоналом более низкой квалификации. Таким образом, подобные автоматы должны обладать высокой степенью интеллектуальности, так как они заменят труд высококвалифицированных специалистов.

Необходимость в создании специальных интеллектуальных диагностических автоматов, а также их сложность и стоимость существенным образом зависят от того, в какой мере исследуемые системы приспособлены для определения их состояний. Поэтому одно из наиболее перспективных направлений состоит в разработке методов синтеза технических систем с учетом требований диагностики.

С момента появления первых ЭВМ ведутся работы по созданию автоматически действующих диагностических устройств. Известные типы диагностических устройств предназначены главным образом для автоматической проверки работоспособности и поиска неисправностей в изделиях радиоэлектронной промышленности. Подобные устройства имеются и для обслуживания изделий других отраслей производства, в частности автотракторных двигателей, без их предварительного демонтажа. Ведутся также исследования по созданию общих методов синтеза диагностических устройств, по разработке специфических методов диагностики различного рода систем и по оптимизации программ диагностики в соответствии с выбранными критериями.

Техническая диагностика изучает и устанавливает признаки (симптомы), характеризующие состояние технических систем, а также методы экспериментального определения состояния этих систем (методы диагностики). Изучение и установление признаков, характеризующих состояние, предполагает эмпирическое изучение конкретных систем, которые выступают в качестве объектов диагностики. Изучение методов диагностики связано с анализом технических систем и процессов постановки диагноза (процессов диагностики). Этот анализ предполагает наличие математических моделей, позволяющих широко использовать современный аппарат математики для решения задач технической диагностики. Поэтому предметом исследования являются как конкретные техни-

ские системы и процессы диагностики, так и их математические модели. Техническая агностика охватывает целый ряд задач эмпирического, теоретического и прикладного рактера. Эти задачи связаны с разработкой принципов и методов проверки работо-особности систем в целом, поиском неисправных или отказавших элементов, профи-актическим обслуживанием, позволяющим предупредить неисправности в системе. сновными задачам технической диагностики являются:

- построение математических моделей объектов и процессов диагностики;
- построение оптимальных программ диагностики, т.е. оптимальных последова-льностей проверок, позволяющих определить состояние технической системы;
- разработка методики оценки целесообразности и экономической эффективности томатизации процесса диагностики;
- разработка методов синтеза автоматически действующих диагностических уст-йств;
- разработка практических рекомендаций, которые должны учитываться при проек-ровании технических систем, для того чтобы существенно облегчить и ускорить их ди-ностику.

При построении математических моделей технических систем делается предполо-ние, что система состоит из элементов, каждый из которых может находиться только дном из двух несовместимых состояний – исправен, неисправен. Состояние системы елом является функцией, зависящей от состояния ее элементов. Система исправна аботоспособна), если все ее элементы исправны. Неработоспособность системы в лом вызывается неисправностью любого одного или нескольких ее элементов. Для становления работоспособности системы достаточно определить все ее неисправ-е элементы и восстановить их работоспособность. Таким образом, каждое из воз-жных состояний технической системы, содержащей N элементов, можно представить мерным вектором, i-я компонента которого равна 1, если i-й элемент системы испра-н, и равна 0, если он неисправен.

Состояние элементов системы определяется выполнением некоторой последова-льности проверок (программы диагностики). Проверка – это совокупность операций, оизводимых над диагностируемым объектом с целью получения некоторого результа-по которому можно судить о состоянии *по* крайней мере *одного* элемента системы. В сло основных операций, выполняемых при осуществлении проверки, входит контроль изнаков, характеризующих состояние системы.

Математические модели строятся на основе эмпирического исследования техниче-их систем и процессов, которое включает:

- изучение нормального функционирования системы;
- выделение элементов системы и связей между ними;
- выделение возможных состояний системы;
- анализ технических возможностей контроля признаков, характеризующих состоя-системы;
- сбор и обработку статистических материалов, позволяющих определить распре-ление вероятностей возможных состояний системы, а также закономерности прояв-ия неработоспособности элементов системы;
- сбор экспериментальных данных о затратах, связанных с осуществлением проверок.

Задача построения оптимальных программ диагностики носит в основном матема-еский характер. Решение этой задачи, полученное для конкретной технической сис-

темы, дает возможность определять ее состояние с минимальными затратами. При томатизации процесса диагностики оптимальная программа должна служить осно для разработки алгоритма функционирования интеллектуального диагностического ройства.

Весьма большое значение имеет разработка методики оценки целесообразност экономической эффективности автоматизации процесса диагностики. Это объясняе тем, что во многих случаях сложность автоматически действующих диагностических ройств приближается к сложности современных электронных цифровых вычислите ных машин. Такие устройства зачастую оказываются недостаточно надежными и эко мически мало эффективными. Разработка такой методики позволит в каждом конкр ном случае определить разумную степень автоматизации процесса диагностики и брать соответствующий принцип действия диагностического устройства.

Очень часто современные технические системы проектируются без учета требо ний диагностики. Это проявляется главным образом в отсутствии необходимого чис контрольных точек, которые требуются для установления признаков, или в недостаточ удобном их расположении. Кроме того, автоматизация процесса диагностики требуе свою очередь специальной организации технических систем, допускающей быстрое удобное присоединение диагностических устройств. Поэтому важное значение име разработка научно обоснованных рекомендаций, учет которых уже на этапе проектир вания технических систем позволит выбирать принцип действия системы, отвечающ требованиям технической диагностики.

Достаточно полная характеристика предмета и задач той или иной научной дисц плины невозможна без рассмотрения связей этой дисциплины с другими. Выделени предмета и формулировка задач создают базу для такого рассмотрения. Кроме тог выявление этих связей позволяет определить более четко специфические особеннос предмета и задач данной дисциплины. К числу научных дисциплин, наиболее близких технической диагностике, прежде всего необходимо отнести теорию автоматическо контроля и теорию надежности.

Тесные связи между технической диагностикой и контролем являются следствие того, что задачи технической диагностики возникли как логическое продолжение и ра витие задач контроля. И если говорить о целях контроля и диагностики, то их можн сформулировать одинаково: определить, в каком из заранее установленного множеств различимых состояний находится исследуемая система. Специфика рассматриваемы дисциплин становится ясной, если конкретизировать, что именно следует понимать по системой и ее состояниями. В наиболее простом случае число состояний, характери зующих систему, может быть два: «работоспособна» или «неработоспособна». Выделе ние этих двух состояний системы достаточно для того, чтобы говорить о контроле, н недостаточно, чтобы говорить о технической диагностике.

Математические модели объектов диагностики. В настоящее время предложен несколько математических моделей объектов диагностики. Эти модели отражают воз можные состояния, элементы и структуру некоторых классов технических систем. Ана лиз математических моделей позволяет определять множество проверок $П = \{p_i\}$, по ре зультатам которых можно судить о состоянии исследуемых систем, находить подмноже ство различимых состояний (два состояния различимы, если выполнение всех проверо из П достаточно для определения, в каком из этих двух состояний находится система), а также устанавливать необходимые для диагностики преобразования структуры системы

К таким преобразованиям относятся, в частности, введение дополнительных контрольных точек и разрывов цепей прохождения сигналов.

В наиболее простых математических моделях не учитывается структура системы, т.е. предполагается, что система состоит из некоторого числа не связанных между собой элементов. Множество возможных проверок для таких моделей включает общую проверку работоспособности системы в целом и проверки, устанавливающие состояние каждого из элементов в отдельности. В некоторых моделях множество всех несвязанных элементов системы разбивается на ряд непересекающихся подмножеств (модули) и допускаются проверки работоспособности каждого модуля.

Имеются модели объекта диагностики, при построении которой (хотя и в неявной форме) учитывается структура системы. Для того чтобы задать эту модель, необходимо указать воздействия, которые должны быть приложены к внешним входам системы, и функциональную связь между воздействиями и реакциями, наблюдаемыми на внешних выходах системы в зависимости от состояния системы. Любая возможная для данной модели проверка состоит в определении реакции системы на заданное воздействие. Такая модель применялась для построения тестов, охватывающих как проверку работоспособности, так и диагностику состояния контактных схем.

Другая модель объекта диагностики основана на том, что диагносцируемая система рассматривается как конечное множество связанных между собой элементов. Каждый элемент системы отвечает определенной реакцией на приложенную к нему совокупность воздействий, в число которых могут входить реакции других элементов. Воздействия и реакции, которые могут появиться в процессе нормального функционирования системы (т.е. когда все ее элементы исправны), называются допустимыми. Реакция неисправного элемента при любых условиях является недопустимой. Появление недопустимой реакции на выходе хотя бы одного элемента свидетельствует о неработоспособности системы в целом. Два элемента системы связаны между собой, если реакция первого элемента является воздействием для второго и если недопустимая реакция первого элемента вызывает недопустимую реакцию второго независимо от состояния второго элемента и от остальных воздействий, приложенных к нему. Каждая возможная для этой модели проверка состоит в контроле реакции одного из элементов системы на определенную совокупность воздействий. Для задания рассматриваемой модели необходимо указать множество элементов, множество возможных состояний системы и схему объекта, отражающую связи между элементами. Данная модель применялась для анализа электронных систем, осуществляющих преобразование непрерывных сигналов.

Естественно, что небольшое количество математических моделей объектов диагностики, известных в настоящее время по литературным источникам, явно недостаточно для диагностики всего многообразия технических систем и устройств. Можно указать ряд нерешенных задач, связанных с построением и исследованием математических моделей объектов диагностики. К таким задачам относится разработка методики построения математических моделей конкретных объектов диагностики; создание обобщенных математических моделей, описывающих широкие классы объектов диагностики, таких, как конечные автоматы, конкретные схемы, пассивные электрические цепи и т.д.; разработка методов анализа математических моделей объектов диагностики. Следует отметить, что анализ таких моделей позволяет определить множество возможных для данной модели проверок, а тем самым перейти к построению математической модели процесса диагностики, которая должна содержать также сведения о затратах (время, стоимость)

на выполнение отдельных проверок, о достоверности результатов этих проверок, о рас пределении вероятностей возможных состояний и о критерии, в соответствии с которы должна быть построена оптимальная программа диагностики.

