

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

**Учреждение образования
«Брестский государственный технический университет»**

Кафедра интеллектуальных информационных технологий

**Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»**

Кафедра электронных вычислительных систем

**Институт повышения квалификации и переподготовки руководящих работников и специалистов по информационным технологиям и радиоэлектронике,
кафедра информационных систем и технологий**

В.А. Головки, А.А. Дудкин, Л.П. Матюшков

ОСНОВЫ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия по учебным дисциплинам «Основы искусственного интеллекта», «Традиционные и интеллектуальные информационные технологии», «Автоматизация проектирования вычислительных машин и систем» для специальностей 1-40 03 01 «Искусственный интеллект», 1-40 02 01 «Вычислительные машины, системы и сети», 1-53 01 02 «Автоматизированные системы обработки информации»

2-е издание, исправленное и дополненное

УДК 004.8 (07)
ББК 32 813
М 33

Рецензенты:

кафедра электронно-вычислительных систем учреждения образования
«Белорусский государственный университет информатики и радиоэлектроники»,
заведующий кафедрой д.т.н., доцент М.М. Татур

заведующий лабораторией логического проектирования
государственного научного учреждения
«Объединенный институт проблем информатики
Национальной академии наук Беларуси» д.т.н., проф. П.Н. Бибило

Головки В.А., Дудкин А.А., Мятюшков Л.П.
М 33 Основы компьютерных технологий: учебно-методическое пособие. – 2-е изд., испр. и
доп. – Брест: Издательство УО «БрГТУ», 2015. – 196 с.

ISBN 978-985-493-347-4

В данном методическом пособии рассмотрены понятия теории алгоритмов, формальных языков, грамматик и автоматов, формальные модели алгоритмов, а также 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

Издание адресовано студентам, магистрантам и специалистам, интересующимся проблемами современных компьютерных технологий в построении вычислительных машин, систем и сетей, методов искусственного интеллекта и их использования в различных приложениях.

Во втором издании устранены замеченные неточности, дополнены разделы п. 3.5. Нейронные сети как инструмент решения сложных задач и п. 5.4. Инструментальные средства проектирования, глава 8 дополнена п. 8.4. Задания для лабораторных работ по дисциплине «Автоматизация проектирования вычислительных машин и систем».

УДК 004.8 (07)
ББК 32 813

ISBN 978-985-493-347-4

Сборник авторов, 2015
© БрГТУ, 2015
Издательство БрГТУ, 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
глава 1. АЛГОРИТМЫ И АВТОМАТЫ	6
1.1. Общие понятия	6
1.2. Описание алгоритмов	12
1.3. Методы реализации алгоритмов	24
1.4. Ассоциативные исчисления и нормальный алгоритм Маркова	31
1.5. Абстрактные модели автоматов	35
1.6. Машина Поста и программирование на ней	44
1.7. Архитектура вычислительных систем	51
Контрольные вопросы	54
глава 2. ФОРМАЛЬНЫЕ ЯЗЫКИ И ОПИСАНИЕ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ	55
2.1. Синтаксис и семантика формальных языков	55
2.2. Алгоритмические языки и операционные системы	59
2.3. Механизмы поиска информации	64
2.4. Базы данных и знаний. СУБД и языки запросов	68
Контрольные вопросы	73
глава 3. ПРОБЛЕМЫ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ	74
3.1. Общие подходы к созданию систем искусственного интеллекта	74
3.2. Попытки создания общих решателей задач	77
3.3. Диалоговые методы решения задач и экспертные системы	78
3.4. Распознавание образов и обработка изображений	84
3.5. Нейронные сети как инструмент решения сложных задач	88
Контрольные вопросы	105
глава 4. ОБЩИЕ ПОДХОДЫ К МОДЕЛИРОВАНИЮ ПРОЦЕССОВ ОБЪЕКТОВ	106
4.1. Математическое и компьютерное моделирование	106
4.2. Модели оптимизации многомерных функций и принятие решений	107
4.3. Методы защиты информации и электронная подпись документов	108
4.4. Защита от несанкционированного доступа	113
4.5. Безбумажные системы ведения документации	113
Контрольные вопросы	114
глава 5. АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ	115
5.1. Основные понятия автоматизированного проектирования	115
5.2. Языки описания вычислительных систем	118
5.3. Технологии производства вычислительных систем	128
5.4. Инструментальные средства проектирования	133
Контрольные вопросы	136
глава 6. РЕАЛИЗАЦИЯ СЕТЕВЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ	137
6.1. Основные понятия об интеллектуальных информационных технологиях	137
6.2. Нейронные системы и нейрокомпьютеры в сетях	141
6.3. Системы технического зрения	152
6.4. Мультиагентные системы	156
6.5. Транзакции в сетях	160
Контрольные вопросы	160

Глава 7. ПЕРСПЕКТИВЫ РАЗВИТИЯ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ	161
7.1. Влияние нанотехнологий и мобильной связи на развитие информационных технологий.....	161
7.2. Интеллектуализация принятия решений в информационных технологиях.....	163
7.3. Рост сферы услуг в современном информационном обществе и его влияние на экономические и социальные процессы.....	164
Контрольные вопросы.....	167
Глава 8. ПРАКТИЧЕСКИЕ И ЛАБОРАТОРНЫЕ ЗАДАНИЯ	168
8.1. Общие указания к практическим заданиям.....	168
8.2. Задания для практических занятий.....	169
8.3. Задания для лабораторных работ в системе Mathcad.....	171
8.4. Задания для лабораторных работ по дисциплине «Автоматизация проектирования вычислительных машин и систем».....	186
8.5. Пример заданий для письменной работы.....	189
ЗАКЛЮЧЕНИЕ	190
ЛИТЕРАТУРА	191
ГЛОССАРИЙ	193

ВВЕДЕНИЕ

Современные компьютерные технологии проникли во все области науки и техники. Наблюдается рост «интеллектуализации» различных автоматизированных систем в технике, образовании и быту. Поэтому крайне необходимо для специалистов всех профилей владеть этими подходами в использовании современных компьютерных технологий во всех областях деятельности.

Знание общих закономерностей в создании, адаптации и эксплуатации различных компьютерных систем крайне необходимо всем специалистам, ориентирующимся на использование информатики в своей практической деятельности. Владение общими подходами и закономерностями развития вычислительных и информационных систем позволит повысить эффективность их применения также за счет квалифицированного выбора готовых программных продуктов или при заказе индивидуальных проектов.

Поэтому в пособии на доступном уровне для специалистов разных профилей излагаются сведения об основных элементах и подходах, применяемых в проектировании, конструировании и эксплуатации современных интеллектуальных систем (примером является проектирование информационных технологий для создания безлюдных производств, сложных конструкций, роботов с имитацией ряда функций человека, бытовой техники с программным управлением, перенастраиваемых систем на выпуск изделий близкого класса в зависимости от потребностей рынка). Особую роль, конечно, играют и процессы моделирования различных объектов с использованием сетей ЭВМ, безбумажного ведения документации с применением электронной цифровой подписи, включая и документирование коммерческих операций.

Любые системы машин быстро совершенствуются, им на смену всегда приходят более сложные и эффективные, но законы и принципы, лежащие в основе их создания и функционирования, более долговечны и составляют фундамент, воздвигаемый поколениями людей. Вещь данного пособия также состоит в том, чтобы в краткой и доступной форме ознакомить с основами этих знаний. Их составляют теория алгоритмов и автоматов, формальные логики, базы данных и знаний, операционные системы и системы управления базами данных, безбумажное ведение различных систем документирования, нейронные сети и нейроимпьютеры; сети ЭВМ и современные интеллектуальные технологии. Цель книги – ознакомить читателя с основными идеями и положениями этих теорий.

Одновременно делается попытка создать учебно-методическое пособие для нескольких смежных специальностей за счёт некоторой избыточности теоретических и практических вопросов для каждой из них при сохранении общей методической направленности на повышение роли индивидуальной работы студентов в освоении знаний и их закреплении. В какой-то мере для этой цели некоторые положения дублируются в заданиях для практических занятий лабораторных работ. Кроме того, такое построение пособия облегчает подбор конкретного материала по изучаемой дисциплине, руководствуясь общими указаниями к выполнению практических заданий и перспективами развития информационных технологий.

Материал книги подготовлен группой авторов: Головкин В.А. написал параграфы 3.4, 4.1, 4.2 (совместно с А.А. Дудкиным), параграф 3.1 (совместно с Матюшковым Л.П.), заключение; Дудкин А.А. – главу 5, параграфы 1.2, 1.7, 5.3; 1.3, 8.4 и 1.5 (совместно с Л.П. Матюшковым); Матюшков Л.П. – введение, главы 7 и 8 (совместно с Головкиным В.А.), параграфы 1.4, 1.5, 2.1-2.5, 3.2, 4.5, параграфы 4.4 и 2.3 (совместно с А.А. Дудкиным).

Рекомендуется для студентов специальностей: 1-40 03 01 «Искусственный интеллект» и 0 02 01 «Вычислительные машины, системы и сети» и 1-53 01 02 «Автоматизированные системы обработки информации»

Глава 1. АЛГОРИТМЫ И АВТОМАТЫ

1.1. Общие понятия

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.

Необходимость формального описания компьютера и его отдельных частей в процессе его проектирования требует применения специального математического аппарата, который необходим при любых разработках различных методов обработки информации, при синтезе и анализе любых информационных процессов. Для этого вводится понятие абстрактного (т. е. существующего не реально, а лишь в воображении) шифрового автомата.

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 последо-

ельности 01 01 01 ... Если рассматривать только начало последовательности, то никаких представлений о ней в целом сделать нельзя. Однако такое представление создается, если известно, 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 обо 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.

Интуитивное определение алгоритма. Существует несколько определений термина «алгоритм». Вообще под алгоритмом часто понимается некоторое формальное предписание, действуя согласно которому, можно получить нужное решение задачи. Эта формулировка отвечает интуитивной точке зрения на алгоритм, сложившейся еще в древности.

Классическим примером 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 искомым результат и процесс числений останавливается, иначе 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. КУ2 (процесс повторяется по вышеназванной схеме).

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

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.

По определению акад. А.Н. Колмогорова, алгоритм или алгоритм — это всякая система числений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи. В инженерной практике часто используется следующее определение: алгоритм — конечная совокупность точно сформулированных правил решения какой-то задачи.

Алгоритмы, в соответствии с которыми решение поставленных задач сводится к арифметическим действиям, называются численными. Численными являются также любые алгоритмы 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 способа-метода 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 Y достижима с площадки X , если существует путь, ведущий от X к Y через промежуточные коридоры и площадки. Очевидно, что если площадка Y вообще достижима с площадки X , то она достижима посредством простого пути, когда каждая из площадок проходится один раз, например (рис. 1.1.1), простой путь с площадки A в D : ABD или $ABCD$. Площадка K с площадки A вообще недостижима.

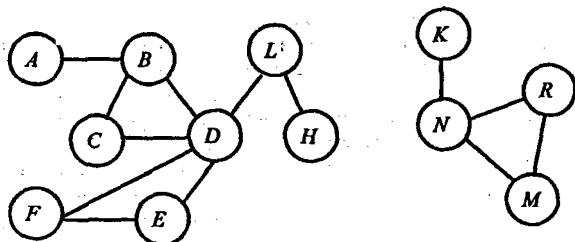


Рисунок 1.1.1 – Модель лабиринта

Решим задачу: начав поиск с площадки A , выяснить, достижима ли произвольная площадка Y ; если она достижима, то найти путь из A в Y , иначе после поиска вернуться в A .

Устройство лабиринта заранее неизвестно. Под решением задачи понимается указание общего метода поиска для любого расположения площадок A и Y . Предположим, что в начале и конце коридоров можно делать пометки, проходил ли данный коридор и сколько раз. Коридоры, которые не проходились ни разу, будут без пометок, пройденным один раз будем присваивать метку 1, а пройденным дважды – метку 2.

Находясь на какой-либо площадке, можно попасть на смежную двумя путями: прохождением по неотмеченному коридору, пометив его начало и конец знаком 1; возвратом с данной площадки по коридору с меткой 1, заменив метку коридора знаком 2. Находясь на какой-либо площадке, можно различать следующие признаки:

- 1) это площадка Y (цель достигнута);
- 2) петля (от данной площадки расходятся, по крайней мере, два коридора с меткой 1 и неотмеченных коридоров нет);
- 3) от данной площадки отходит, по крайней мере, один коридор без меток;
- 4) это исходная площадка A ;
- 5) отсутствие всех предыдущих признаков.

В соответствии с этими признаками ходы делаются до тех пор, пока не наступит остановка (табл. 1.1.1).

Таблица 1.1.1 – Метод поиска пути из *A* в *Y*

Признак	Ход
Площадка <i>Y</i>	Остановка
Петля	Возврат по последнему пройденному коридору
Имеется коридор без пометок	Движение по одному из коридоров без пометок
Площадка <i>A</i>	Остановка
Отсутствие 1-4	Возврат по пройденному коридору

Относительно рассматриваемого метода справедливы следующие утверждения:

1. При любом расположении *A* и *Y* за конечное число ходов наступит остановка либо в *Y*.

2. Если остановка на площадке *A*, то площадка *Y* недостижима.

Рассмотрим два небольших примера (рис. 1.1.2).

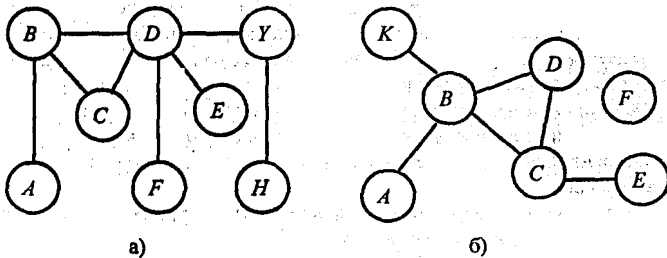


Рисунок 1.1.2 – Лабиринты: а) с достижимыми площадками; б) с недостижимыми площадками

Шаги процесса поиска достижимой площадки *F* в лабиринте, показанном на рис. 1.1.2,а, представлены в табл. 1.1.2, а недостижимость площадки *F* для лабиринта, изображенного на рис. 1.1.2,б, □ в табл. 1.1.3.

Таблица 1.1.2 – Поиск достижимой площадки *F*

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

Таблица 1.1.3 – Недостижимость площадки *F*

Номер хода	Признак	Направление движения	Метка 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. Например, 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 детерминирован-

ную 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 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 из предыдущей не дает результата, должно быть указано, что считать результатом.

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

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

Относительно перечисленных свойств алгоритма укажем их интерпретацию для 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 результатом выполнения каждого шага.

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

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

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

Математическое определение алгоритма. Алгоритмами в современной математике принято называть конструктивно задаваемые соответствия между изображениями объектов в абстрактных алфавитах.

Абстрактным алфавитом называется любая конечная совокупность объектов, называемых буквами или символами данного алфавита. При этом природа этих объектов нас совершенно не интересует. Символом абстрактных алфавитов можно считать буквы алфавита какого-либо языка, цифры, любые значки и даже слова некоторого конкретного языка. Основным требованием к алфавиту является его конечность. Алфавит, как любое множество, задается перечислением его элементов.

Итак, объекты реального мира можно изображать словами в различных алфавитах. Это позволяет считать, что объектами работы алгоритмов могут быть только слова. Тогда можно сформулировать следующее определение:

Алгоритм есть четкая конечная система правил для преобразования слов из некоторого алфавита в слова из этого же алфавита.

Слово, к которому применяется алгоритм, называется входным. Слово, вырабатываемое в результате применения алгоритма, называется выходным. Совокупность слов, к которым применим данный алгоритм, называется областью применимости этого алгоритма.

Можно выделить три основных типа универсальных алгоритмических моделей, различающихся исходными эвристическими соображениями относительно того, что такое алгоритм. Первый тип основан на функциональном подходе и рассматривает понятие алгоритма с точки зрения того, что можно вычислить с его помощью. Наиболее развитая и изученная модель этого типа – рекурсивная функция – является исторически первой формализацией понятия алгоритма.

Второй тип основан на представлении алгоритма как некоторого детерминированного процесса, способного выполнять в каждый отдельный момент некоторые примитивные операции, или инструкции. Основной теоретической моделью этого типа, созданной в 30-х годах, является машина Тьюринга, которая представляет собой автоматную модель, в основе которой лежит анализ процесса выполнения алгоритма как совокупности набора инструкций.

Третий тип алгоритмических моделей – это преобразования слов в произвольных алфавитах, в которых элементарными операциями являются подстановки, т. е. замены части слова (подслова) другим словом. Примерами моделей этого типа являются нормальные алгоритмы Маркова и канонические системы Поста.

Параллельный алгоритм. Параллельный алгоритм – алгоритм, операции которого могут выполняться одновременно. Приведенные выше определения алгоритма рассматривают его как процесс последовательной обработки структур данных, протекающий в дискретное время так, что в каждый следующий момент времени структура данных получается по известным правилам по структуре данных величин, имевшихся в предыдущий момент. Структура элементарных шагов в определении алгоритма не оговаривается, поэтому, при преобразовании величин могут допускать или предписывать параллельную их обработку. Алгоритм может быть параллельным, когда некоторые шаги обработки выполняются параллельно. Так, алгоритм сложения векторов может быть представлен как последовательный, в виде: $C_i = A_i + B_i$ для $i = 1, 2, \dots, n$, или в матричной записи: $C = A + B$, семантика которого допускает параллельную обработку элементов векторов.

Если при решении некоторой задачи процессоры выполняют одинаковую последовательность вычислений, но используют разные данные, то говорят о параллелизме по данным. Например, при поиске по базе данных каждый процессор может работать со своей частью базы данных. Если процессоры выполняют разные задания одной задачи, выполняя разные операции, то говорят о функциональном параллелизме.

Основная цель параллельных вычислений – уменьшение времени решения задачи. Чтобы ускорить решение задачи, не достаточно иметь параллельную вычислительную систему. Кроме этого, нужно еще создать для такой системы специальную (параллельную) программу. Для того чтобы алгоритм мог быть эффективно реализован в виде параллельной программы, он должен обладать *внутренним параллелизмом* (алгоритм можно разбить на параллельно, но не обязательно независимо выполняемые части). Распараллеливание, осуществляемое до начала выполнения алгоритма, называют статическим, а осуществляемое во время выполнения алгоритма – динамическим.

С параллельностью связано понятие масштабируемости. Масштабируемая параллельная система – это такая параллельная система, производительность которой пропорциональна числу содержащихся в ней процессоров. Масштабируемость зависит от возможностей мультимножественных сетей. Масштабируемость зависит также от параллельного алгоритма: алгоритм, проверенный и хорошо работающий на вычислительной системе (ВС) с малым числом процессоров может плохо работать (не давать ожидаемого ускорения) на системе с большим числом процессоров.

Рассмотрим, например, алгоритм вычисления выражения $a \cdot b + \sqrt{c \cdot d}$ (рис.1.2.3). В начале нужно вычислить произведения $a \cdot b$ и $c \cdot d$, затем извлечь корень, и, наконец, вычислить сложение. Описанный алгоритм можно изобразить следующим образом:

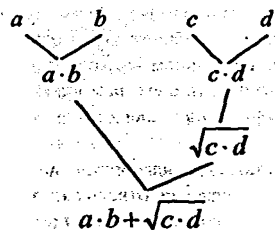


Рисунок 1.1.3 – Схема параллельного алгоритма

Мы видим, что процесс обработки данных может быть выражен в виде однонаправленного графа. Такой граф можно изобразить на плоскости, причём каждую арифметическую операцию располагать максимально высоко (если ось времени направлена вниз), но не выше тех операций, результат которых нужен для её вычисления. В таком случае высота графа будет равна минимальному времени (числу шагов) решения этой задачи на идеальной параллельной ВС с неограниченным числом вычислителей.

Степень параллелизма (параллельная сложность) алгоритма называется число его операций, которые можно выполнить параллельно (ширина графа, построенного описанным выше способом). Степень параллелизма алгоритма сложения векторов равна n . Операции, которые всегда могут быть выполнены на идеальной ЭВМ параллельно независимо от их количества, иногда называют операциями фиксированной глубины.

1.2. Описание алгоритмов

Всякий алгоритм, как уже говорилось ранее, отражает последовательность преобразований информации. Алгоритм обычно состоит из операционной и информационной частей.

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

При построении арифметических операторов предпочитают включать в каждый из них определенное законченное действие, например вычисление определителя, извлечение корня и т. п. На этом этапе последовательность размещения операторов не рассматривается.

Для осуществления вычислений на ЭВМ существенна очередность арифметических операторов, так как один из операторов может обеспечить возможность выполнения следующего. При построении описания вычислительной схемы алгоритма необходимо разместить арифметические операторы в такой последовательности, чтобы выполнение вычисления одного из них обеспечивало возможность реализации следующих за ним операторов. Естественно, что в частных случаях перестановка некоторых операторов местами допускается.

Процесс разработки алгоритма. 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 ментов, 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 программа алгоритма.

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

Способы описания алгоритмов. В процессе разработки алгоритмов для их анализа, проверки числа элементов, удобства программирования и других целей используется множество способов описания алгоритмов. Среди них наибольшее распространение получили:

- словесный (на естественном языке);
- схемы алгоритмов (графический);
- операторные схемы;
- псевдокоды;
- языки программирования;
- 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 оптимизации алгоритма, четвертые — на этапе общения человека с ЭВМ и т. д.

Для иллюстрации средств записи алгоритмов приведем простейший пример. Пусть задана последовательность $X_1, X_2, \dots, X_i, \dots, X_n$ из n произвольных действительных чисел и требуется подсчитать в этой последовательности количество чисел M для случая $X_i \geq a$ и количество чисел B для случая $X_i < a$.

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

1. $B := 0, M := 0$.
2. $i := 1$.
3. Если $X_i < a$, то ПКУ4, иначе ПКУ6.
4. $B := B + 1$.
5. ПКУ7.
6. $M := M + 1$.
7. Если $i < n$, то ПКУ8, иначе ПКУ10.
8. $i := i + 1$.
9. ПКУ3.
10. Конец.

Знак « $:=$ » понимается как символ операции присваивания величине, стоящей слева от знака, значения величины, стоящей справа.

Эта последовательность операций и представляет собой запись алгоритма для человека, владеющего алгеброй и правилами выполнения арифметических действий. Однако в ней присутствуют «необычные» операторы 4, 6 и 8, которые бессмысленны для немашинной математики, поскольку в ней такие равенства, как $B = B + 1$ и другие, аналогичные ему, невозможны. В такой записи в том, что старое значение B увеличивается на единицу и новое значение

присваивается снова величине с именем *B*. Это равносильно операции, когда результат записывается в заданную клетку таблицы, а вычисления проводятся на черновике. После вычисления нового результата на черновике старый результат в этой клетке таблицы стирается и на его месте записывается новый и т. д. Такая запись порождена удобством для отображения процессов вычислений на машине, где роль таблицы выполняет память машины. В этом случае после вычислений машина направляет новый результат на заранее выбранный адрес памяти, что приводит к автоматическому 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 подчеркнем принадлежность оператора к одной из этих групп будем записывать A_1, A_2, A_4, A_6, A_8 и соответственно P_3, P_5, P_7, P_9 .

Мы остановимся более детально на операторной форме представления алгоритмов их схем, так как предварительная подготовка задачи для программирования крайне 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 алгоритмизации и программирования резко возрастают. Естественно, что возник вопрос о расчленении вычислительного процесса на такие простые части, чтобы каждую из них программировать отдельно, а затем, соединив частные программы, получить всю программу 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 по одной строке 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 при его буквенном обозначении.
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 запятой, то от оператора, записанного слева, нет передачи управления оператору, записанному справа.
5. Передача управления оператору, записанному не рядом справа, обозначается стрелкой.

Например, операторная схема может выглядеть так:

$$A_1 A_2 P_3 A_4; A_5 A_6 A_7$$

Для удобства записи горизонтальную часть стрелки можно опускать. При этом начало стрелки отмечается малой вертикальной стрелкой, направленной от оператора и снабженной номером оператора, которому передается управление с указанием номера, передающего управление оператору.

Тогда эту же схему можно представить так:

$$A_1 A_2 P_3 A_4; A_5 A_6 A_7$$

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

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

$$A_1 A_2 P_3 A_4 P_5 A_6 P_7 A_8 P_9 A_{10}$$

Во многих случаях бывает удобно группу элементарных операторов обозначать одной зной. В этом случае придерживаются правила, что управление извне (от операторов, не принадлежащих данной группе) может получать лишь один элементарный оператор группы. ю группу элементарных операторов называют обобщенным оператором.