Построение диагностических тестов и оптимальных программ диагностики Диагностику состояния сложных технических систем можно проводить двумя сущест венно различными методами.

При использовании первого метода состояние системы определяется путем вы полнения некоторого числа проверок, порядок (очередность) осуществления которы безразличен. Такой метод диагностики получил название комбинационного. Совокуп ность проверок, достаточных для диагностики всех заранее заданных различимых со стояний системы и выполняемых при реализации комбинационного метода, называетс диагностическим тестом.

При использовании второго метода диагностики проверки, осуществление которы достаточно для определения всех заранее заданных различимых состояний системы выполняются в некотором порядке. Порядок выполнения проверок может быть строг фиксированным или же зависеть от результатов предыдущих проверок. Результат ка дой проверки анализируется непосредственно после его получения, и если состояни системы еще не определено, то выполняется следующая по порядку проверка. Про граммы, реализующие второй метод диагностики, можно подразделить на условные, которых каждая последующая проверка назначается в зависимости от исхода предыду щей, и последовательные в которых проверки выполняются в некотором строго фикси рованном порядке.

Диагностические интеллектуальные системы в медицине. Одним из важных на направлений искусственного интеллекта является разработка методов управления и ис- следования, пригодных для сложных систем, столь сложных, что человек не в состояни ни наблюдать их, ни управлять ими полностью, не говоря уже о невозможности произво дить точные расчеты, позволяющие предсказывать их поведение.

Отличительной особенностью сложной системы является огромное количеств информации, с которой приходится сталкиваться при наблюдении и управлении ею. Эт информацию составляют, в частности, данные о состоянии и поведении отдельных час тей системы.

Таким образом искусственный интеллект связан с проблемами управления слож ными системами, с проблемами передачи, обработки и оценки информации, с примене нием для этой цели современных электронных математических машин. Какое же отно шение имеет все это к медицине?

Важнейшей задачей медицины, и в частности лечебной, является разработка таки планов лечения (терапевтических или хирургических), которые приводили бы человече ский организм из патологического состояния в практически здоровое. Другими словами все усилия медицины направлены на выработку и осуществление некоторого оптимал ного метода управления такой сложной системой, какой является человеческий орга низм. Таким образом, здесь мы встречаемся с задачей, которая носит отчетливо интел лектуальный характер.

В области медицины существуют два важных направления. Первое из них связан с проблемой управления человеческим организмом в процессе лечения – разработко новых методов обработки обширной медицинской информации и оценкой на этой осно ве постоянно меняющейся ситуации, в частности с постановкой диагноза, выбором оп тимальных планов лечения и т.д.

Второе направление связано с применением математического аппарата и электронных математических машин с целью глубокого изучения и математического описания деятельности тех или других функциональных систем управления внутри организма.

В 1960 г. в Институте хирургии им. А. В. Вишневского создана специальная лаборатория, оснащенная современными электронными машинами. Уже сейчас на основе опыта работы этой лаборатории ясно видны большие возможности использования достижений кибернетики в медицине. Остановимся на двух проблемах, разработанных в большей степени, чем другие.

Первая проблема состоит в разработке и создании динамической диагностической системы, позволяющей по обширной информации, получаемой в результате обследования больного (электрокардиография, фонокардиография, рентгеновское исследование, зондирование сердца и сосудов и т.п.), составить точное представление о его состоянии в данный момент времени.

Создание таких диагностических систем имеет для медицины самостоятельное значение, которое трудно переоценить. Однако в хирургии возможность с помощью кибернетической машины быстро оценить состояние больного во время операции является первым этапом решения более сложной задачи – создания системы автоматического управления жизненными функциями организма в процессе операции (автоматическое регулирование уровня артериального давления больного, автоматический наркоз, автоматическое управление работой аппаратов искусственного кровообращения, искусственной почки и т.д.).

Второй проблемой является создание автоматической медицинской информационной системы. Сущность разработанной системы состоит в том, что каждый клинический случай, каждая история болезни кодируются на перфокарте. Массив таких перфокарт представляет собой обширный медицинский архив. Затем на математических машинах осуществляется логический процесс, позволяющий быстро отыскать в этом массиве прецедент для каждого рассматриваемого клинического случая. Такая система дает возможность создать обширную память, охватывающую опыт многих клиник одной страны или даже нескольких стран.

Таким образом, можно не только аккумулировать обширный медицинский опыт, но и в любой момент быстро его использовать. Кроме отыскания клинических прецедентов такая информационная система позволяет осуществить математическую обработку клинического опыта, чтобы установить взаимосвязь между симптоматикой и заболеваниями. Только по врожденным порокам сердца объем информационной системы охватывает сейчас более тысячи историй болезни.

Перейдем теперь к более подробному рассмотрению основ построения диагностической системы.

Следует отметить, что разработанный диагностический процесс представляет собой не одномоментный акт, а некоторую динамическую процедуру. Больной не подвергается сразу всем видам исследований. Многие из них тяжелы для больного, а некоторые даже связаны с риском для жизни. Поэтому исследование больного начинается с самых простых испытаний, таких, как анамнез, электрокардиография, фонокардиография, рентгеноскопия и др. Затем на основании полученных данных проводится диагностическая оценка. Если оказывается, что этих данных достаточно, то система ставит окончательный диагноз.

Если же этих данных недостаточно для постановки диагноза, то система указывает, какое следующее испытание из группы более тяжелых нужно произвести для того,

чтобы восполнить знания новыми необходимыми данными. Производится это нов следующее испытание, и снова производится оценка. Если на основании данных, пол ченных от этого нового испытания, можно окончательно поставить диагноз, то диагн ставится и испытания прекращаются. Если же диагноз поставить нельзя, то опять ук зывается следующее испытание, вносящее дополнительную информацию. Таким обр зом, диагностический процесс включает в себя как оценку медицинской информации состоянии больного, так и управление собственно процессом диагноза.

Внутреннюю организацию системы можно разделить на две части. С одной стор ны, это аккумулированная медицинская память, т.е. медицинский опыт в данной облас заболеваний, и, с другой стороны, – логический процесс мышления, который позволя воспользоваться этим медицинским опытом и теми данными о состоянии больного, к торые вводятся в машину.

Проблема построения диагностического процесса при помощи математическ машин требует разработки логических основ процесса машинного диагноза, на базе к торых можно было бы создать машинный алгоритм. Как указывалось ранее, диагност ческий процесс следует планировать как динамическую процедуру, состоящую из ряд этапов, каждый из которых заключается в производстве серии испытаний, оценке ситу ции, возникшей в результате проведенных испытаний, и назначении следующей групп испытаний. В основу упомянутых оценок должны быть положены определенные крит рии, и вся логика существенно зависит от того, что берется в качестве таких критериев.

При выполнении определенных количественных оценок, естественно, пользуют некоторыми числовыми характеристиками, связанными с существом рассматриваемо процесса. Эти числовые характеристики стремятся получить из медицинского опыта м тематической обработкой накопленного материала в виде историй болезни и т.д. Одна несовершенство знаний, недостаточность материала и т.д. приводят к тому, что эти ч словые данные часто весьма неточны. Поэтому очень существенным является пробл ма автоматического улучшения указанных данных самой математической машиной процессе ее работы, т.е. организация самообучающегося процесса.

3.7. Интеллектуальные информационные технологии

Понятие технологии имеет своими истоками материальное производство. Поэто технологию до появления ЭВМ обычно определяли как науку о способах воздействия н сырье, материалы и полуфабрикаты соответствующими орудиями производства. О новным носителем технологии выступал человек. Он знал, каким способом, средства и инструментами реализовать процесс преобразования исходных ресурсов или пол фабрикатов. Появление ЭВМ в качестве своеобразного инструмента, способного выст пить в качестве носителя технологии, привело к выработке понятия информационно технологии, а в последующем и интеллектуальной информационной технологии. Новс качество в производстве появилось благодаря введению в реализацию технологий ко плекса средств для сбора, обработки, хранения, передачи и отображения информаци без участия человека или с его незначительным участием. Технологии, реализуемые применением таких средств стали называть информационными. Главным элементом них была дистанционная передача информации и ее обработка, иногда с выработко простейших управленческих решений на основе типовых стационарных схем.

Развитие самой вычислительной техники привело в последние годы к существе ному прогрессу в создании систем с элементами «искусственного» интеллекта, позв ляющих решать задачи, считающиеся интеллектуальными (творческими) для человек

В первую очередь это были системы, обладающие способностями к анализу конкретных ситуаций и самостоятельному поиску рациональных решений на основе баз знаний, содержащих не только важнейшие для данного процесса сведения, но и правила вывода для выработки необходимой информации. Близкие по характеру результаты были достигнуты в ряде отраслей при использовании экспертных систем, аккумулирующих знания ведущих специалистов в конкретных отраслях с правилами получения, а иногда и объяснения рекомендуемых решений.

Технологии с такими элементами можно назвать «интеллектуальными» информационными технологиями. Роль «интеллектуальных» технологий в перспективе будет возрастать. Такой вывод напрашивается в связи с построением информационного общества во всех передовых странах. В частности, об этом говорят успехи в автоматизации проектно-конструкторских задач, автоматизации управления предприятиями, начатые разработки по созданию «электронных» правительств, управлению космическими полетами, автоматизации военных операций, прогнозированию сложных процессов, управлению транспортными средствами на Земле из космоса и др.

Особо впечатляют «интеллектуальные» информационные технологии в создании шахматных программ, которые воплотили ряд основных достижений в развитии систем «искусственного» интеллекта и суперкомпьютеров. Качество их работы росло стремительными темпами за последние примерно 40 лет. Если в 1968 году на матче между шахматными программами СССР и США демонстрировалась игра на уровне 3 разряда, то в 2008 году сильнейшие из вычислительных систем для игры в шахматы демонстрируют результаты на уровне чемпиона мира. Увлечение ученых созданием шахматных программ базируется на уверенности, что полученные здесь результаты приведут к прорыву и в других областях деятельности человека, т.е. шахматы в вычислительной технике рассматриваются как подопытный кролик в медицине. Основная причина выбора такого объекта состоит в том, что правила игры формализованы, накоплен значительный багаж сыгранных гроссмейстерами шахматных партий и окончаний, создана теория стратегии и тактики игры, что позволило создать системы принятия решений и копирования лучших достижений в игре за счет громадной базы данных и знаний, а также быстродействия суперкомпьютера. Это достижение явилось плодом работы многих специалистов в области шахмат, теории принятия решений, «искусственного» интеллекта, программирования, передачи и обработки данных, сетей связи, математиков, электронщиков и многих других.