С помощью операторной схемы алгоритма составляется схема решения задачи, которая зняется операторами, свойственными машинному вычислительному процессу. Для каждого затора в порядке его записи затем составляется своя частная программа. Совокупность этих ных программ и дает программу решения задачи. Использование операторной схемы алго- ма при программировании рассчитано на получение 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 срыва или поручать продолжение начатой работы другому лицу.

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

К логическим задачам относятся преимущественно многочисленные задачи управления ланирования. При их решении на ЭВМ схемы алгоритмов приобретают важное значение, как они являются пока почти единственным формальным аппаратом, отображающим ди- нiku сложных процессов.

Схема алгоритма, с одной стороны, служит как формальный язык для записи содер- ния и логических связей задачи, а с другой — она сравнительно легко преобразуется в про- му для ЭВМ.

Схемы алгоритмов — функционально ориентированные графические изображения, с мостью которых, используя текст и специальные символы, описывают последовательность гов процесса во времени, связи рассматриваемой системы с внешней средой и ветвление оцесса. Краткое текстовое или формальное описание шагов приводится внутри символов. едения о входных и выходных данных некоторых шагов процесса могут даваться в зако- ованной форме. Схемы алгоритмов обладают тем преимуществом, что для их понимания требуется специальных знаний.

Для удобства чтения схем алгоритмов в ряде стран разработаны стандартные графиче- е обозначения для наиболее часто употребляемых и специфических операторов.

Поясним понятие схемы алгоритма. Пусть требуется вычислить функцию

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

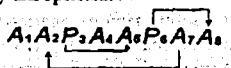
всех дискретных значений x , с шагом 0,1 из сегмента $[-7; 5]$.

Так как y представлен через аналитические выражения x^2 и $x^2 + \sqrt{x}$, по которым могут гь произведены вычисления в результате подстановки в них значений x , то алгоритм для ичисления очевиден, остается построить схему алгоритма.

Из условий задачи вытекает необходимость выполнения следующих операций.

1. Ввести исходное значение $x := X_0 := -7$. ПКУ2.
2. $M := x^2$. ПКУ3.
3. Проверить $x \geq 0$. 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 $A_1, A_2, P_3, A_4, A_5, P_6, A_7, A_8$ (или еще короче: 1, 2, ..., 8). Если расположить эти символы в строчку и с помощью стрелок указать связи, то получим операторную схему алгоритма:



Если теперь вместо символов операторов вычертить прямоугольники, в них написать содержание действий, выполняемых данными операторами, и показать связями последовательность этих действий, то получим схему алгоритма (рис. 1.2.4).

Граф-схема алгоритма (ГСА) – конечный связный ориентированный граф $G = (V, E)$, где V – множество вершин графа $v_i \in V, 1 \leq i \leq N$, которые соответствуют операторам; E – множество дуг графа $\{e_k = e_{i,j} = (v_i, v_j)\} \in E, k = 1, \dots, M, i = 1, \dots, N, j = 1, \dots, N, i \neq j$, которые задают порядок следования операторов; $N = |V|$ – число вершин графа; $M = |E|$ – число дуг графа.

Таким образом, ГСА есть графическое изображение последовательности операций, согласно которым получают решение задачи. Каждый этап процесса обработки информации представляется в виде геометрических символов (блоков), имеющих определенную конфигурацию в зависимости от характера выполняемых операций. Перечень символов, их наименование, отображаемые ими функции, форма и размеры определяются ГОСТами.

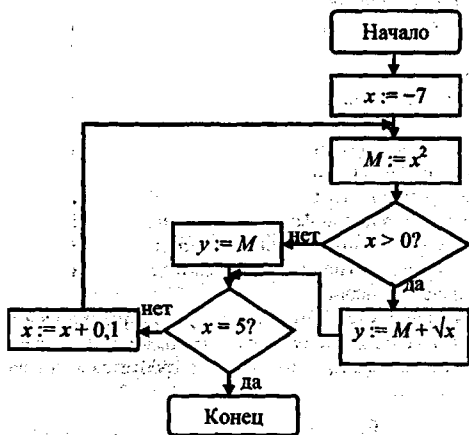


Рисунок 1.2.1 – Схема алгоритма

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 одна стрелка, и выходной, из которого не выходит ни одна стрелка.
3. Все узлы граф-схем отмечены своей логической переменной P_i или арифметическим оператором A_j .
4. Из каждого узла P_i отходят, как правило, две стрелки, а с узла A_j – только одна стрелка.

Если в ГСА логические переменные и арифметические операторы заменить соответствующими описаниями, то граф-схема превратится в схему алгоритма (рис. 1.1.5).

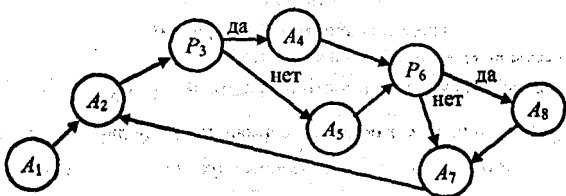


Рисунок 1.2.2 – Граф-схема алгоритма

При всем многообразии алгоритмов решения задач в них можно выделить три основных вида вычислительных процессов: линейный; ветвящийся; циклический. Линейным называется такой вычислительный процесс, при котором все этапы решения задачи выполняются в естественном порядке следования записи этих этапов. Ветвящимся называется такой вычислительный процесс, в котором выбор направления обработки информации зависит от одних или промежуточных данных (от результатов проверки выполнения какого-либо логического условия). Циклом называется многократно повторяемый участок вычислений.

Модель представления параллельного алгоритма. Параллельная обработка информации выполняется на кластере, в общем случае гетерогенном. Он представляется множеством $P = \{p_1, p_2, \dots, p_m\}$, где p_i – отдельный узел (компьютер) кластера, m – число узлов кластера. Каждый компьютер p_i имеет характеристику производительности s_i , которая определяет время выполнения одной условной единицы вычислений. Узлы кластера соединены коммуникационными каналами. Каждый канал между узлами p_i и p_j , обозначенный l_{ij} , имеет характеристику b_{ij} , которая определяет ширину канала и скорость передачи данных между p_i и p_j .

Поток, обрабатываемый параллельным алгоритмом, представляется множеством объектов $J = \{j_1, \dots, j_n\}$. Назовем алгоритмы обработки объекта определенного типа сценарием. Сценарии обработки в общем случае содержат множество операций, каждая из которых может избирательно применяться к объектам различных типов, либо применяться с различными параметрами зависимости от типа обрабатываемого объекта. Система обработки способна выполнять множество операций обработки $O = \{o_1, \dots, o_k\}$ $k > m$. Сценарий обработки отдельного типа кадра

содержит подмножество операций $O_j = \{o_{j_1}, \dots, o_{j_n}\}$, $\bigcup_{j=1}^n O_j = O$. Операции в каждом сценарии упорядочены отношениями следования $o_{j_a} \succ o_{j_b}$, т. е. для кадра с типом j

операция a выполняется перед операцией b . Каждая операция обработки o_i характеризуется количеством вычислений $w_{o_i}^j$, представленной количеством единиц вычисления данной операции для определенного типа кадров j . Две операции над различными кадрами, назначенные на один и тот же процессор, не могут выполняться одновременно. Два экземпляра одной и той же операции над различными кадрами также не могут выполняться в один и тот же промежуток времени.

Все сценарии обработки представляются в форме направленного ациклического графа в виде кортежей $G = (V, E, W, C)$, где

V – множество вершин графа $v_i \in V, 1 \leq i \leq N$. Каждая вершина ассоциируется с операцией обработки данных. Множество вершин представляет декомпозицию параллельной программы обработки потока изображений на отдельные операции (гранулы параллелизма);

E – множество дуг графа $(e_{i,j} = (v_i, v_j)) \in E, i = 1, \dots, N, j = 1, \dots, N, i \neq j$. Дуга представляет собой отношение предшествования между операциями алгоритма и определяет задачу результатов обработки от источника к приемнику;

W – матрица вычислительной стоимости операций $W = \bigcup W_{ij}^j$; C – множество стоимости дуг, элемент $c_{i,j} \in C$ определяет объем информации между двумя операциями обработки данных, передаваемой по дуге $e_{i,j} \in E$. Связанные дугой операции используют один фрагмент данных, поэтому для всех сценариев и типов кадров дуги имеют равную стоимость.

Построение приложения для параллельной обработки потока данных включает: создание сценариев, описывающих логическую структуру программы; назначение операций обработки вершинам графа и определение параметров вызова для каждого типа данных; отображение графа сценариев на топологию вычислительного кластера. Параллельная программа представляется преобразованием $((O), (P)) \rightarrow \bigcup_{p \in P} (O_p)$, где $\forall O_k, O_p: O_k \neq O_p, O_k \cap O_p = \emptyset$. Каждое множество операций O_p размещается на выбранном узле кластера.

Теперь цель обработки потока можно записать как минимизацию времени обработки объектов данных:

$$T_{opt} = \min\{\max T_N\}, \quad (1.2)$$

где T_N означает момент окончания обработки объекта N . Оптимизационный алгоритм использует для оценки расписания имитационную модель, эквивалентную реальному вычислительному процессу.

Вычислительные гранулы объединяются в специализированные библиотеки, которые динамически подключаются при выполнении параллельного приложения. Реализация различных классов алгоритмов обработки информации в виде библиотек вычислительных гранул позволяет расширить сферу применения системы организации параллельных вычислений и дает возможность создавать новые классы приложений.

Этапы параллельного проектирования. Получить параллельный алгоритм решения задачи можно путем распараллеливания имеющегося последовательного алгоритма или путем разработки нового параллельного алгоритма. Возможно, для осуществления распараллеливания алгоритм решения задачи придется заменить или модифицировать (например, устранить некоторые зависимости между операциями).

Разработка алгоритмов параллельных вычислений состоит в следующем:

- Выполнить анализ имеющихся вычислительных схем и осуществить их разделение (декомпозицию) на части (подзадачи), которые могут быть реализованы в значительной степени независимо друг от друга.
- Выделить для сформированного набора подзадач информационные взаимодействия, которые должны осуществляться в ходе решения исходной поставленной задачи.
- Определить доступную для решения задачи вычислительную систему и выполнить распределение имеющего набора подзадач между процессорами системы.
- Разделить вычисление на независимые части.
- Выделить информационное взаимодействие между частями.
- Масштабировать задачи.
- Распределить все задачи между процессорами.

При рассмотрении параллельных алгоритмов вводится понятие параллельной ГСА, состав которой входят вершины распараллеливания/синхронизации, функциональность которых обычно совмещается (фрагмент такой ГСА приведен на рис. 1.1.6). Иногда в состав ГСА вводятся вершины дополнительных типов с целью обеспечения возможности моделирования выполнения алгоритма сетью Петри. Однако не любой ориентированный граф, со-

енный из вершин указанных выше типов, может быть отождествлен с корректным алгоритмом. Например, из операторной вершины не может выходить более одной дуги. По- у на практике обычно ограничиваются рассмотрением подкласса граф-схем алгоритмов, летворяющих свойствам безопасности, живости и устойчивости.

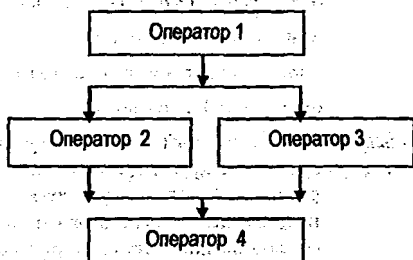


Рисунок 1.2.3 – Фрагмент параллельной ГСА

Сети Петри. Среди многих методов описания и анализа дискретных параллельных гем выделяют подход, который основан на использовании сетевых моделей, относящихся ятам специального вида, предложенных Карлом Петри для моделирования асинхронных юрмационных потоков в системах преобразования данных. Сети Петри обеспечивают сание как параллельных алгоритмов, так и собственно ВС и ее устройств.

Структура сети представляется ориентированным двудольным графом. Множество V шин графа разбивается на два подмножества T и P , $V = T \cup P$, $T \cap P = \emptyset$. Дугами мсвязываться вершины из множеств P и T . Динамика развития процессов отражается в шинах P метками (марками). Распределение меток по вершинам P называют маркироиением сети. Каждое маркирование соответствует определенному состоянию сети.