Быстрое развитие информационных и коммуникационных технологий стало оказывать сильное влияние на стиль работы, бизнеса, учебы и общения во всех странах. В наиболее развитых из них появились поддерживаемые государствами специальные программы по построению информационного общества. Информационным технологиям стала отводиться решающая роль в повышении производительности труда, улучшении качества новой продукции и сокращении сроков её разработки, завоевании лучших позиций на мировых рынках за счет улучшения динамики учета пожеланий клиентов и глобализации рекламы.

Остановимся более подробно на дистанционных информационных технологиях в управлении государством, коммерческими операциями и их роли в создании рабочих мест.

В основу любой дистанционно осуществляемой операции ставится транзакция. Под транзакцией будем понимать такое взаимодействие двух объектов, расположенных на расстоянии, когда цикл их взаимодействия включает запрос – выполнение задания – ответ. Такая трактовка пригодна и для описания дистанционных коммерческих операций,

когда под транзакцией понимается любая сделка между хозяйствующими субъектами, обладающая свойствами неделимости, согласованности, надежности, изолированности.

На основе понятия «транзакция» реализуется современная сетевая экономика, использующая бизнес-процессы, основой которых является обязательное взаимодействие субъектов через сети различного ранга (Интернет, интранет, экстранет), многочисленные специальные корпоративные сети и др. Бизнес-процесс в этом случае трактуется довольно широко: торговля, обмен идеями и информацией, выполнение совместных работ, выполнение целевого заказа, платежей и т.п.

К электронной коммерции будем относить любой вид деловых операций, предусматривающих использование современных информационных технологий и коммуникационных сред.

Увеличение в мире количества электронных коммерческих операций не может не затрагивать и экономику Республики Беларусь с учетом поставленных перед ней задач перехода на наукоемкие, энергосберегающие и материалосберегающие технологии и создание белорусской «силиконовой» долины, расширение экспорта.

Остановимся на отраслях, которые могут служить локомотивами для продвижения этих вопросов в Республике Беларусь. Так как любая торговля, инвестиции и кредитование осуществляются с участием банков, то в банковских операциях в Республике Беларусь, прежде всего, проявляются ростки новых отношений. Их объективным отражением является рост количества безналичных платежей, поступающих как от физических, так и юридических лиц, и как следствие – повышение количества документов в электронной форме. Юридической основой развития этих операций на государственном уровне являются такие важнейшие документы, как Законы Республики Беларусь «Об информации» и «Об электронном документе». Согласно этим законам документированная информация может выступать как объект права собственности в различных формах, включая и документы, подписанные электронной цифровой подписью. Законом устанавливается обязательная структура электронного документа (ЭД), включающая общую часть (информация, отражающая суть документа) и особенную часть (одну или несколько цифровых электронных подписей).

Названные законы открывают путь к использованию «безбумажных» систем документирования в различных отраслях деятельности и к равноправному обращению в юридическом плане как обычных документов, так и «электронных». Предполагается, что при использовании сертифицированных процедур электронной цифровой подписи электронные документы будут эквивалентны обычным и будут иметь такую же доказательную силу.

Однако требуется еще пройти значительный путь, чтобы обеспечить функционирование электронных документов по описанной схеме: создать сертификационные центры и центры распространения сертифицированных средств шифрования и электронной подписи, отработать процедуры подтверждения цифровой подписи. Поэтому пока в России и Республике Беларусь полноценное функционирование описанной схемы дополнительно гарантируется договорами между хозяйствующими субъектами о взаимном признании процедур электронной подписи и границах возможного ведения хозяйственных дел на основе электронных документов.

Подписание электронных документов, кроме того, тесно связано с вопросами сохранения коммерческой тайны и часто основывается на применении аналогичных математических методов шифрования информации. Действуя в этом направлении, Национальный банк Республики Беларусь издал инструкцию о порядке передачи служебной

информации ограниченного распространения в электронном виде по открытым каналам связи. Пока еще она при реализации имеет под собой договорную основу между банками РБ и Национальным банком РБ, который устанавливает в других банках свой программно-технический комплекс, который обеспечивает конфиденциальность передачи информации с клиентских рабочих мест других банков. Под управлением центрального банка проводится как установка клиентского программного обеспечения, так и генерация открытых и тайных личных ключей шифрования. Электронная цифровая подпись осуществляется на клиентском рабочем месте ответственным за него работником. Чтобы избежать злоупотреблений с повторами и перехватами сообщений, пользователем указывается системное время отправки документа и выполнения наложения электронной цифровой подписи (ЭЦП) на него. Подписанное зафиксированное сообщение отсылается клиенту, от которого получается подписанное и зашифрованное сообщение (квитанция) по факту приема, проверки достоверности и целостности принятого документа.

После расшифровки квитанции проверяется достоверность ЭЦП клиента-получателя, сверяется дата и время своего отправленного сообщения с датой и временем, указанными в квитанции. Расшифрованная квитанция сохраняется в качестве ЭД получателя, подтверждающего факт и время переданной информации. Спецификой обладает процедура выполнения электронных платежей и выдачи наличных денег физическим лицам через банкоматы и кассы с использованием электронных пластиковых карт. В РБ используются пластиковые карты, имеющие применение как внутри РБ, так и в других государствах.

Абсолютное большинство составляют пластиковые карты РБ для дистанционного доступа к банковским счетам клиентов, на которые в основном вносится заработная плата, хотя допускается и внесение средств из других источников. Главными базовыми компонентами здесь являются банкоматы, инфокиоски и централизованная платежная система для обслуживания счетов клиентов, как правило, в крупных населенных пунктах РБ для любого пользователя из другого или того же населенного пункта страны. В целом по РБ находится в обращении более миллиона банковских пластиковых карточек. На основе секретного кода клиенты получают доступ к своим счетам. Основным недостатком белорусских пластиковых карточек является отсутствие на них чипа (микропроцессора) и фотографии клиента, так как эти элементы повышают защищенность пользователя от кражи или потери карты, а также от использования ее другими лицами в магазинах и т.п.

Пока слабо распространены или отсутствуют такие услуги дистанционного банковского обслуживания, как: телефонный банкинг (перевод денег или их контроль на счет с обычного домашнего или мобильного телефона); видео-банкинг (общение клиента со служащими банка через специальные телемониторы); компьютерный банкинг (доступ к счету с помощью персонального компьютера путем прямого соединения с сетью банка); Интернет-банкинг (управление банковским счетом через Интернет); SMS-банкинг (услуга для владельцев сотовых телефонов); TV-банкинг (совершение банковских операций с помощью интерактивного телевидения). Слабое развитие этих видов услуг сдерживается и недостаточным техническим, программным и информационным обеспечением, соответствующим стандартам ISO.

Кроме того, нужна и более глубокая проработка ведения хозяйственной деятельности на основе дистанционного взаимодействия как между самими банками, так и хозяйствующими субъектами.

Назрел вопрос о создании дружественной для пользователя системы банкоматов, собственниками которых являются разные банки, чтобы клиент мог обслуживаться без

дополнительных комиссионных сборов в «чужом» банкомате. Первый такой пример по-
дали Беларусбанк и Промстройбанк. Ключевыми вопросами для всех банков здесь яв-
ляются зарядка кассет банкоматов наличными деньгами и амортизация оборудования.
На наш взгляд, нельзя не воспользоваться тем, что электроника позволяет отследить,
чей клиент пользовался банкоматом, и какие суммы он взял. На основе клиринговых
расчетов банки могут компенсировать общую часть затрат на выдачу наличных денег, а
остальную часть возместить. Надо решать также вопрос и о компенсации затрат на экс-
плуатацию оборудования: они должны компенсироваться не по суммам выданных денег,
а по количеству транзакций (обращений) «чужого» пользователя. Эта проблема может
решаться аналогичным образом и при развитии других сетевых услуг.

На этом примере четко прослеживается необходимость корпоративных действий
всех банков в использовании, приобретении и разработке технических, программных,
коммуникационных и информационных средств.

Именно использование информационных технологий позволяет объективно оцени-
вать затраты каждого участника корпорации на основе всестороннего учета стоимости
каждой транзакции и использования технического программного и информационного
обеспечения. В частности, можно компенсировать друг другу затраты, связанные с за-
рядкой кассет наличностью в банкоматах, количеством транзакций по конкретному бан-
комату для учета амортизационных затрат и аренде или использовании площадей для
банкомата, вкладу по поддержанию информационных систем, электроэнергии и т.д.
Аналогичный подход возможен и при обмене информацией и ее получении из общей ба-
зы данных и т.д. Решение такого рода проблем поднимает вопрос о совершенствовании
систем автоматизации деятельности банков Республики Беларусь с учетом мировых
тенденций.

Новая организация труда работников предприятий и государственного аппарата
позволяет в ряде случаев использовать формы дистанционной работы сотрудников.

К дистанционной работе будем относить ту, которая выполняется работником
предприятия с использованием информационно-вычислительной техники и средств
коммуникаций всегда или время от времени вне пределов предприятия. Ведущим фак-
тором здесь является наличие связи работника, его рабочего места с предприятием че-
рез электронные средства связи.

Дистанционная работа имеет много разновидностей в зависимости от местонахожде-
ния рабочего места и режима его использования. Чаще выделяют следующие ее формы:

- рабочее место находится в квартире сотрудника;
- рабочее место подвижное (в транспорте), и связь с предприятием реализуется
через мобильные средства (обычно телефон);
- рабочее место находится на территории заказчика;
- рабочее место вынесено на территорию информационного центра (дистанцион-
ный сервисный центр, биржа и т.п.);
- рабочее место в определенные дни и часы находится на предприятии, а в ос-
тальное время может быть мобильным, домашним и т.п.

Такие новые формы работы являются удобными для предприятия и работника, но
порождают и ряд проблем, связанных с контролем деятельности работника, сохранен-
ем конфиденциальной информации, оплатой труда, возмещением расходов работни-
по организации рабочего места и платежам за средства коммуникации, оплатой поме-
щения и временного использования компьютера работника, особенностями налогоо-
ложения и др.