Сеть Петри определяется четверкой $N = \{P, T, F, M_0\}$, где

$P = \{p_i\}$, $i = 1, 2, \dots, n$ – непустое конечное множество позиций (мест);

$T = \{t_j\}$, $j = 1, 2, \dots, m$ – непустое конечное множество переходов;

$F: T \times P \cup P \times T \rightarrow \{0, 1\}$ – функция инцидентности, такая, что каждое место инцидент- какому-либо переходу ($T \times P \rightarrow \{0, 1\}$ - функция следования), каждый переход инцидентен ому либо месту ($P \times T \rightarrow \{0, 1\}$ - функция предшествования), т. е. она задает множества $\{t_j, p_i\}$ и $\{p_i, t_j\}$;

$M_0: P \rightarrow N_0$ – начальное маркирование (состояние) сети; N_0 – множество положи- нных целых чисел.

Дуги, предшествующие позиции p_i , обозначим множеством $J(p_i) = \{(t_j, p_i) | J(t_j, p_i) = 1\}$, ги, предшествующие переходу t_j , множеством $J(t_j) = \{(p_i, t_j) | O(p_i, t_j) = 1\}$.

Здесь запись $J(t_j, p_i) = 1$ означает наличие дуги (t_j, p_i) , а запись $O(p_i, t_j) = 1$ – дуги (p_i, t_j) . Аналогично, дуги, следующие из p_i и t_j , представим множествами $J(p_i) = \{(p_i, t_j) | O(p_i, t_j) = 1\}$, $O(t_j) = \{(t_j, p_i) | J(t_j, p_i) = 1\}$.

Входные позиции перехода t_j объединяются в множества его предшественников $J(t_j) = \{p_i \in P | O(p_i, t_j) = 1\}$, а выходные позиции – в множества позиций-последователей $O(t_j) = \{p_i \in P | J(t_j, p_i) = 1\}$.

Маркирование сети представляется вектором $M = \{m(p_i)\}$, где $m(p_i)$ – число меток позиции p_i . Переход t_j возбужден при маркировании M и может сработать, если выполнено условие $\forall p_i \in \text{Pre}(t_j) [m(p_i) - O(p_i, t_j) \geq 0]$, т. е. число меток $m(p_i)$ больше или равно числу дуг (p_i, t_j) , что соответствует $m(p_i) - O(p_i, t_j) \geq 0$. Условием срабатывания является истинность функции инцидентности при заданных p_i и t_j .

Срабатывание перехода t_j приводит к тому, что каждая позиция $p_i \in \text{Pre}(t_j)$ теряет $O(p_i, t_j)$ меток, а каждая из позиций $\text{Post}(t_j)$ получает $J(t_j, p_i)$ меток.

Сеть Петри представляет собой двудольный ориентированный граф, в котором позиции соответствуют вершины, изображаемые кружками, а переходам – вершины, изображаемые утолщенными черточками; функциям J соответствуют дуги, направленные от позиций к переходам, а функциям O – дуги, направленные от переходов к позициям.

Как и в системах массового обслуживания, в сетях Петри вводятся объекты двух типов динамические, которые изображаются метками (маркерами) внутри позиций, и статические, которым соответствуют вершины сети Петри.

Распределение маркеров по позициям называют маркировкой. Маркеры могут перемещаться в сети. Каждое изменение маркировки называют событием, причем каждое событие связано с определенным переходом. Считается, что события происходят мгновенно и независимо при выполнении некоторых условий.

Каждому условию в сети Петри соответствует определенная позиция. Совершенно наоборот соответствует срабатывание (возбуждение или запуск) перехода, при котором маркер из входных позиций этого перехода перемещаются в выходные позиции. Последовательность событий образует моделируемый процесс.

В сетях Петри условия – это позиции, а события – переходы. В соответствии с этим граф сети Петри является двудольным ориентированным мультиграфом. Изображение позиции и перехода на графе показано на рис. 1.2.4.

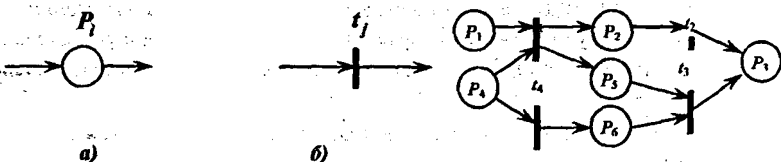


Рисунок 1.2.4 – Графическое изображение сети Петри:
а) позиция; б) перехода

Ориентированные дуги могут соединять только позиции и переходы в прямом и обратном направлении (свойство двудольности). Сеть Петри является мультиграфом, так как допускается кратность дуг между позициями и переходами (вершинами графа).

Правила срабатывания переходов конкретизируются следующим образом: переход срабатывает, если для каждой из его входных позиций выполняется условие $N_i > K_i$, где N_i – число маркеров в i -й входной позиции, K_i – число дуг, идущих от i -й позиции к переходу; при срабатывании перехода число маркеров в i -й входной позиции уменьшается на K_i , а в j -й в

ой позиции увеличивается на M_j , где M_j – число дуг, связывающих переход с j -й позицией. Пример смены разметки показан на рис. 1.2.5. Процесс срабатывания переходов и смены метки называют работой сети Петри.



Рисунок 1.2.5 – Смена разметки сети Петри: а) до срабатывания перехода t_i ; б) после срабатывания перехода t_i .

Произвольная ГСА является частным случаем сети Петри с одной меткой, которая соответствует текущему состоянию автомата, при выполнении последовательной части алгоритма и с несколькими метками, чье количество соответствует количеству параллельных частей алгоритма, при выполнении параллельной части алгоритма управления. Множество сходов t при этом соответствует множеству условных вершин, множество событий – множеству наборов управляющих сигналов, а множество мест S при этом соответствует множеству состояний автомата и множеству флагов операционного автомата.

Если при маркировании M возбуждено несколько переходов, то порядок их срабатывания не определен, и, следовательно, может быть представлено несколько последовательно срабатывающих переходов.

В качестве примера сети Петри рассмотрим ВС (рис. 1.2.6), содержащую процессор (П) в связях с объектом (УСО), которые могут работать параллельно, и запоминающее устройство (ЗУ).

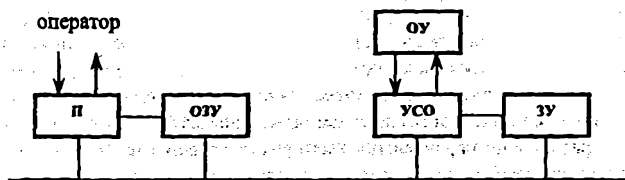


Рисунок 1.2.6 – Однопроцессорная ВС.

Процессы в данной ВС порождаются периодическими и инициативными запросами от оператора и инициативными запросами от оператора. Рассмотрим очередь запросов оператора, которая включает выполнение на процессоре П двух операций: операция 1 использует текущие данные из ЗУ; операция 2 выполняется после первой и завершения работы УСО с ОУ и ЗУ. Сеть Петри, описывающая данный процесс, приведена на рис. 1.2.7.

Переходы: t_1 – ввод данных с ЗУ в ОУ; t_2 – операция 1; t_3 – работа УСО с ОУ и ЗУ; t_4 – операция 2; t_5 – подготовка ЗУ к работе.

Позиции: P_1 – П свободен; P_2, P_4 – ввод данных с ЗУ в ОУ выполнен; P_3 – операция 1 выполнена; P_5 – работа УСО закончена; P_6 – УСО свободно; P_7 – ЗУ готово к работе; P_8 – операция 2 выполнена.

Переход t_1 срабатывает, если есть запрос, свободный процессор и готовое к работе УСО. Возможность параллельной работы процессора и УСО отражена параллельными ветвями t_2 и t_3 . Операция 2 (переход t_4) выполняется по завершению операции 1 и работы УСО. После срабатывания перехода t_4 освобождается процессор, ЗУ и фиксируется метка о завершении процесса. После подготовки ЗУ (переход t_5) может обрабатываться следующий запрос оператора.

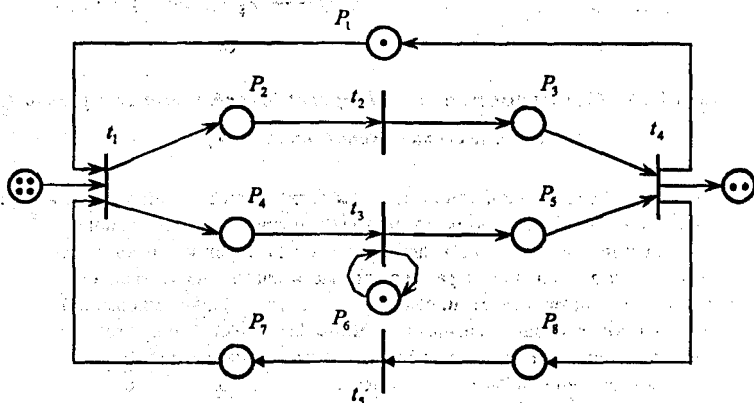


Рисунок 1.2.7 – Модель процесса в виде сети Петри

Представление алгоритма в виде диаграммы расписания. Для удобства представления параллельных алгоритмов наряду с другими способами используют также временные диаграммы выполнения операторов при заданных значениях времени начала (окончания) и выполнения. Операторы обозначаются прямоугольниками с длиной, равной времени выполнения соответствующего оператора. Для указания связей между прямоугольниками операторами используются стрелки, которые соответствуют дугам в графе алгоритма. Когда связей становится много, из-за множества стрелок проследить эти связи становится трудно.

Для того чтобы диаграмма оставалась наглядной при любом количестве связей и однозначно отражала расписание параллельного алгоритма введем следующие дополнительные правила. Ось ординат разобьем на интервалы, каждый из которых соответствует одному из параллельно работающих процессоров. В каждом интервале будем размещать только те прямоугольники-операторы, которые закреплены за соответствующим этому интервалу процессором. Операторы на диаграмме будем обозначать в центре прямоугольников цифровой кружочке, а связи между операторами – цифрами слева и справа от этого кружочка. В левой части прямоугольника-оператора будем записывать номера предшествующих по информационной связи операторов, а в правой части – номера операторов, следующих за данным оператором. Поскольку каждый прямоугольник диаграммы размещается в интервале «своего» процессора, описанный способ их нумерации отражает также связь между процессорами. Поэтому этот способ представления является исчерпывающим наглядным представлением расписания параллельного алгоритма и может быть удобным программисту для написания параллельной программы. При этом информация о номерах связанных операторов может использоваться для оценки объема передаваемых данных, как между процессами, так и между процессорами.

Графовые модели программного обеспечения. Одним из наиболее ответственных звеньев создания программного обеспечения (ПО) различного назначения является разработка спецификаций на будущую программу. Под спецификацией в общем виде понимается задание задачи, которую должна решать программа. При создании спецификаций возникает необходимость в подборе точных математических понятий, адекватных задаче и доступных потребителям создателям программ. В многообразии средств, применяемых в спецификациях, важное место занимают графовые средства — математические понятия, имеющие с относительно простыми видами связей между объектами и допускающие наглядное графическое изображение.

Наиболее распространенными подклассами графов, применяемых в спецификациях являются деревья, конечно-автоматные диаграммы, обобщенные диаграммы переходов, Петри, диаграммы «объектов связей», семантические сети и схемы алгоритмов.

Обширный перечень разновидностей графов обусловлен спецификой областей применения проектируемого ПО. Так, диаграммы «объектов связей» и деревья используются для активного представления предметных областей при проектировании ПО баз данных, семантические сети — для представления семантики естественных языков в системах искусственного интеллекта, конечно-автоматные диаграммы и сети Петри — для описания процессов функционирования дискретных систем, схемы алгоритмов с различными видами структурирования — для описания ПО методом последовательной детализации.

Интерпретация перечисленных видов графов, их получение и практическое использование рассматриваются в соответствующих разделах специальных дисциплин. Остановимся на особенностях построения графовой модели структуры программы, применяемой на этапах проектирования, анализа производительности и контроле структуры ПО. Для перечисленных видов в математической модели программы достаточно задать порядок следования действий, выделяемых операциями или командами. Тогда программа представляется ориентированным графом $G(X, U)$, множество X вершин которого соответствует линейным участкам программы, а дуги U указывают на связи между участками. При необходимости каждая вершина графа взвешивается числом p_i , указывающим на длину (число команд, операций) участка.

Анализ графовой модели на соответствие формальным правилам построения позволяет при выполнении программы выявить такие ошибки в ее структуре, как наличие вершин, недостижимых из начальной ни по одному из путей («лишние» вершины), а также вершин, из которых не достижима конечная вершина («ступиковые» вершины).

В корректно составленной программе через каждую вершину должен проходить хотя бы один путь, соответствующий варианту реализации. Эффективным методом выявления структурных ошибок по графу программы является использование матрицы достижимости $R = \|r_{ij}\|_{n \times n}$ графа, элементы которой формируются по правилу:

$$r_{ij} = \begin{cases} 1, & \text{если вершина } x_j \text{ достижима из } x_i \text{ хотя бы по одному пути;} \\ 0, & \text{если вершина } x_j \text{ не достижима из } x_i \text{ ни по одному из путей.} \end{cases}$$

Матрица достижимости R может быть получена из матрицы смежности A путем моделирования движения по графу в направлении дуг либо аналитически по формуле:

$$R = \sum_{i=1}^n A_i,$$

где A_i — i -кратное произведение матрицы смежности A самой на себя.

Ниже представлена матрица достижимости графовой модели некоторой программы. Вершины x_s, x_f соответствуют формальному началу и концу программы. При определении структурных ошибок по матрице R проверяются следующие соотношения:

$$\sum_{j=1}^n r_{sj} = n-1; \quad \sum_{j=1}^n r_{jf} = n-1. \quad (1.2.1)$$

Соотношения (1.2.1) указывают на отсутствие лишних и тупиковых компонент в графовой модели программы. При невыполнении соотношений локализация ошибочных компонент выполняется проверкой условий: $r_{ij} = 0$ и $r_{jf} = 0$.

Анализ матрицы достижимости графа программы (рис. 1.2.8) показывает на наличие "лишних" x_7, x_8, x_9 ($r_{27} = r_{28} = r_{29} = 0$) и "тупиковых" компонент x_2, x_3, x_4 ($r_{2f} = r_{3f} = r_{4f} = r_{5f} = 0$). Графовая модель программы может использоваться для определения ее статистических характеристик и показателей надежности.

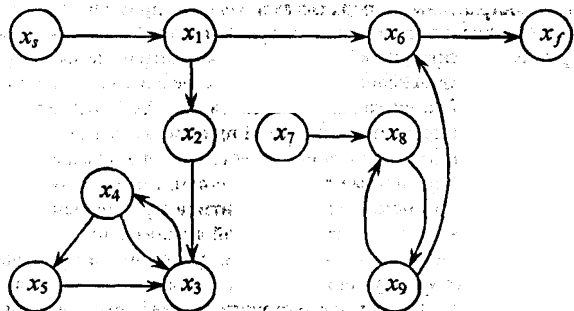


Рисунок 1.2.8 – Графовая модель программы

1.3. Методы реализации алгоритмов

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 дискретный характер, и имеются соответствующие модели вычислений, то в этом случае метод может практически сразу использоваться для машинной реализации, например, метод нахождения наибольшего общего делителя двух целых положительных чисел. Проблемы могут лишь возникнуть в связи с задачей представления в памяти очень больших чисел или же из-за ограничений на время ожидания ответа. Дополнительных затрат потребовала переориентация методов, использующих непрерывные математические модели вычислений. Непрерывные модели вычислений стали заменяться дискретными. Например, вычисление интеграла методом подстановок свелось к суммированию. Если раньше задача взятия одномерного интеграла на отрезке решалась для произвольных границ отрезка, то теперь она стала решаться для фиксированных границ отрезка. В первом случае точность результата определялась на базе погрешностей вычислений по итоговой формуле, а во втором случае она еще стала зависеть и от количества точек деления отрезка, определяющего число элементарных трапеций при суммировании. Таким образом, появилась специфическая задача автоматического выбора числа точек деления отрезка, чтобы избежать больших погрешностей метода.

Эти проблемы породили задачу анализа устойчивости вычислений по дискретным алгоритмам.

Во многих вычислительных алгоритмах, разработанных до появления ЭВМ, исследовались только вопросы получения погрешностей из-за аппроксимация непрерывных методов (делей) вычислений дискретными операциями. Погрешности представления данных и наложения систематических ошибок при большом числе простейших операций над приближенными числами не изучались.

Такого рода трудности стали обнаруживаться, когда теоретически сходящиеся процессы в некоторых случаях на ЭВМ не дали достоверных результатов. Поэтому, применяя ЭВМ для решения математических задач, необходимо учитывать ее особенности как вычислительного инструмента (в первую очередь это касается персональных ЭВМ и программируемых микрокалькуляторов с небольшой разрядностью для представления чисел):

- 1) количество цифр в изображении чисел, над которыми производятся действия, и полнота представления их;
- 2) большую скорость операций над числами, хранящимися в оперативной памяти;
- 3) сравнительно малую скорость ввода исходных данных и программ, а также вывода результатов;
- 4) сравнительно малую скорость обмена числами между оперативной и внешней памятью;
- 5) ограниченную емкость оперативной памяти при практически безграничной емкости оптического накопителя;
- 6) возможность случайных сбоев в работе машин.

Выбирая численный метод, нельзя забывать о необходимости проверки правильности 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 вычисляемыми величинами (например, $\sin^2 x + \cos^2 x = 1$).

Некоторые численные методы практически не требуют контроля правильности вычислений. К этому классу, в частности, относятся сходящиеся итерационные методы. Их достоинством является то обстоятельство, что получение ошибочного результата при одной из итераций не приводит к ухудшению окончательного результата вычислений, а лишь увеличивается количество итераций, которые должна выполнять машина. При быстрой сходимости итерационного процесса затраты машинного времени будут сравнительно небольшими. Приведем пример такого итерационного процесса. Пусть необходимо вычислить $y = \sqrt{x}$ с точностью до 0,01.

На ЭВМ для извлечения квадратного корня из x многократно применяется следующая процедура вычисления:

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

Всего $(i + 1)$ раз, пока не выполнится неравенство $|y_{i+1} - y_i| \leq \epsilon$, тогда y полагают равным 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 одной ошибкой.

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

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

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

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

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

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

Ответ: $y = 1,41$.

Процесс вычисления y с одним сбоем (пусть на шаге 3 вместо 1,43 было получено $y_3 = 0,43$, и с этого шага продолжим вычисления):

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

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

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

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

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

Ответ: $y = 1,41$.

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

1) необходимость сведения ее к выполнению последовательности арифметических действий (практически можно использовать и другие элементарные действия, для которых у составлены стандартные программы);

2) трудоемкость процесса программирования;

3) необходимость отладки программы на машине.

По условиям задачи всегда требуется знать и точность решения. Точность решения обычно задают значением максимально допустимой погрешности. В связи с этим необходимо выбирать, а иногда и специально разрабатывать соответствующий численный 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 складывается 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 алгоритмы небольшой связностью.

Связностью алгоритма называется количество данных, которые нужно накапливать в запоминающих устройствах машины для перехода от одного этапа вычислений к другому.

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

Рассмотрим пример.

Пусть требуется вычислить многочлен:

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0.$$

Если сначала вычислить все одночлены полинома, то для перехода от этапа вычисления величин $y_i = a_i x^i$ к этапу вычисления y требуется запомнить числа y_0, y_1, \dots, y_n , что приводит 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: $y = y_n$.

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

Почти все решения задач связаны с вычислением некоторых функций и, в частности, функции одной переменной $f(x)$.

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

1. Функция $f(x)$ задается в явном виде с помощью аналитической формулы, содержащей конечное число основных операций (арифметических, а также операций типа $|x|$, 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 заданной точностью. Поэтому многие способы задания функций, теоретически требующие выполнения конечного числа операций, на практике сводятся к способам с конечным числом операций. Например, задание функции в виде суммы членов сходящегося степенного ряда можно свести к первому случаю, т. е. в памяти машины хранится конечное число первых членов 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 базируется на двух основных факторах:

1) экономия количества ячеек памяти, необходимого для осуществления выбираемого 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 алгоритмы, требующие использования внешней памяти. Пусть функция задана явно с помощью формулы, содержащей конечное число операций и описанной различными элементарными функциями (x^k , $\ln x$, $\sin x$ и т. п.). Программы для вычисления элементарных функций имеют все современные ЭВМ. Поэтому на основе стандартных программ можно получить программы для вычисления функции. Если оказывается, что вычисления длительны и громоздки, то часто прибегают к описанию функции многочленами, таблицами и т. п.

Возможна и обратная ситуация, когда большие таблицы замещаются несколькими простыми функциями, чтобы ускорить время выборки данных и сэкономить количество ячеек памяти.

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

Рассмотрим реализацию способа задания функции на одном из подинтервалов путем выбора соответствующего многочлена.

Степень многочлена подбирается так, чтобы выполнялись условия:

1) степень должна быть возможно более низкой;

2) значение многочлена должно отличаться от соответствующего значения функции на величину, меньшую, чем некоторое малое число ϵ , характеризующее допустимую погрешность получаемых значений функции.

Если $P_n(x)$ — многочлен, то на соответствующем подинтервале должно выполняться неравенство $|f(x) - P_n(x)| < \epsilon$.

В этом случае в память вместо большой таблицы значений функции или большой программы для вычисления ее значений вводят небольшую таблицу коэффициентов многочленов и программу их вычисления. По значению x выбирается соответствующий многочлен и по стандартным программам вычисляется его значение.

Комбинаторные методы реализации алгоритмов. Характерной особенностью задач проектирования ВС и их компонентов на сверхбольших интегральных схемах (СБИС) является большая область поиска решений. Многие задачи являются NP -полными (число шагов в процессе поиска растет по экспоненте с возрастанием размерности задачи). Большинство же задач являются NP -сложными, т. е. более сложные, чем NP -полные.

Все задачи решаются методами комбинаторного поиска, основанными на построении дерева поиска. Эти методы порождают ветвящиеся процессы последовательного конструирования искомых решений, реализуя многошаговые переходы от начальной ситуации, отра-

жающей лишь исходные данные решаемой задачи, через некоторое множество промежуточных ситуаций (в каждой из которых решается одна и та же задача — изменяются лишь данные) к заключительным ситуациям, в которых будут представлены найденные решения. Решением является либо путь из корня дерева в конечную вершину, либо множество дуг, либо множество конечных вершин.

Разработка методов решения конкретных задач связывается с конкретизацией понятия "ситуация" и "элементарный шаг", с определением способов последовательного конструирования решений, с введением некоторых оптимизирующих правил. Общая схема решения задач может быть представлена деревом поиска следующим образом. Корню дерева ставится в соответствие исходная (начальная) ситуация, а остальным вершинам — ситуации, которые могут быть достигнуты в процессе поиска. Дуги соответствуют шагам шагающего процесса и связывают вершины, соответствующие таким ситуациям, одна из которых преобразуется в другую посредством одного шага. Если из вершины выходит несколько дуг, то говорят, что ситуация расщепляется, а если выходит лишь одна дуга, то имеет место редуцирование ситуации. Некоторые из конечных вершин дерева поиска представляют решения. В процессе поиска требуется найти все ведущие к ним пути или один из них (лучший или оптимальный в каком-либо смысле).

Разработка алгоритмов осуществляется по такой схеме: формулируются понятия текущей ситуации и элементарного шага, правила редуцирования, возврата и прекращения поиска. Полный перебор в порожденном этими правилами дереве поиска дает оптимальное решение. Затем формируются некоторые усеченные деревья поиска, точнее описываются правила их построения, обход которых даст некоторое приближенное решение.

При этом актуальной является задача сокращения перебора. Все способы ее решения имеют одну основу. Вводится оценка решения. Далее ищется одно решение и вычисляется его оценка h . Затем ищутся другие с $h' < h$. При этом оценка может быть либо задана априори, либо зависеть от предыдущего шага или от результатов выполнения нескольких предыдущих шагов. Далее вводится количество k вершин, рассматриваемых на одном уровне, при этом k может быть константой или зависеть от глубины шага.

Для примера рассмотрим задачу разбиения множества строк булевой матрицы на минимальное число допустимых подмножеств (будем называть их строчными минорами), где допустимым считается минор, у которого число ненулевых столбцов не превышает k . Эта задача имеет практическое применение при синтезе логической схемы, реализующей заданную систему булевых функций и состоящей из блоков, количество входных полюсов которых ограничено числом k .