Кроме того, возникает и ряд правовых проблем: следует регулировать отношения между работником и работодателем, вопросы по налогам за двоякое использование оборудования для домашнего быта и предприятия и т.п.

Остановимся на некоторых из этих особенностей, когда рабочее место располагается в квартире работника (на дому). В этом случае работник может иметь собственный компьютер с возможностью подключения к сетям связи. В этом случае общение с работодателем идет путем передачи электронных документов, а иногда и по телефону или электронной почте. В таких случаях контакты с коллегами и начальником носят ограниченный характер. Эта форма работы положительно воспринимается работником при необходимости воспитывать детей, при случаях заболеваний в семье, при необходимости иметь гибкий график работы в течение дня, недели и т.п.

Иногда встречается и промежуточный вариант телеработы, когда рабочее место может находиться в квартире работника или на предприятии с регламентацией времени пользования того и другого места. Этот вариант может оказаться более приемлемым для работодателя, так как можно регламентировать время контактов работника с коллегами и его руководителем.

Такого рода деятельность может иметь место и в условиях Республики Беларусь, например, выполнение переводов, программирование, сбор заказов и т.п.

Мобильные виды дистанционной работы в РБ вполне возможны для сотрудников с разъездным характером работы или сервисным обслуживанием по вызову.

Если исходить из материалов исследований Евросоюза, проведенных Фраунгоферовским институтом (ФРГ) путем опроса предприятий и органов власти, в 2000 г. имелась потребность в 875 000 дистанционных рабочих мест, из которых 500 000 относились к мобильной работе, 350 000 – к работе на альтернативной основе частично дома и на предприятии, 22 000 – к чисто дистанционной работе и 3 000 – к прочим видам работ.

Широко распространилась дистанционная работа в США, где примерно 7,6 миллиона человек работают полностью на дому и 9,2 миллиона человек работают время от времени по дистанционным схемам работы.

По оценкам немецких специалистов на базе анкетных опросов отмечается повышение производительности труда на дистанционных рабочих местах на 45% и более, а по данным фирмы IBM на ее дочерних предприятиях в Германии также отмечается повышение производительности труда.

В Евросоюзе планируется инвестировать образование нескольких миллионов дистанционных рабочих мест. Попытки оценить объемы этих инвестиций наталкиваются на ряд трудностей, связанных со спецификой работы такого рода на каждом конкретном предприятии. Тем не менее по отдельным элементам, обеспечивающим такую работу, были сделаны прикидочные расчеты, они охватывали такие аспекты, как средние стоимости программного обеспечения, компьютеров и аппаратных средств к ним, реализуемые технологии передачи данных, создание рабочей среды на дистанционном рабочем месте, обеспечение надежности информационных технологий и конфиденциальности данных и документов, борьбу с вирусами, возможные налоги и издержки производства. В зависимости от полноты набора этих элементов и загрузки коммуникаций и оборудования стоимость одного дистанционного рабочего места можно оценить от 3 500 до 7 000 евро в год.

Отличается некоторой спецификой и оплата труда дистанционных работников, а также выплата компенсаций за использование собственного оборудования и жилой площади. Проблемными вопросами являются те, которые касаются разграничения за-

трат работника на нужды предприятия и собственные, а также связанные с ними налоговые платежи.

Интерес к дистанционной работе обычно проявляют высококвалифицированные специалисты (программисты, инженеры, конструкторы, экономисты, переводчики), которые склонны к самостоятельной работе, требующей их знаний и способностей, инициативы. По оценкам министерства экономики ФРГ, реальное число потенциальных дистанционных работников в Евросоюзе 8 миллионов, из которых только в Германии около 2,5 миллионов.

Особая роль в дистанционной экономической работе отводится маркетинговым исследованиям и рекламе. Современные маркетинговые исследования существенно опираются на средства коммуникации, Интернет и различные региональные и отраслевые базы данных, содержащие большой объем информации практически по всем областям знаний и крупнейшим фирмам, на огромное количество рекламных материалов.

Учитывая, что маркетинговые исследования часто можно оформить как специальный заказ, они могут быть источником дистанционной работы разового плана. Аналогичные работы могут выполняться по поиску возможных путей приобретения или продажи различной «интеллектуальной» собственности (патенты, промышленные образцы, товарные знаки, произведения литературы, музыки и т.п.).

Следует обратить внимание на сложность выполнения такого рода работ из-за трудностей поиска необходимой информации. Более того, все чаще заходит речь о патентовании средств, позволяющих эффективно решать поисковые задачи в различных сетях и Интернете. В частности, фирма Amason.com предложила однощелчковый поиск электронной мышью для выполнения закупочных операций.

Рекламные заказы в Интернете становятся сложными и дорогостоящими из-за трудностей вывода потребителя на потенциального изготовителя товара или предоставляющую услуги фирму. В связи с этим только мощные фирмы могут найти средства на рекламу, носящую глобальный характер. Специфика рекламы на базе сетевых операций содержит такой элемент, как использование медиасредств: в частности, возможность показывать короткометражные фильмы и ролики о рекламируемом объекте в действии.

Кроме того, работу многих программных и информационных продуктов можно показывать, пользуясь их демонстрационными версиями, передавая их во временное или даже постоянное пользование, чтобы потребитель смог оценить потенциальные возможности продукта для своих условий. Первые попытки таких рекламных действий уже имеются и в Республике Беларусь. По аналогии можно демонстрировать и отдельные элементы учебного процесса при рекламе дистанционных форм обучения, включая схемы проведения отдельных опытов, выполнения и оценки заданий и т.д. Такого рода деятельность тоже может осуществляться как дистанционная работа.

Начинает распространяться такой вид рекламы, когда пользователю на бесплатной основе предоставляется сам продукт, но с ограничением на количество раз или времени использования. По такого рода продуктам облегчаются и коммерческие операции по их установке у заказчика с безналичными платежами по сетям связи. Обычно продавец в таких случаях имеет программы для анализа особенностей оборудования заказчика и сообщает ему особый код. Перенести же объект в обход продавца на другое оборудование невозможно или крайне сложно.

Из перечисленных направлений развития исследований и прикладных работ следует выделить особо важные ввиду их всеобщей применимости: защита и безопасность

данных (в Германии, например, по этим вопросам издается специальный журнал «Datenschutz und Datensicherheit»; общие аспекты создания и функционирования дистанционных рабочих мест (экономические, правовые и ИТ-вопросы); использование автоматизированных средств идентификации личности с учетом уникальных физических особенностей человека; создание средств для современной рекламы в сетях с использованием медиа-средств и демонстрационных версий; автоматизация дистанционной оплаты коммунальных услуг, получения различных справок и документов, оплаты покупок, предварительных заказов на различные услуги, управления банковскими счетами и т.п.

Эти вопросы должны стать предметом общегосударственных и отраслевых программ, так как они должны решаться системно и требуют затрат, которые не дают мгновенной окупаемости для отдельного предприятия.

3.8. Защита информации и создание безбумажных систем документирования

С каждым годом повышается роль средств, обеспечивающих информационную безопасность различных технологий, использующих компьютеры. Ранее информационная безопасность ограничивалась сферами разведки, дипломатии, иногда коммерческих операций. Сейчас приходится обеспечивать информационную безопасность практически для всего взрослого населения страны. Это объясняется, прежде всего, выполнением дистанционных коммерческих операций от простейших платежей за квартиру через электронное оборудование до сложнейших расчетов через систему международных банков, а также защитой прав человека на охрану информации о нем как о конкретной личности в связи с появлением различных электронных картотек (медицинских, налоговых, правоохранительных и т.п.). Последнее привело к необходимости использовать сложнейшие информационные технологии в защите самих компьютеров и информации, хранящейся в них и передаваемой по сетям связи.

Успехи в защите информации достигаются там, где используется организованная система, опирающаяся на широкий комплекс с взаимодействием различных средств: привычных всем замков и сейфов, охрану объекта людьми, хитроумные системы на основе лазерной техники и различных камер наблюдения и, наконец, программные методы. Во многих случаях программные методы являются основными, так как с их помощью решается главная задача – защита от несанкционированного доступа даже украденной (скопированной) информации, а также авторизация (подпись) документов, существующих только на машинных носителях, и дистанционное управление банковским счетом.

Математической основой этих методов является теория криптографии (тайнописи). Ее истоки уходят в очень древние времена. На первых порах она часто использовала довольно простые шифры замены знаков (каждого в тексте или группы) путем подстановки другого знака методом, который был известен получателю и допускал обратную операцию. Широко известен среди них метод Цезаря, который задавал в каждом сообщении шаг сдвига для буквы в тексте: вместо истиной буквы записывалась другая, отстоящая от нее в алфавите на заданный шаг в закольцованном алфавите. Например, алфавит {А, В, С, Д, Е, F, R}, истинное сообщение ARFA, закодированное с шагом 3 будет – Д С В Д.

Секрет такого шифра в состоянии раскрыть даже школьник. Поэтому теория кодирования пошла по пути усложнения трудностей в прочтении закодированных текстов таким образом, чтобы, даже зная метод кодирования, но, не зная ключа (в нашем примере шаг сдвига), с помощью современной техники за практически разумное время (жизнь человека) нельзя было найти ключ к чтению текстов.

Компьютерный этап криптографии начался с трудов американского математика Клода Шеннона. Он выделил два общих принципа, использующихся в практическом шифровании: рассеивание (влияние одного знака на шифротекст) и перемешивание (создание трудностей в использовании статистических свойств из-за повтора знаков или характера сообщений, например, стандартные донесения с датами, наименованиями типов вооружений и их количества и т.п.). Практические трудности здесь стали заключаться в том, что надо не только затруднить раскрытие истинного сообщения, но и само шифрование сделать достаточно легким.

Таким образом, для решения этих задач пришли к идее сами ЭВМ использовать в шифровании и дешифровании текстов под управлением человека, который остался лишь носителем небольшой секретной информации (ключей).

Постепенно созрели идеи и о необходимости использования государственных стандартов шифрования, чтобы уверенно и безопасно пользоваться программами и алгоритмами, прошедшими экспертизу в специальных государственных органах. В мире первым таким стандартом стал DES (Data Encryption Standart, США). Он опирался на открытый для всех шифроалгоритм, основанный на реализации принципов рассеивания и перемешивания. В нем открытый текст, криптограмма и ключ представлялись в виде двоичных последовательностей длиной соответственно 64, 64 и 56 битов. Спустя более 30 лет после его опубликования, с помощью специального компьютера «Большой взлом» (Deep Crack) за 56 часов при переборе $18*10^{16}$ комбинаций удалось осуществить взлом контрольной шифровки. В связи с этим на смену DES пришел AES (Advanced Encryption Standart – улучшенный стандарт шифрования), в котором для перебора всевозможных ключей (2^{256}) потребуется столько операций, которые практически нельзя осуществить даже на специальном комплексе из многих суперкомпьютеров за приемлемое время.

Аналогичного типа стандарты стали использоваться в качестве ГОСТов и в других странах (Россия и др.). Они относятся к классу систем с симметричным алгоритмом шифрования, т.е. один и тот же ключ используется для шифрования и дешифрования текста. В этом случае возникла другая сложность: как передать адресату ключ по открытым каналам связи без риска его перехвата. Поэтому дальнейшее усовершенствование криптосистем пошло в направлении создания двухключевых систем: их первый ключ публикуется для применения всеми пользователями для шифрования данных, а для их расшифровывания имеется секретный второй ключ, который нельзя получить из первого. Суть математических преобразований для этих целей опирается на теорию функций с «лазейкой», когда $y=f(x)$ легко вычисляется, а обратная функция $x=f(y)$ невычислимая без знания дополнительной информации (лазейки). Простейший пример такого типа представляет вычисление целочисленной функции $y=x \bmod n$, когда ее результатом является остаток от деления целого числа на целочисленный модуль n. Например, $y=12 \bmod 7=5$, но обратная операция отыскания x как функции от остатка 5 не выполняется однозначно (один и тот же остаток при делении на 7 могут дать 5, 12, 19,...) Неизвестное x можно вычислить лишь при знании «лазейки», например, номера числа в ряду 5, 12, 19, ...

Покажем реализацию одноключевой системы такого типа с открытой передачей ключей на основе функций с лазейкой. Для открытого распространения ключей Диффи и Хелман предложили использовать функцию $f(x)=a^x \pmod p$, где p – очень большое простое число, x – целое от 1 до (p-1) число, a – целое число из полуинтервала $(1<a\le p)$, степени которого в некотором порядке равняются 1, 2, 3,... (p-1) по модулю p. Предполагается, что всем пользователям закрытой системы известны a и p (сообщает админист-

ратор системы при регистрации). Пользователь i случайным образом выбирает целое число x_i ($1<x_i<(p-1)$) и держит его в секрете. Потом он вычисляет $y_i=a^{x_i}(\mathrm{mod} p)$ и помещает его в открытый для пользователей сети справочник. При желании установить секретную связь с другим пользователем сети j, он берет его число y_j и с помощью секретного ключа x_i вычисляет число $Z_{ij}=(y_j)^{x_i}(\mathrm{mod}\ p)$. Аналогично j вычисляет Z_{ji} (оно равно Z_{ij}), и далее это число они могут использовать как ключ для обмена зашифрованными сообщениями.

Пример: (p=7, a=3 x=1, 2, 3, 4, 5, 6,)

$X_i = 3$ (секретное число пользователя *i*), $y_i = 3^3\ (mod7) = 6$ (передается в справочник);

$X_j = 5$ (секретное число пользователя j), $y_j = 3^5(\mathrm{mod}7) = 5$;

$Z_{ij} = 5^3\ (\mathrm{mod}7) = 125(\mathrm{mod}7) = 6$; $Z_{ij} = 6^5(\mathrm{mod}7) = 7776(\mathrm{mod}7) = 6$

Предположим, цифра 6 будет означать страницу в каком-то справочнике или книге, где содержатся функции для шифрования текстов. (Администратор может выдавать им такой справочник и время от времени его менять; важно, что их ключ (6) остается секретным и для администратора сети, так как «6»не передается по открытому каналу).

Информационная безопасность вычислительных сетей различного ранга от локальных (предприятия) до общегосударственных и корпоративных может совершенствоваться всегда вместе с развитием науки и техники. Поэтому встает вопрос о стоимости систем защиты информации. На практике используется критерий, опирающийся на оценку стоимости «взлома» системы. Если «взлом» системы обходится дороже украденной информации, то логично защиту считать достаточной, так как получение таких данных законными путями будет дешевле.

Чтобы затруднить раскрытие сообщений, иногда применяют не только методы шифрования, но и методы стеганографии, задачи которых – скрыть сам факт пересылки секретного сообщения. Для этой цели используются два типа файлов: само сообщение (первый закодированный файл) и контейнер (второй файл), задача которого скрыть содержимое основного файла, которое потом с помощью специального ключа извлекается из вспомогательного файла, выглядящего как безобидное сообщение (например, открытка с поздравлением к 8 Марта и т.п.)

Литература

[2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22, 25, 27, 28, 29, 30, 31, 33, 34, 37, 41, 42, 43, 44, 45]

Контрольные вопросы

1. В чем заключается особенность схемы эксперимента Тьюринга для оценки меры интеллектуальности системы?
2. В чем заключается особенность моделей ситуационного управления при описании объекта и создании схемы управления им?
3. Каковы основные особенности построения общего решателя задач?
4. Какие перспективы имеют методы диалогового решения задач и специфика их использования?
5. Какова специфика создания и использования экспертных систем?
6. В чем состоит особенность формального элемента нейронной сети?
7. Какова роль обучения нейронных сетей?
8. Чем отличается нейрокомпьютер от традиционной ЭВМ?
9. Какое принципиальное отличие имеет интеллектуальная информационная технология от традиционно используемой в промышленности?

10. За счет чего достигается эффективность работы современных шахматных программ?

11. Что является основой для создания дистанционных рабочих мест и средств обучения?

12. В чем состоит отличие документа с электронной подписью от обычного?

13. В чем заключаются особенности диагностики технических систем и живых организмов.

14. Какие преимущества дает автоматизация управления движением автотранспорта?

ЗАКЛЮЧЕНИЕ

Перспективы внедрения в повседневную жизнь человека различных систем «искусственного» интеллекта грандиозны:

- стремительное нарастание количества бытовой техники с программным управлением и возможностями принятия отдельных решений без непосредственных контактов с хозяевами;

- от года к году увеличивается количество услуг, связанных с повседневной жизнью человека (оплата бытовых расходов, заказ билетов, гостиниц и других услуг, выполнение действий по отношению к отсутствующему или спящему человеку, обеспечение его безопасности, компоновка и запись кинофильмов, телерепортажей, идущих одновременно в заданном порядке, для последующего просмотра с автоматическим созданием их краткого дайджеста и т.д.);

- развитие систем «электронных» правительств, обеспечивающих контакты избирателей по получению информации о постановлениях и готовящихся проектах;

- обеспечение систем безопасности в массовых мероприятиях мгновенный анализ результатов видеонаблюдения и передвижения людей в различных видах транспорта;

- создание гибких автоматизированных производств с возможностями выполнять индивидуализацию типовых изделий под требования заказчика;

- различные формы информационного обеспечения по индивидуальным заказам и т.д.

Можно специально выделить процессы обучения, так как их совершенствование коренным образом должно повлиять на индивидуальную подготовку отдельного специалиста:

- внедрение дистанционных систем обучения с индивидуальный скоростью подготовки для каждого учащегося;

- использование баз знаний во многих областях деятельности на уровне «экспертных» систем, когда их пользователь не будет иметь очень высокой квалификации в данной области, но обеспечит получение хороших общих решений в рамках поставленных ему задач;

- применение в преподавании вспомогательных систем, ускоряющих процесс обучения и позволяющих сосредоточиться учащимся на главных элементах (например, автоматическая запись исполняемой музыки на ноты, различные подсобные вычисления).

ЛИТЕРАТУРА

1. Айзерман, М.А. Логика. Автоматы. Алгоритмы / М.А. Айзерман [и др.]. – М.: Физматгиз, 1963. – 556 с.

2. Артоболевский, И.И. Автоматическая медицинская информационная система отыскания клинического прецедента / И.И. Артоболевский, А.А. Вишневский, М.Л. Быховский // Экспериментальная хирургия и анестезиология». – 1962. – № 3. – 40-48 с.

3. Афанасьев, М.Б. Технические средства организации дорожного движения / М.Б. Афанасьев, Ю.А. Кременец, М.П. Печёрский. – М: Академкнига, 2005. – 352 с.

4. Балабанов, И.Т. Электронная коммерция. – СПб.: Питер, 2001. – 336 с.

5. Банкаускі веснік. Інфармацыйны выпуск. – Мінск: 2004. – № 18/275.

6. Барановская, Т.П. Информационные системы и технологии в экономике / Т.П. Барановская [и др.]. – М.; ФиС, 2003. – 416 с.

7. Бирюков, Б.В. Машина и творчество: Результаты, проблемы, перспективы / Б.В. Бирюков, И.Б. Гутчин. – М.: Радио и связь. 1982. – 152 с.

8. Врубель, Ю.А. Потери в дорожном движении. – Мн.: БНТУ, 2002. – 422 с.

9. Глушков, В.М. Основы безбумажной информатики. – М.: Наука Ф.М., 1982. – 552 с.

10. Головко, В.А. Нейронные сети: обучение, организация и применение. Кг.4 Учебное пособие для вузов / В.А. Головко – М.: ИПРЖР, 2001. – 256 с.

11. Головко, В.А. Нейроинтеллект: теория и применение. Книга 1: Организация и обучение нейронных сетей с прямыми и обратными связями. – Брест, 2009.

12. Головко, В.А. Нейроинтеллект: теория и применение. Книга 2: Самоорганизация, отказоустойчивость и применение нейронных сетей. – Брест, 2009.

13. Гринберг, А. С. Информационный менеджмент / А.С. Гринберг, В.А. Король. – Минск: Акад. управл. при Президенте РБ, 2002. – 479 с.

14. Гулин, Н.В. Бизнес-офис предприятия. – Минск: БГЭУ, 2004. – 1-279 с.