Текущая ситуация характеризуется множеством Q допустимых миноров, образованных к данному моменту, и множеством R строк матрицы, еще не включенных в миноры (в начальной ситуации $Q = \emptyset$ и $R = R_0$ — множество строк исходной матрицы).

На множестве всех пар (R', r) , где $R' \subseteq R_0$ и $r \in R_0$, определяется бинарное отношение совместимости \circ : если минор, образованный множеством $R' \cup \{r\}$ допустимый, то $R' \circ r$. Текущую ситуацию опишем булевой матрицей $S = [Q \circ R]$, задающей отношение \circ миноров Q со строками из R . Для начальной ситуации матрица S не содержит ни одной строки. Возможность разбиения q в начальной ситуации полагаем $|R_0| + 1$. Элементарным шагом поиска является включение строки из R в один из миноров Q или образование нового минора, состоящего из этой строки.

Алгоритм решения этой задачи определяется следующими правилами поиска.

Правило редуцирования. Если в матрице S существует нулевой столбец, то образуется новый минор и в него включается строка из R , соответствующая найденному столбцу. Если столбцов несколько, то берется первый. В начальной ситуации выбирается первая строка из R_0 .

Правило расщепления 1. В матрице S находим минимальный по числу единиц столбец. Будет соответствовать строка $r \in R$, для которой число совместимых с ней миноров из Q t наименьшим. Если минимальных столбцов несколько, то выбираем первый. Текущая ситуация расщепляется на ситуации, соответствующие вариантам включения строки r в совместимые с ней миноры и варианту образования нового минора.

Правило получения решения. Если в заключительной ситуации (для которой $R = \emptyset$) q , то Q запоминается в качестве текущего решения и $q = |Q|$.

Правило возврата. В заключительной ситуации, а также в промежуточной ситуации с помощью текущего группирования q , осуществляется возврат к последней ситуации с нерешенным перебором, для которой $|Q| < q$.

Правило окончания поиска. Если ситуация с незавершенным перебором вариантов расщепления нет, то закончить поиск. Имеющееся Q и является искомым.

Описанный алгоритм является точным, так как в каждой ситуации рассматривается вся совокупность вариантов расщепления, а в ней всегда есть вариант продолжения поиска, который ведет к минимальному разбиению.

Введение дополнительных правил позволяет как сократить перебор, так и получить значение заданного свойства.

Правило расщепления 2. В матрице S находим минимальный по числу единиц строку. Будет соответствовать минор Q_i , для которого минимально число совместимых с ним строк. Если минимальных миноров несколько, то выбираем первый. Текущая ситуация расщепляется на ситуации, соответствующие вариантам включения в минор Q_i одной из найденных строк, и варианту, когда в выбранный минор не включается ни одна из этих строк.

Пусть M^* — матрица, l -я строка которой есть покомпонентная дизъюнкция всех строк минора Q_i . Теперь можно ввести новое правило редуцирования.

Правило редуцирования 2. Если в матрице M^* существует строка с номером j , поглощающая некоторую строку r из R (булев вектор a поглощает булев вектор b ; если все компоненты вектора a , соответствующие единичным компонентам вектора b , также имеют значение 1), то строка r включается в минор Q_i .

Повышение размерности решаемых задач такого рода достигается разработкой приложенных алгоритмов, которые не гарантируют получение минимального группирования, позволяют стоить достаточно качественные решения за приемлемое для практики время. Таким алгоритмам соответствуют некоторые усеченные деревья поиска. В вырожденном случае такое дерево будет состоять из одной лишь ветви, и соответствующий ей алгоритм идет цепным (без возврата), основанным лишь на редуцировании текущей ситуации. Особенно, наиболее приемлемыми с точки зрения практики являются именно такие алгоритмы, и алгоритмы с сильно ограниченным перебором. При их построении основное внимание уделяется нахождению эвристических правил построения пути из корня дерева поиска в его целевую вершину, дающую наилучшее приближение к оптимальному.

Проблема разрешимости и оценка сложности алгоритма. Под алгоритмической проблемой разрешимости понимается задача построения алгоритма для решения заданной массовой задачи. В случае, когда искомым алгоритм невозможен, говорят, что данная алгоритмическая проблема неразрешима, т. е. невозможно указать единый способ решения всех задач данной серии (для отдельных задач решение возможно). Классические неразрешимые задачи на построение при помощи циркуля и линейки: удвоение куба, трисекция угла, квадратура круга. Не разрешимой является проблема выводимости для исчисления предикатов, и проблема Гильберта (решение диофантовых уравнений). Доказано, что не существует алгоритма, который по многочлену от нескольких переменных с целыми коэффициентами определял бы, есть ли у него целочисленные корни. Неразрешимость элементарной теории

арифметики: не существует алгоритма, который бы по утверждению о натуральных числах выдавал ответ на вопрос: истинно оно или ложно?

Для доказательства невозможности существования алгоритма, дающего решение в или иной серии задач, стало возможным благодаря определению алгоритма через точное тематическое описание класса вычислимых функций натурального аргумента (частично курсивные и рекурсивные функции) и благодаря уточнению математического определения класса вычислительных процессов, реализуемых абстрактными вычислителями (машин Тьюринга). Черчем А. в 1936 г. была установлена их эквивалентность.

С практической точки зрения эффективность алгоритма может быть ограничена тем как много времени и памяти (ОЗУ) требуется для выполнения алгоритма. Продолжительность реализации алгоритма и объем требуемой памяти зависят от характеристик компьютера и установленного программного обеспечения. Чтобы сравнить различные алгоритмы, необходимо условиться о стандартной модели и условиях вычислений. Модель определяет примитивные операции, их стоимость, а точнее объем требуемой памяти и время выполнения. Время выполнения алгоритма это сумма всех времен выполнения примитивных операций этого алгоритма. Объем требуемой памяти это пространство памяти, необходимое для хранения всех данных, размещаемых в памяти в ходе исполнения алгоритма.

Ниже приведены стандартные понятия для оценки сложности. Для каждого отдельно случая оценки существует определенный набор данных, который называется входными данными. Размер входных данных определяется объемом памяти необходимым для размещения этих данных. В большинстве приложений не интересна точная сложность алгоритма для всех возможных входных значений, а интересно изменение оценки сложности при увеличении размера входного набора. Однако время выполнения также может зависеть непосредственно от качества входного набора: т. е. от того, как точки или отрезки расположены в пространстве, от длины отрезков и так далее. Для вычислительных процессов важно знать сложность выполнения алгоритма при входном наборе размерности n . Функция оценки сложности по времени и занимаемому объему памяти позволяет выявить возможность реализации алгоритма для наилучшего случая из всех допустимых вариантов входного набора размера n . Точно также вводится функция сложности алгоритма. Сложность для худшего случая это пессимистическая оценка сложности алгоритма. В алгоритмическом анализе сложность по времени и по объему памяти выражается как функция от размерности входного набора, умноженная на постоянный коэффициент. $O(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C и N_0 и $g(N) \leq C \cdot f(N)$ для всех $N \geq N_0$. $\Omega(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C и N_0 и $g(N) \geq C \cdot f(N)$ для всех $N \geq N_0$. $\Theta(f(N))$ обозначается множество всех функций $g(N)$ таких, что существуют положительные константы C_1 , C_2 и N_0 и $C_1 \cdot f(N) \leq g(N) \leq C_2 \cdot f(N)$ для всех $N \geq N_0$.

Алгоритм с оценкой сложности по времени $O(nc)$, где c — некоторая постоянная, называют полиномиальным.

Алгоритмы, которые имеют полиномиальную сложность по времени, являются эффективными. Полиномиально разрешимые задачи можно успешно решать на компьютере даже в тех случаях, когда они имеют большую размерность. Для других задач не удается найти полиномиальный алгоритм, поэтому их называют трудноразрешимыми. К ним относятся большое число задач алгебры, математической логики, теории графов, теории автоматов и других разделов дискретной математики. В большинстве своем это так называемые переборные задачи. Переборный алгоритм имеет экспоненциальную сложность и может хорошо работать только для небольших размеров задачи $O(2^N)$.

Класс задач, который не допускает поиска эффективных алгоритмов, известен как NP -полных задач. Но, может быть, возможно, найти приближенное к оптимальному решению за полиномиальное время. На практике поиск решения, приближенного к оптимальному решению, может быть вполне достаточно. Алгоритм, который позволяет получить приближенное к оптимальному решению, называется алгоритмом аппроксимации (ближения).

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 математической формулировки понятия алгоритма. В связи с этим перейдем к более строгому описанию алгоритмов.

Удобными для этой цели оказались понятия и средства, накопленные в процессах преобразования различных слов как цепочек любых символов, что фактически использовалось и при решении задачи поиска пути в лабиринте. На абстрактном уровне вычислительная машина всегда выполняет операции по переработке исходного текста (набор букв и чисел, описывающих условие задачи) в некоторый заключительный текст (описание результата решения задачи). В машине буквы и цифры задаются в виде последовательностей нулей и единиц, поэтому решение задачи фактически сводится к переработке 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, составляющие алфавит, будем называть буквами.

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

Любая конечная последовательность букв некоторого алфавита называется словом в алфавите. Например, в алфавите $A = \{a, b, c\}$ словами будут последовательности $ab, ac, c, bbbb$ 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 $H \rightarrow M, \dots, C \rightarrow T$, где H, M, \dots, C, T – слова этого алфавита.

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

Например, $H = ab, M = bcb, K = abc bcbab$. Заменяя в слове K слово H на M , можно получить такие слова: $bcbcbcbab$ или $abc bcbcb$ и, наоборот, заменив M на H , получим $aabcbab abcabab$.

Подстановка $ab \rightarrow bcb$ недопустима к слову $bacb$, т. к. ни ab , ни bcb не входит в это слово. 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 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 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 $acdbe$ – смежные (подстановка $bc - cb$). Слова $abcde$ и $cabde$ эквивалентны.

Ассоциативному исчислению можно поставить в соответствие бесконечный лабиринт приняв каждое слово данного алфавита за площадку и соединив смежные площадки (слов 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 $H \rightarrow M$. 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 конечного лабиринта, становится непригодным из-за невозможности в конечное время обследовать лабиринт. Многие конструкторские и другие задачи сводятся к такой же проблеме (конструкция слова).

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

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

Вообще любой процесс вывода формул, математические выкладки и преобразования также являются дедуктивными цепочками в выбранном подходящим образом ассоциативном исчислении. Алгоритмические процессы также могут трактоваться как ассоциативное исчисление.

Естественно предположить, что построение ассоциативных исчислений может быть универсальным методом для задания детерминированного процесса «переработки» исходных данных, т. е. для задания алгоритма. Для этой цели необходимо уточнить понятие алгоритма.

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

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

Пример.

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

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

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

Рассмотрим применение системы подстановок B на конкретном примере для слов $babaac$ и $bcaabc$:

$babaac \rightarrow bbcaaac \rightarrow$ остановка
 $bcaabc \rightarrow bcacbcac \rightarrow bcaccacac \rightarrow bcacabc \rightarrow$ процесс бесконечен, т. е. остановка не найдется, так как мы получили исходное слово.

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

Задается алфавит и схема подстановок B . Для произвольного слова P просматриваются формулы подстановок в том порядке, в каком они заданы в схеме B , и разыскивается формула, левая часть которой входит в P . Если такой формулы нет, то процесс обрывается. В противном случае берется первая из таких формул и делается подстановка ее правой части вместо первого вхождения ее левой части в P , что дает слово P_1 , с которым поступают аналогично. Обрывается процесс в двух случаях: во-первых, когда получим такое слово P_i , что ни одна из формул не входит в него; во-вторых, когда при получении P_i нам придется применять последнюю формулу.

Различные нормальные алгоритмы отличаются друг от друга лишь алфавитами и системами допустимых подстановок. Приведем пример нормального алгоритма, описывающего процесс сложения чисел:

Алфавит	Система подстановок v
$A\{1, +\}$	$1 + \rightarrow +1$
	$+1 \rightarrow 1$
	$1 \rightarrow 1$

Слово P : $11 + 11 + 111$.