15. Дрю, Д. Теория транспортных потоков и управление ими. – М: Транспорт 1972. – 521 с.

16. Иносэ, Х. Управление дорожным движением; перевод с английского / Х. Иносэ, Т. Хамада. – М.: Транспорт, 1983. – 346 с.

17. Искусственный интеллект. Кн.1: Системы общения и экспертные системы / Под ред. В. Попова. – М.: Радио и связь, 1990. – 420 с.

18. Климченя, Л.С. Электронная коммерция. – Минск: Выш. шк., 2004. – 191 с.

19. Кременец, Ю.А. Технические средства регулирования дорожного движения / Ю.А. Кременец, М.П. Печёрский. – М: Транспорт, 1981. – 321 с.

20. Криницкий, Н.А. Алгоритмы и роботы. – М.: Радио и связь, 1983. – 168 с.

21. Криницкий, Н.А. Программирование и алгоритмические языки / Н.А. Криницкий [и др.]. – М.: Наука, 1979. – 509 с.

22. Комарцова, Л.Г. Нейрокомпьютеры: ученое пособие для вузов / Л.Г. Комарцова, А.В. Максимов – Москва: Изд. МГТУ им. Н.Э. Баумана, 2004. – 400 с.

23. Костров, А.В. Основы информационного менеджмента. – М.: ФиС. – 336 с.

24. Кузубов, В.Н. Принятие оптимальных решений в экономике и менеджменте с применением компьютерных технологий. – Москва: Юнити, 2002. – 144 с.

25. Луканин, В.Н. Автотранспортные потоки и окружающая среда / В.Н. Луканин [и др.]. – М: Инфра-М, 1988. – 645 с.

26. Майоров, С.А. Электронные вычислительные машины. Введение и специальность: уч. пособие для вузов / С.А. Майоров, Г.И. Новиков. – М.: Высш. шк., 1982. – 175 с.

27. Матюшков, Л.П. Основы машинной математики / Л.П. Матюшков, А.А. Лихтарович – Минск: Нар. Асвета, 1988. – 240 с.

28. Матюшков, Л.П. Перспективы информационных технологий в развитии дистанционных рабочих мест и коммерческих операций. «Вучоныя запіскі БрДУ». – Брест БрДУ, 2006. – Т.2. – Ч.1. – 107-115 с.

29. Никитенко, П.Г. Ноосферная экономика и социальная политика: стратегия инновационного развития. – Минск: Белорус. Наука, 2006. – 479 с.

30. Пархоменко, П.П. Основы технической диагностики / П.П. Пархоменко, В.С. Сагомонян. – Москва, Энергоиздат, 1981. – 349 с.

31. Пасхин, Е.Н. Автоматизированная система обучения ЭКСТЕРН / Е.Н. Пасхин, А.И. Митин. – М.: Изд-во МГУ, 1985. – 144 с.

32. Попов, Е.П. Роботы и человек / Е.П. Попов, А.С. Ющенко. – М.: Наука, 1984. – 112 с.

33. Попов, Э.В. Общение с ЭВМ на естественном языке. – М.: Наука, 1982. – 360 с.

34. Поспелов, Г.С. Искусственный интеллект – прикладные системы / Г.С. Поспелов, Д.А. Поспелов. – М.: Знание, 1985. – 46 с.

35. Танаев, В.С. Синтез схем алгоритмов выбора решений / В.С. Танаев, М.П. Поварич. – Минск: Наука и техника, 1974. – 112 с.

36. Технические средства диагностирования: справочник / Под редакцией В.В. Клюева. – М.: Машиностроение, 1989. – 745 с.

37. Тихонов, А.Н. Вводные лекции по прикладной математике / А.Н. Тихонов, Д.П. Костомаров. – М.: Наука, 1984. – 192 с.

38. Трахтенброт, Б.А. Алгоритмы и вычислительные автоматы. – М.: Сов. радио, 1974. – 199 с.

39. Успенский, В.А. Машина Поста. – М.: Наука, 1979. – 95 с.

40. Царев, В.Ф. Электронная коммерция / В.Ф. Царев, А.А. Канторович. – СПб.: Питер, 2002. – 320 с.

41. Шуть, В.Н. Оптимизация управления автотранспортом перед светофором в улично-дорожной сети города / В.Н. Шуть, С.В. Анфилец [и др.] // Вестник БрГТУ. – 2008. – № 5: Физика, математика, информатика. – 110-112 с.

42. Шуть, В.Н. Тестовый контроль электрического монтажа // Вопросы радиоэлектроники, серия ЭВТ. – 1981. – № 14. – 41-48 с.

43. Шуть, В.Н. Диагностика электрического монтажа // Вопросы радиоэлектроники, серия ЭВТ. – 1982. – № 12. – 36-45 с.

44. Якимахо, А.П. Управление объектами интеллектуальной собственности в РБ / А.П. Якимахо. – Минск: Амалфея, 2005. – 472 с.

45. Янсен, Ф. Эпоха инноваций. – М.: ИНФРА-М, 2002. – 308 с.

ПРИЛОЖЕНИЕ

Глоссарий

Бейсик	– популярный алгоритмический язык программирования высокого уровня, первоначально созданный для целей обучения программированию. Язык имеет относительно простой синтаксис, что облегчает его быстрое освоение.
Схема алгоритма	– функционально-ориентированное графическое изображение, с помощью которого, используя текст и специальные символы, описывают последовательность шагов процесса во времени, связи рассматриваемой системы с внешней средой и ветвление процесса.
Графический дисплей	– дисплей, обеспечивающий представление графической информации.
Графопостроитель	– устройство вывода, предназначенное для представления данных в виде графического изображения на бумаге.
Джойстик	– устройство для ручного управления движением курсора на экране дисплея.
Дисплей	– устройство, обеспечивающее визуальное представление цифровой, алфавитно-цифровой и (или) графической информации на экране электронно-лучевой трубки, в плазменных панелях, на жидких кристаллах, светодиодах и т.п. в форме, удобной для оператора.
Интеллект	– способность мышления, рационального познания.
Интеллектуальный видеотерминал	– видеотерминал, включающий в свой состав процессор и память, доступную для программирования пользователем.
Интерпретатор	– обслуживающая программа, осуществляющая пооператорную трансляцию и выполнение исходной программы.
Информатика	– отрасль науки, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в различных сферах человеческой деятельности.
Информационная технология	– технология, использующая в качестве своего носителя ЭВМ как инструмент управления орудиями производства.
Интеллектуальная информационная технология	– информационная технология, использующая при своей реализации творческие решения или их элементы для управления орудиями производства.
Информация	– сведения, совокупность каких-либо данных, знаний и т.п.
Исчисление	– способ задания множества путем указания исходных элементов и правил вывода для построения новых элементов из исходных и уже построенных.
Итерация	– результат многократного применения какой-либо математической операции.
КОБОЛ	– язык программирования высокого уровня, ориентированный на решение экономических и коммерческих задач.
Команда машинная	– элементарное предписание для ЭВМ, определяющее действия машины в течение некоторого отрезка времени (содержит указание операции, адреса операндов и другие служебные признаки).

Компилятор	– обслуживающая программа, выполняющая трансляцию на машинный язык программы, записанной на исходном языке программирования.
Локальная сеть микро – ЭВМ	– сеть микро-ЭВМ, сосредоточенная на ограниченной территории (в пределах одного или нескольких зданий) и не использующая средств связи общего назначения.
Маркер (курсор)	– специальный знак на экране дисплея для указания определенных позиций или элементов.
Манипулятор	– приспособление для выполнения вспомогательных операций по захватыванию и перемещению предметов с управлением по командам.
Меню	– список альтернатив, появляющихся на экране дисплея и предлагаемых для выбора пользователю.
Моделирование	– исследование объектов на их моделях.
Модель	– описание объекта (или системы), используемое в качестве его заменителя.
Нейрокомпьютер	– вычислительная система, состоящая из множества нейронных элементов и базирующаяся на нейросетевых принципах функционирования.
Нейронная сеть	– совокупность нейронных элементов и связей между ними. Лежит в основе организации как биологических объектов, так и искусственных вычислительных систем.
Обучающая машина	– устройство, предназначенное для реализации обучающих программ (учебного материала, в котором описываются подлежащие усвоению знания, умения и навыки, а также способы их формирования).
Оператор	– допустимая в языке программирования синтаксическая конструкция, отражающая определенное действие в программе (присвоение значения, передачу управления и т.п.).
Операционная система	– комплекс взаимосвязанных управляющих и обслуживающих программ, обеспечивающих автоматическое управление вычислительными процессами и ресурсами ЭВМ при решении задач.
Отладка программ	– выявление ошибок в программе и доказательство ее правильности (по крайней мере, путем решения ряда контрольных задач с известными заранее результатами).
Параметр	– величина, входящая в формулы и выражения, значение которой является постоянным в пределах одной задачи.
Персональный компьютер	– небольшая по размерам и стоимости универсальная микро-ЭВМ, предназначенная для индивидуального пользователя.
Подпрограмма	– часть программы, допускающая многократное обращение к ней из различных точек программы.
Пользователь	– лицо, использующее данное вычислительное устройство для выполнения необходимых ему работ.
Программа	– последовательность инструкций, реализующих алгоритм. Программы обычно могут быть написаны: а) в машинном коде, который непосредственно выполняется процессором; б) на языке типа ассемблер; в) на языке высокого уровня.