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

$P = 11 + 11 + 111$	$P_5 = + 1 + 111 111$
----	----
$P_1 = 1 + 111 + 111$	$P_6 = ++ 1111111$
----	----
$P_2 = + 1111 + 111$	$P_7 = + 1111111$
----	----
$P_3 = + 111 + 1111$	$P_8 = 1111111$
----	----
$P_4 = + 11 + 11111$	$P_9 = 1111111$
----	----

Мы видим, что этот алгоритм суммирует количество единиц. Следует обратить особое внимание на элементарность и однозначность шагов, выполняемых при реализации алгоритма. Естественно, что представления любого алгоритма в такой форме достаточно, чтобы попытку его реализации некоторому автомату. Нормальный алгоритм Маркова можно рас-

смаивать как стандартную форму для задания любого алгоритма. Такая форма на практике используется лишь в теоретических построениях. В процессе построения теории чисел алгоритмов выяснилось, что любой логический алгоритм можно простыми методами свести к численному. Таким образом, теория численных алгоритмов (она тождественна понятию «теория вычислимых функций») стала универсальным аппаратом для исследования алгоритмических проблем.

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

Докажем теорему.

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

$$A_0, A_1, A_2, \dots, A_n, \dots$$

Аналогично записи возможных решений также включим в пронумерованную последовательность $B_0, B_1, B_2, \dots, B_m, \dots$

После проведения нумерации, очевидно, что любой алгоритм, перерабатывающий пары условий A_n и записей решений B_m , можно свести к вычислению значений некоторой функции $m = \Phi(n)$, так как после введения нумерации можно иметь дело лишь с соответствующими номерами записей условий и решений, а не с самими записями. Тогда можно говорить об алгоритме, перерабатывающем номер записи условия в номер записи решения, который будет численным алгоритмом.

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

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

Наоборот, если есть алгоритм вычисления функции $\Phi(n)$, то имеется и алгоритм решения исходной задачи, так как по записи условий задачи можно найти соответствующий номер n^* , затем вычислить $m^* = \Phi(n^*)$ и по m^* определить запись решения.

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

$$n = 2^{a_1} \cdot 3^{a_2} \cdot 5^{a_3} \cdot \dots \cdot P_{m-1}^{a_m}, \text{ где } P_0 = 2, P_1 = 3, \dots, P_{m-1}, \text{ т. е. } P_m - m\text{-е простое число.}$$

В силу теоремы о единственности разложения любого числа на простые множители, следует, что каждому числу n однозначно соответствует набор $A = \{a_1, a_2, \dots, a_m\}$, и, наоборот, каждому набору A однозначно соответствует число n .

Например, если $n = 60$, то $60 = 2^2 \cdot 3^1 \cdot 5^1$, т. е. $a_1 = 2, a_2 = 1, a_3 = 1$.

Если $n = 98$, то $98 = 2^1 \cdot 3^0 \cdot 5^0 \cdot 7^2$, т. е. $a_1 = 1; a_2 = 0, a_3 = 0, a_4 = 2$.

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

Приведем несколько примеров:

1. Каждой паре чисел a_1 и a_2 , для которой мы ищем ее наибольший общий делитель, может быть поставлен в соответствие геделевский номер этой пары $n = 2^{a_1} \cdot 3^{a_2}$. Алгоритм Евклида сведется к вычислению функции $q = \Phi(n)$.

2. Пусть требуется перенумеровать все слова в некотором алфавите A . Это легко сделать, поставив в соответствие каждой букве алфавита какое-либо число. Тогда каждому слову в алфавите A будет соответствовать последовательность чисел. Проводя затем обычный способ геделизации, получим геделевский номер этой последовательности.

Таким образом, не только арифметические алгоритмы сводятся к вычислению значений целочисленных функций. Любой нормальный алгоритм Маркова с помощью геделизации может быть также сведен к вычислению значений целочисленных функций. Поэтому алгоритм вычисления целочисленных функций можно считать универсальной формой алгоритма. Естественно, что эти утверждения верны для случаев, когда всех условий задачи может быть бесконечно много, но множество это все же счетное.

1.5. Абстрактные модели автоматов

Понятие автомата. Типы автоматов. Автомат — это алгоритм, определяющий некоторое и, возможно, преобразующий его в другое множество. Неформальное описание автоматов выглядит следующим образом: автомат имеет входную ленту, управляющее устройство с конечной памятью для хранения номера состояния, а также может иметь вспомогательную (рабочую) и выходную ленты.

Существует два типа автоматов:

- 1) распознаватели-автоматы без выхода, которые распознают, принадлежит ли входная пара заданному множеству L ;
- 2) преобразователи-автоматы с выходом, которые преобразуют входную цепочку x в парку y при условии, что $x \in L$.

Входную ленту можно рассматривать как линейную последовательность ячеек, причем каждая ячейка может хранить один символ из некоторого конечного входного алфавита. Работа автомата бесконечна, но занято на ней в каждый момент времени только конечное число ячеек. Граничные слева и справа от занятой области ячейки могут занимать специальные концевые маркеры. Маркер может стоять только на одном конце ленты или может отсутствовать вообще.

Входная головка в каждый момент времени читает (обозревает) одну ячейку входной ленты. За один такт работы автомата входная головка может сдвинуться на одну ячейку влево или остаться на месте, при этом она выполняет только чтение, т. е. в ходе работы автомата символы в ячейках входной ленты не меняются.

Рабочая лента — это вспомогательное хранилище информации. Данные с нее могут читаться автоматом, могут и записываться на нее.

Управляющее устройство — это программа, управляющая поведением автомата. Оно представляет собой конечное множество состояний вместе с отображением, описывающим, как меняются состояния в соответствии с текущим входным символом, читаемым входной головкой, и текущей информацией, извлеченной с рабочей ленты. Управляющее устройство также определяет направление сдвига рабочей головки и то, какую информацию записать на рабочую ленту.

Автомат работает, выполняя некоторую последовательность рабочих тактов. В начале такта читается входной символ и исследуется информация на рабочей ленте. Затем, в зависимости от прочитанной информации и текущего состояния, определяются действия автомата:

- 1) входная головка сдвигается вправо или стоит на месте;
- 2) на рабочую ленту записывается некоторая информация;
- 3) изменяется состояние управляющего устройства;
- 4) на выходную ленту (если она есть) записывается символ.

Поведение автомата удобно описывать в терминах конфигурации автомата, которая включает в себя:

- а) состояние управляющего устройства;
- б) содержимое входной ленты с положением входной головки;
- в) содержимое рабочей ленты вместе с положением рабочей головки;
- г) содержимое выходной ленты, если она есть.

Управляющее устройство может быть недетерминированным. В таком случае для каждой конфигурации существует конечное множество возможных следующих тактов, любой из которых автомат может сделать, исходя из этой конфигурации. Управляющее устройство будет детерминированным, если для каждой конфигурации будет возможен только один следующий такт.

Существуют следующие типы автоматов:

- 1) машины Тьюринга и Поста;
- 2) линейно-ограниченный автомат;

3) автомат с магазинной памятью (МП-автомат);

4) конечный автомат.

Сложность рабочей ленты определяет сложность автомата. Так, например:

1) машины Тьюринга и Поста имеют неограниченную в обе стороны ленту;

2) у линейно-ограниченного автомата длина рабочей ленты представляет собой линейную функцию длины входной цепочки;

3) у МП-автомата рабочая лента работает по принципу магазина LIFO («послед пришел – первый вышел»);

4) у конечного автомата рабочая лента отсутствует.

Формальное определение автомата. Неинициальный автомат – это пятерка в $A = (Q, X, Y, \delta, \gamma)$, где:

Q – множество состояний (алфавит состояний);

X – входной алфавит;

Y – выходной алфавит;

δ – функция переходов (отображение $Q \times X \rightarrow Q$);

γ – функция выходов (отображение $Q \times X \rightarrow Y$).

Функционирование автомата можно задать множеством команд вида $qx \rightarrow py$, $q, p \in Q, x \in X, y \in Y$.

Пусть на некотором такте t , управляющее устройство находится в состоянии q , а входной ленты читается символ x . Если в множестве команд есть команда $qx \rightarrow py$, то на такте t , на выходную ленту записывается символ y , а к следующему такту $t+1$ управляющее устройство перейдет в состояние p , т. е.:

$$y(t) = \gamma(q(t), x(t)), q(t+1) = \delta(q(t), x(t)).$$

Если же команда $qx \rightarrow py$ отсутствует, то автомат оказывается заблокированным и не реагирует на символ, принятый в момент t , а также перестает воспринимать символы в последующие моменты времени.

В соответствии с определением неинициального автомата в начальный момент состояние автомата может быть произвольным.

Если зафиксировано некоторое начальное состояние $q_0 = q(0)$, то такой автомат называют инициальным, т. е. $q(0) = q_0$.

Автоматы Мили и Мура. На практике наибольшее распространение получили два класса автоматов – автоматы Мили и Мура.

Закон функционирования автомата Мили задается уравнениями:

$$q(t+1) = \delta(q(t), x(t)); y(t) = \gamma(q(t), x(t)), t = 0, 1, 2, \dots$$

Закон функционирования автомата Мура задается уравнениями:

$$q(t+1) = \delta(q(t), x(t)); y(t) = \gamma(q(t)), t = 0, 1, 2, \dots$$

Из сравнения законов функционирования видно, что, в отличие от автомата Мили, в выходной сигнал в автомате Мура зависит только от текущего состояния автомата и в явном виде не зависит от входного сигнала. Для полного задания автомата Мили или Мура достаточно к законам функционирования необходимо указать начальное состояние и определить внутренний, входной и выходной алфавиты. Между автоматами Мили и Мура существует соответствие, позволяющее преобразовать закон функционирования одного из них в другой и обратно. Автомат Мура можно рассматривать как частный случай автомата Мили, имея в виду, что последовательность состояний выходов автомата Мили опережает на один такт последовательность состояний выходов автомата Мура, т. е. различие между автоматами Мили и Мура состоит в том, что в автоматах Мили состояние выхода возникает одновременно с вызывающим его состоянием входа, а в автоматах Мура – с задержкой на один такт, т. е. как в автоматах Мура входные сигналы изменяют только состояние автомата.

При представлении автомата Мура графом дуги помечаются символами входного алфавита, а каждая вершина графа – состоянием и символом выходного алфавита. При формальном сравнении определений автоматов Мили и Мура может показаться, что автомат

а может быть задан как входонезависимый автомат Милли. Однако это не соответствует образу функционирования автоматов Мура в соответствии с введенным определением. В автомате Мура реализована иная временная связь между переходами из одного состояния в другое и выходом, по сравнению с автоматом Милли, у которого выход, соответствующий данному входу и определенному состоянию, порождается во время перехода автомата в следующее состояние. У автомата Мура сначала порождается выход, а потом — переход в следующее состояние, причем выход определяется только состоянием автомата.

Комбинационные автоматы. Автомат, значения выхода Y которого зависят только от текущих входов X в данный момент времени и не зависят от входных воздействий в предыдущие моменты времени, называется комбинационным или автоматом без памяти. В структурных моделях автоматы этого класса называются комбинационными схемами. Их функционирование описывается математической моделью вида $Y = f(X)$. Функция, описывающая это соотношение для одного выхода, называется булевой или логической функцией. Такие функции задаются с помощью полностью или не полностью определенных таблиц, которые называются таблицами истинности и таблицами решений соответственно. Более компактной формой представления является задание булевых функций в виде булевых формул, которые в отличие от таблиц всегда определены на всех 2^n наборах (n — число входных переменных). В табл. 1.5.1 представлены примеры функций двух переменных x_1 и x_2 .

Таблица 1.5.1 — Логические функции

Объединение аргументов $x_2 x_1$	Функция И $y = x_2 \& x_1$	Функция ИЛИ $y = x_2 + x_1$	Функция И-НЕ $y = \neg(x_2 \& x_1)$	Функция ИЛИ-НЕ $y = \neg(x_2 + x_1)$	Функция исключающее ИЛИ $y = x_2 \oplus x_1$
00	0	0	1	1	0
01	0	1	1	0	1
10	0	1	1	0	1
11	1	1	0	0	0

При представлении логической функции математическим выражением используют два ее представления.