Промышленный робот	– программируемый многофункциональный манипулятор, предназначенный для перемещения материалов, деталей, инструмента или специализированных устройств по переменным программируемым траекториям для выполнения широкого круга задач.
Процедура	– порядок выполнения ряда последовательных действий.
Процедура рекурсивная	– процедура, в описании которой содержится обращение к ней самой.
Распределенная система	– система обработки данных, в которой отдельные функции обработки выполняются независимыми устройствами.
Речевой ввод	– ввод информации в ЭВМ с помощью голоса. В этом случае ЭВМ реализует процесс распознавания речи (либо отдельных слов, либо фраз).
Робот	– способная действовать целенаправленная система управления и переработки информации, оборудованная датчиками восприятия информации о внешней среде и исполнительными механизмами.
Световое перо	– устройство, используемое для указания элемента изображения на экране графического дисплея.
Семантика алгоритмического языка	– толкование содержательного смысла единиц алгоритмического языка.
Сеть ЭВМ	– система соединенных между собой и обменивающихся информацией ЭВМ.
Синтаксис алгоритмического языка	– совокупность правил однозначного описания содержания алгоритмов в алгоритмическом языке.
Система	– совокупность частей, связанных общей функцией.
Система команд	– полный набор всех инструкций, допустимых в машинном языке данной ЭВМ.
Системное программное обеспечение	– набор обслуживающих программ, предназначенных для трансляции, редактирования, отладки и загрузки прикладных программ пользователя.
Структура данных	– упорядоченное множество элементов данных (вектор, матрица и др.)
Технология	– наука о способах воздействия на сырье, материалы, полуфабрикаты и информацию соответствующими орудиями производства.
Транслятор	– программа, предназначенная для перевода описаний алгоритмов с одного формального языка на другой (обычно с алгоритмического на машинный).
Файл	– последовательность размещаемых на внешних ЗУ записей, рассматриваемая в процессе пересылки и обработки как единое целое.
Численный метод	– метод приближенного или точного решения математических задач, основанный на построении конечной последовательности действий над конечным множеством чисел.
Эмуляция	– имитация функционирования одной системы средствами другой системы без потери функциональных возможностей или искажения получаемых результатов.
Язык алгоритмический	– средство точного формулирования вычислительных процессов для их последующей реализации на автоматических вычислительных машинах.
Язык ассемблера	– символический язык программирования, структура операторов которого определяется форматами команд и данными машинного языка.
Язык высокого уровня	– язык программирования, средства которого допускают описание проблемы в наглядном, легко воспринимаемом виде.

ПРИЛОЖЕНИЕ Б

Министерство образования Республики Беларусь
УО «Брестский государственный технический университет»

УТВЕРЖДАЮ
Ректор Брестского государственного
технического университета

_____ П.С. Пойта
«__» _____ 2009 г.
Регистрационный № УД - ___/баз

Основы искусственного интеллекта

Учебная программа для специальности:
1-40 03 01 "Искусственный интеллект"

2009 г.

Пояснительная записка

Цель курса «Основы искусственного интеллекта» - ознакомить студентов с основными результатами и методами, применяемыми в процессе овладения специальностью «Искусственный интеллект».

Основными задачами изучения дисциплины являются:

– усвоение современных тенденций в интеллектуализации развития современных вычислительных систем;

– усвоение процессного (пооперационного) подхода в описании современных систем управления и качества в соответствии с международными стандартами ИСО;

– получение представления об организации гибких алгоритмов и производств, включая самоадаптацию;

– ознакомление с первичными сведениями о нейросетях и нейрокомпьютерах;

– ознакомление с принципами описания формальных систем и интеллектуальных баз знаний и экспертных систем; защиты информации.

В результате изучения дисциплины студент должен знать и овладеть:

– свойствами алгоритмов;

– нормальными алгоритмами;

– методами описания автоматов и использования их в робототехнике;

– формальными языками для описания простейших систем и алгоритмических языков;

– принципами построения баз данных и знаний, СУБД и поисковых систем с грамматикой;

– общими подходами к созданию нейрокомпьютеров и систем с элементами искусственного интеллекта, интеллектуальных информационных технологий и безопасных систем обработки информации – понятиями об электронной подписи, безбумажном документировании.

Примерный тематический план по дисциплине «Основы искусственного интеллекта»

№	Наименование темы	лекции	практические занятия	самостоятельная работа
1	Алгоритмы и вычислительные машины	12	8	10
2	Элементы теории формальных языков	10	2	8
3	Системы искусственного интеллекта в решении прикладных задач	12	6	10
	ИТОГО	34	16	28

Содержание лекционного материала по дисциплине «Основы искусственного интеллекта»

Тема 1. Алгоритмы и вычислительные машины.

Автоматы и алгоритмы. Численные и логические алгоритмы. Общие свойства алгоритмов. Средства описания алгоритмов. Выбор численных методов реализации алгоритмов. Связность алгоритмов. Ассоциативные исчисления и нормальный алгоритм Маркова. Машина Поста и программирование на ней. Конечные автоматы и роботы. Задания конечных автоматов таблицами, графами и матрицами.

Тема 2. Элементы теории формальных языков. Синтаксис и семантика формальных языков. Нотация Бэкуса и ее использование в описании формальных языков. Использование формальных языков в поиске информации. Дескрипторные системы с грамматикой. Общие подходы к построению алгоритмических языков и трансляторов. Операционные системы, базы данных и знаний, СУБД.

Тема 3. Системы искусственного интеллекта в решении прикладных задач.

Проблемы создания искусственного интеллекта. Диалоговые методы решения задач. Экспертные системы. Нейронные сети и нейрокомпьютеры. Интеллектуальные информационные технологии. Защита информации и создание безбумажных систем документирования.

Содержание практических занятий

1. Составление численных и логических алгоритмов с демонстрацией их основных свойств.

2. Выбор машинно-ориентированных численных методов решения задач. Связность и восстанавливаемость алгоритмов.

3. Разработка программы сложения двух чисел на машине Поста.

4. Задание с помощью конечного автомата движения робота по контуру.

5. Построение с помощью нотации Бэкуса формального языка.

6-7. Построение логистической распределительной системы на основе задачи коммивояжера (4ч.).

8. Построение одноключевой системы кодирования информации на основе метода Диффи и Хелмана.

Литература

1. Головко, В.А. Нейронные сети: обучение, организация и применение. Кн.4 учебное пособие для вузов. – М.: ИПРЖР, 2001. – 256 с.

2. Матюшков, Л.П. Основы машинной математики / Л.П. Матюшков, А.А. Лихтарович – Минск: Нар. Асвета, 1988. – 240 с.

3. Комарцова, Л.Г. Нейрокомпьютеры: уч. пособие для вузов / Л.Г. Комарцова, А.В. Максимов. – Москва: Изд. МГТУ им. Н.Э. Баумана, 2004. – 400 с.

4. Провалов, В.С. Информационные технологии управления: уч. пособие. – М.: Флинта: МПСИ, 2008 – 376 с.

5. Климченя, Л.С. Электронная коммерция. – Минск: Выш. шк., 2004. – 191 с.

Дополнительная литература

6. Матюшков, Л.П. Перспективы информационных технологий в развитии дистанционных рабочих мест и коммерческих операций. «Вучоныя запіскі БрДУ». – Брест: БрДУ, 2006. – Т.2. – Ч.1. – с. 107-115.

7. Банкаускі веснік. Інфармацыйны выпуск. – Мінск: 2004. – № 18/275.

Приложение В

Перечень вопросов к зачету по дисциплине «Основы искусственного интеллекта»

1. Автоматы и их ключевая роль в создании систем «искусственного интеллекта»
2. Основные свойства алгоритмов
3. Численные и логические алгоритмы
4. Средства описания алгоритмов
5. Ассоциативные исчисления
6. Нормальный алгоритм Маркова
7. Машина Поста
8. Программирование на машине Поста
9. Конечные автоматы
10. Средства описания конечных автоматов
11. Роботы
12. Особенности вычислений на ЭВМ
13. Синтаксис и семантика формальных языков
14. Нотация Бэкуса
15. Использование формальных языков
16. Дескрипторные системы с грамматикой
17. Общие подходы к построению алгоритмических языков и трансляторов
18. Операционные системы
19. Базы данных и знаний
20. Системы управления базами данных
21. Проблемы создания искусственного интеллекта
22. Диалоговые методы решения задач
23. Экспертные системы
24. Нейронные сети и их применение
25. Нейрокомпьютеры
26. Интеллектуальные информационные технологии
27. Роботы в управлении движением автотранспорта
28. Защита информации и «безбумажные» системы документирования
29. Роль стандартов в защите информации
30. Прогнозирование с помощью нейросетей
31. Особенности применения стеганографии
32. Двухключевые и одноключевые системы шифрования
33. Особенности решения задач методом ветвей и границ

Задачи для практических занятий

Задание №1. Сопоставление особенностей численных и логических алгоритмов на примере задачи отыскания пути в лабиринте и вычисления $y=\sqrt{x}$ по алгоритму $y_{i+1}=0.5(y_i+x/y_i)$ с остановкой процесса по правилу $|y_{i+1} - y_i| \le \varepsilon$

Условия задач определяются индивидуально для каждого студента: в соответствии с рисунком 1.1.2 преподаватель задает пару вершин на графе а) и пару вершин, одна из которых находится на графе а), а вторая на графе б); в качестве x берется сумма букв в фамилии, имени и отчестве студента; y_0 – количество букв в имени студента, $\varepsilon=0.1$.

Задание №2. Выбрать машинно-ориентированный численный метод для вычисления полинома $y=x^5+2x^4+Фx^3+Иx^2+Ox+2$ с минимальной связностью, где Ф,И,О – соответственно число букв в фамилии, имени и отчестве студента.

Задание №3. Написать для любого метода программу сложения двух чисел на машине Поста и проверить её работоспособность для чисел Ф и И, где Ф – количество букв в фамилии, а И – количество букв в имени студента.

Задание №4. Для заданного преподавателем контура (аналогичного рис. 1.6.4) составить программу движения и действий робота.

Задание №5. С помощью нотации Бэкуса построить формальный язык по заданию преподавателя (например, для представления нечетных натуральных чисел; классификация заданных типов объектов и т.д.).

Задание №6-7. Решить задачи коммивояжера различными методами.

Часть первая: для исходных данных в нижеследующей таблице (по строке номера i пунктов начало движения, №1-предприятие, с которого делается завоз продуктов в магазины 2,3,4,5, а по столбцу – конечные пункты j, Ф – количество букв в фамилии студента, а И – в его имени).

	1	2	3	4	5
1	0	1	2	6	И
2	1	0	3	5	3
3	3	4	0	Ф	4
4	4	4	5	0	4
5	3	4	5	3	0

Задача решается 3 путями:

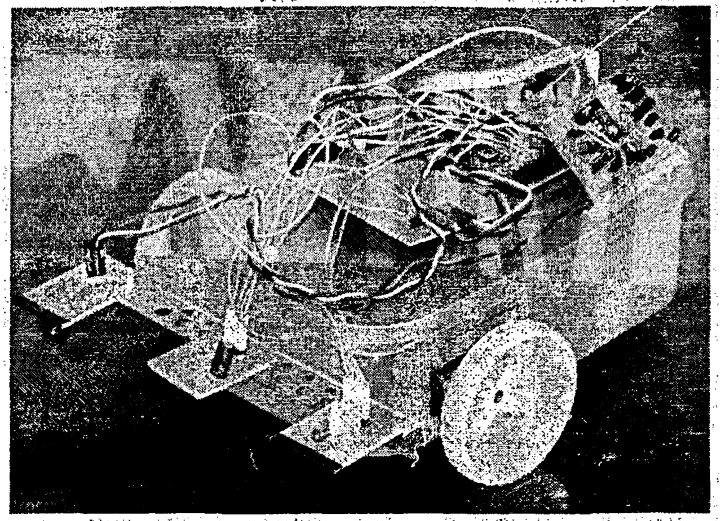
- интуитивным подходом на усмотрение студента;
- инженерным подходом по правилу FIFO, (первый пришел – первый обслуживается);
- строгим методом ветвей и границ.

В заключение все результаты сопоставляются.

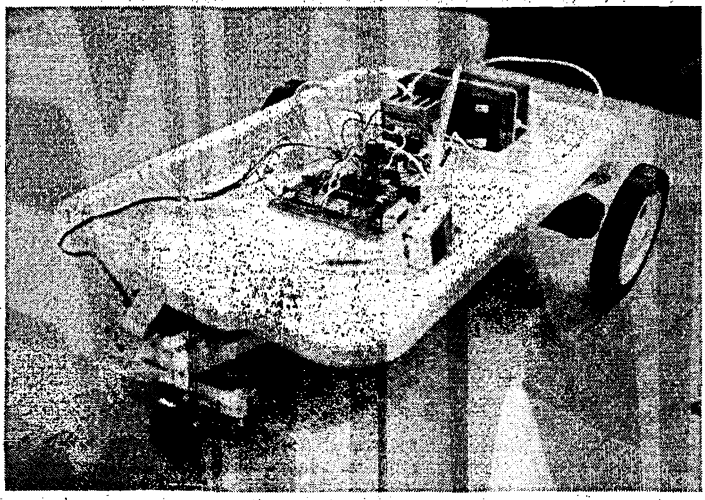
Часть вторая: условия задачи усложняются преподавателем некоторыми ограничениями производственного характера (наличием автомобилей, ограничением на сроки доставки, ограничением по величине заказа). Цель решения второй части приобрести навыки приспособления метода решения задачи к производственной ситуации.

Задание №8. Построить одноключевую систему кодирования информации на основе метода Диффи и Хэлмана. (Работа выполняется парой студентов, осуществляющих между собой связь, а при нечетном количестве студентов – парами и одной тройкой). Для каждой пары (тройки) параметры (a и p) выбираются свои, кроме того каждый из студентов выбирает свое секретное число, на основании которого вычисляется число для отправки в банк данных администратора системы. Для упрощения вычислительной работы могут быть рекомендованы следующие пары чисел (2 и 13), (2 и 11), (3 и 7), (5 и 7).

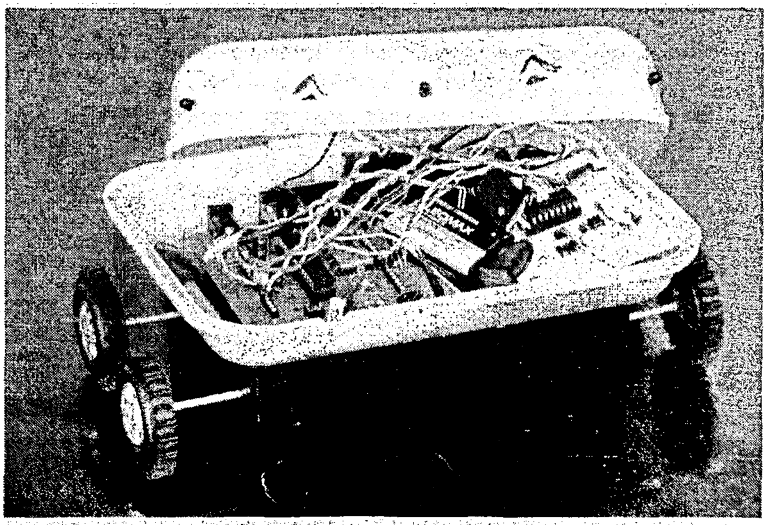
Мобильные роботы, разработанные кафедрой интеллектуальных информационных технологий БрГТУ



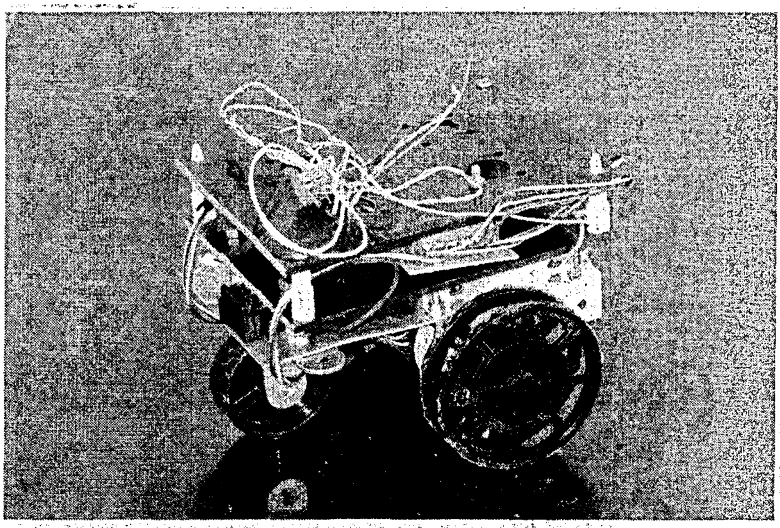
1. SDP-1 – первый робот, созданный в лаборатории кафедры ИИТ. Оснащен инфракрасными датчиками препятствий и радиосвязью



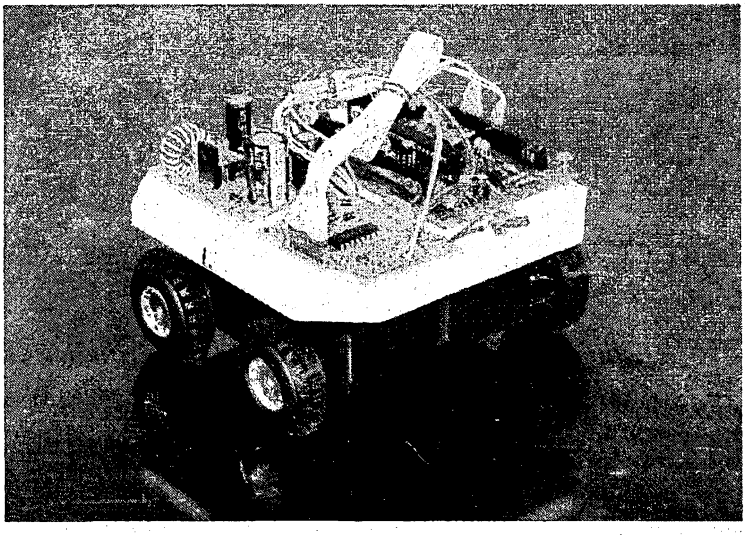
2. SDP-2 – модель мобильной, автономной платформы для проведения экспериментов со стереозрением



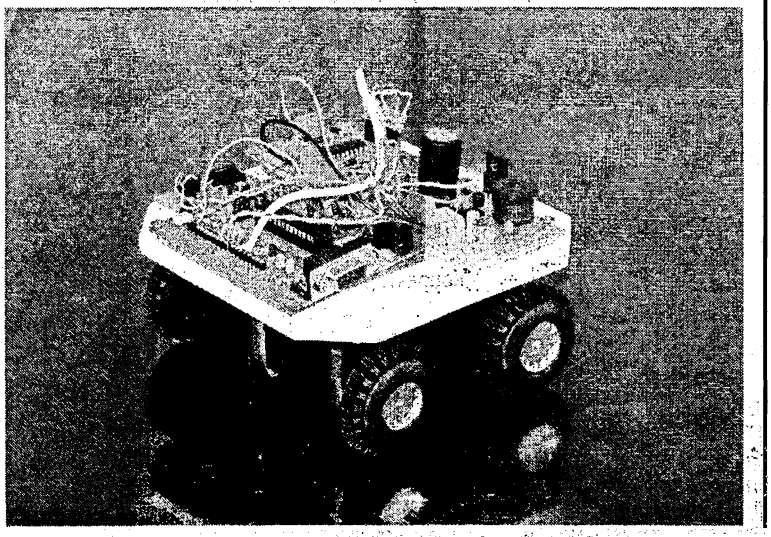
3. Hulk – робот, созданный студентом Деминым В. для 1 Открытого семинара по робототехнике



4. Proto – первая модель робота-футболиста



5. Mid – усовершенствованная модель робота-футболиста с инфракрасной связью



6. Def – продолжение линейки роботов-футболистов

УЧЕБНОЕ ИЗДАНИЕ

Матюшков Леонид Петрович
Головко Владимир Адамович
Шуть Василий Николаевич

ОСНОВЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебно-методический комплекс по дисциплине «Основы искусственного интеллекта»

Рекомендовано редакционно-издательским Советом учреждения образования «Брестский государственный технический университет» для студентов специальности 1-40 03 01 «Искусственный интеллект»

Ответственный за выпуск Матюшков Л.П.
Редактор Строкач Т.В.
Компьютерная вёрстка Кармаш Е.Л.
Корректор Никитчик Е.В.

ISBN 978-985-493-155-5

Издательство БрГТУ.
Лицензия № 02330/0549435 от 08.04.2009 г.
Подписано к печати 13.05.2010 г. Формат 60×84 1/16.
Бумага «Снегурочка». Гарнитура «Arial Narrow».
Усл. п. л. 6,7. Уч.-изд. л. 7,25.
Тираж 100 экз. Заказ № 535.
Отпечатано на ризографе учреждения образования «Брестский государственный технический университет»
224017, Брест, ул. Московская, 267.