Совершенной дизъюнктивной нормальной формой (СДНФ) называется логическая функция, заданная элементарными логическими произведениями, в каждое из которых аргумент или его отрицание входят 1 раз. СДНФ может быть получена из таблицы истинности следующим образом: для каждого набора аргументов, на котором функция равна «1», записывают элементарное произведение переменных, причем переменные, значение которых равно нулю, записывают с инверсией ($\neg x$). Полученные произведения, которые носят название конъюнктивент единичных, суммируют. Далее по законам алгебры логики это выражение минимизируется с целью сокращения в нем количества элементарных произведений, так и некоторых переменных в них, что упрощает техническую реализацию комбинационных автоматов.

Совершенной конъюнктивной нормальной формой (СКНФ) называется логическое произведение элементарных сумм, в каждую из которых аргумент или его отрицание входят 1 раз. СКНФ может быть получена из таблицы истинности: для каждого набора аргументов, на котором функция равна 0, составляют элементарную сумму, причем переменные, значение которых равно 1, записываются с отрицанием. Полученные суммы, которые носят название конъюнктивент нуля, объединяют операцией логического умножения, СКНФ далее по законам алгебры логики минимизируется.

Задание булевых функций в виде графа переходов нерационально, так как для этого для автоматов отсутствует необходимость в отражении динамики переходов.

Описание динамической системы. Развитие вычислительной техники привело к дальнейшему усложнению задач, решаемых с ее помощью. На первом этапе эффективное использо-

вание ЭВМ проявилось в автоматизации умственной деятельности человека, но вместе с мтиоризацией элементов ЭВМ появилась возможность сочетания автоматизации производных процессов с одновременной автоматизацией управления ими непосредственно на объекте 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, способностью реагировать на те же внешние сигналы, что и человек, и выполнять простейшие 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 может иметь манипуляторы (руки), педипуляторы (ноги), колесные механизмы и т. п.); блок управления осуществляющий целенаправленное поведение робота на основе сложной системы распознавания образов.

Многое из того, что человек делает с легкостью, для машин оказывается трудным. Очень нелегко поручить роботу даже простую для человека задачу: сложить из стандартных элементов детского конструктора, например, домик. Есть свои определенные проблемы и в создании механических частей роботов-манипуляторов. Создание конструкции манипулятора является самостоятельной проблемой. Люди спокойно берут в руки многие хрупкие предметы, например куриное яйцо, и не ломают их: человек подсознательно оценивает малые деформации предмета и его прочность, а для робота — это сложная задача.

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

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

В настоящее время большинство находящихся в эксплуатации роботов и робототехнических систем используется в машиностроении для сварки, механической обработки, подачи и перемещения деталей, при измерениях и сборке.

Промышленным роботом называется программируемый многофункциональный манипулятор, предназначенный для перемещения материалов, деталей, инструмента или специализированных устройств по переменным программируемым траекториям и выполнения данных задач.

Используемые сейчас роботы в основном выполняют узкий круг задач. Это объясняется как трудностями в создании механических элементов робота, так и его электронных программных средств управления. В основном это относится к задачам по распознаванию образов и принятию решений, выбору траекторий движения и т. п. Ведь интеллектуальный робот должен обладать способностью распознавать обстановку и автоматически вырабатывать решения о своих дальнейших действиях для выполнения нужных технологических операций.

й в неопределенной или меняющейся ситуации. Еще более интеллектуальным должен быть робот для ведения домашнего хозяйства и имитации сложных механических действий века.

Одной из трудных проблем реализации проекта таких роботов будет обеспечение контакта хозяина с роботом на уровне команд, подаваемых на естественном языке. Вообще 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 управления рабочими органами. Абстрагируясь от конструктивных особенностей промышленных роботов, мы остановимся на математических аспектах управления ими и рассмотрим вопросы программирования их действий на основе конечных автоматов. Работа ЭВМ, робота и всякого другого реального устройства рассматривается во времени. Время обычно считается непрерывным и предполагается изменяющимся от настоящего к будущему. Однако при рассмотрении ряда устройств удобно вводить воображаемое дискретное время.

Пусть полуось времени $(0, \infty)$ каким-то образом разбита на бесконечное число произвольной длины. Границы отрезков для простоты запишем как ряд целых неотрицательных чисел $0, 1, 2, \dots, i, \dots$, полагая, что воображаемое дискретное время принимает только целочисленные значения. Моменты времени $0, 1, 2, \dots, i$ назовем тактами.

Текущий такт, соответствующий настоящему моменту времени, обозначим через i , а следующие ему и последующие такты — соответственно через $(i - 1, i - 2, \dots)$ и $(i + 1, i + 2, \dots)$.

Будем рассматривать функционирование в дискретном времени динамических систем, задаваемых координатами, каждая из которых изменяется на конечном множестве. Всякая динамическая система получает ряд внешних возмущений: звуковые, электрические, световые и т. п. сигналы. Будем считать, что их число также конечно и каждое из возмущений задано на конечном множестве.

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

Рассмотрим множества $\{C\} = \{C_1, C_2, \dots, C_k\}$ и $\{P\} = \{P_1, P_2, \dots, P_r\}$, состоящие из соответственно 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», указать, как определяется состояние системы в каждый такт. Ограничимся рассмотрением одной из простейших систем.

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

В соответствии с определением конечного автомата состояние C^i на любом такте i однозначно определяется состоянием C^{i-1} в предыдущий такт $(i - 1)$ и входом P^i в рассматриваемый 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 рекуррентное соотношение, так как при известных P^i и C^0 можно отыскать C^1 , приняв $i = 1$. Далее по C^1 можно отыскать C^2 и т. д., для всех элементов множеств $\{C\}$ и $\{P\}$.

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

Фиксируя разные символы из множества $\{P\}$, получим r автономных автоматов. Для описания r автономных автоматов достаточно описать поведение конечного автомата. Автономный автомат будем считать полностью заданным, если известны $\{P\}$, $\{C\}$ и $F(C^{i-1}, P_C)$. Функцию F можно задавать различными способами, чаще всего используются таблицы.

Пусть множество входов $\{P\}$ состоит из r символов. Тогда таблица функций переходов строится следующим образом: вычерчивается прямоугольник, содержащий $(r+1)(k+1)$ клеток. В первую клетку в верхнем левом углу прямоугольника соответствуют записываются в верхней и нижней частях ее наименования координат P, C^{i-1} . Вправо от P по строке по клеточке вписываются все возможные значения P , а по столбцу вниз – все возможные значения C . На пересечении столбца P и строки C вписываются соответствующие значения $F(C^{i-1} P)$ при $P^i = P$ и $C^{i-1} = C_m$. В итоге получим таблицу, которую обычно называют основной таблицей конечного автомата (табл. 1.5.2).

Таблица 1.5.2 – Основная таблица конечного автомата

$C^{i-1} \backslash P^j$	P_1	P_2	...	P_r
C_1	C_k	C_1		C_3
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 затруднений.

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

Можно заметить, что каждый из столбцов P_1, P_2, \dots, P_r этой таблицы является основной таблицей автономного автомата. Такие автоматы можно задавать с помощью направленных графов так, чтобы наблюдалось взаимно-однозначное соответствие вершин графа и символов из алфавита C .

Переходу автономного автомата из состояния C_i в C_j будет соответствовать направленная стрелка (рис. 1.5.1).

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

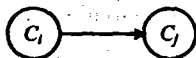


Рисунок 1.5.1 – Элемент задания конечного автомата автономными переходами

Покажем на примере небольшой таблицы, как ее описать на языке графов. Пусть таблица 1.5.3 задает конечный автомат A с двумя входами и четырьмя состояниями.

Таблица 1.5.3 – Основная таблица автомата A

$C^{i-1} \backslash P^j$	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 автономный автомат, над каждым из графов поставим наименования соответствующих входов.

Исходная таблица может быть представлена двумя графами $A(P_1)$ и $A(P_2)$, как это сделано на рис. 1.5.2.

Можно заметить, что от каждого из кружков на рис. 1.5.3. отходит только по одной стрелке. Этот факт следует из определения детерминированного конечного автомата.

Большой компактности таблицу 1.5.2. можно представить одним объединенным графом, который называется диаграммой состояний конечного автомата. Как видно из рис. 1.5.3, на объединенном графе над каждой стрелкой записывается тот вход P , который вызывает переход из одного состояния в то, к которому направлена стрелка. Диаграмма состояний автомата весьма наглядна. Допустим, необходимо установить, каким будет внутреннее состояние автомата A через t тактов при входной последовательности $\{P\} = \{P_1, P_1, P_2\}$ и начальном состоянии C_3 . Решение задачи отыскивается простым обходом от начального состояния $C^0 = C_3$ по стрелке P_1 к следующему состоянию C_4 , от него снова по стрелке P_1 к состоянию C_2 , а от C_2 по стрелке P_2 опять возвращаемся в состояние C_2 , оно и будет искомым.

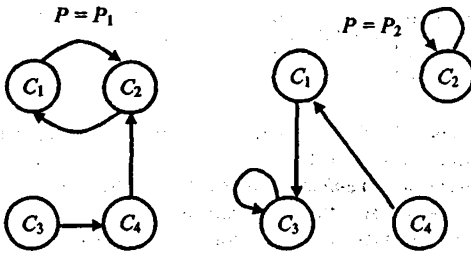


Рисунок 1.5.2 – Задание автомата диаграммой

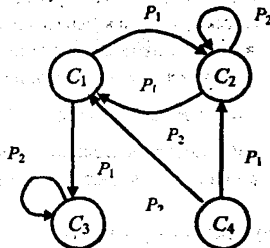


Рисунок 1.5.3 – Объединенный граф автомата

В ряде случаев желательно выполнять подобные операции более формальным путем, чтобы использовать вычислительные средства для отыскивания нужного результата. Для этой цели удобно матричное задание конечного автомата.

Графу P_c автономного автомата можно поставить во взаимно однозначное соответствие матричную матрицу, состоящую из нулей и единиц. Пронумеруем все возможные состояния автомата от 1 до k . Составим матрицу $k \times k$, такую, что в строке i на месте j будем записывать 1, если существует стрелка от i к j на графе A , а в противном случае 0. Обозначим эту матрицу через $A(P_c) = (a_{ij}(P_c))$.

Очевидно, сумма $(a_{ij}(P_c)) = 1$, так как по определению автономного детерминированного автомата от каждого C должна отходить одна и только одна стрелка, и, естественно, в каждой строке i матрицы $A(P_c)$ найдется один элемент $a_{ij}(P_c) = 1$, а остальные будут равны нулю.

При матричном задании конечного автомата начальное состояние C^0 также задается матрицей $A(P_0)$, состоящей из одной строки, в которой записываются $(k-1)$ нулей, а на месте C^0 номером, соответствующим номеру C^0 , – единица.

Пусть заданы входная последовательность $\{P\} = \{P^1, P^2, \dots, P^i\}$ и матрицей A (P_0) начальное состояние C^0 . Тогда состояние автомата на такте i устанавливается как результат вычисления произведения:

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

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

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

$$A(P_1) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \text{ и } A(P_2) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

а начальному состоянию $C^0 = C_3$ — матрица $A(P^0) = (0010)$. Тогда, используя формулу, получим искомый результат: $A_3(P^3) = (0100)$.

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

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 состояния 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° . Автомат служит управляющей системой для робота, который может двигаться за один такт на единичный шаг вперед, л

поворачиваться на 90° по часовой стрелке, либо захватывать деталь, либо отпускать деталь, стоять. Иными словами, по одной команде «двигаться вперед» робот проходит единичный шаг, а по команде «поворот» – поворачивается на 90° по часовой стрелке.

Команда на исполнительные механизмы формируется благодаря анализу входа и внутреннего состояния конечного автомата. Один и тот же вход может определять разные команды исполнительного механизма из-за разных состояний конечного автомата. Покажем, конечный автомат A (задается конструктором), приведенный в нижеследующей таблице, точен для решения поставленной задачи.

Таблица 1.5.3 – Управляющий автомат A

C \ P	P_1	P_2	P_3	P_4
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

Для понимания последующих записей приведем некоторые соглашения по интерпретации команд автомата для исполнительных органов робота. Состояние C_1 будет соответствовать состоянию покоя робота, C_4 – прямолинейному движению, C_3 – повороту, C_4 – захвату или, C_5 – выгрузке детали.

Представим себе, что нам задан контур (рис. 1.5.4) и требуется, захватив деталь в начальной точке с координатами $(X_0; Y_0)$, перенести ее в точку с координатами $(X_1; Y_1)$ и там разгрузить, а затем снова вернуться в точку $(X_0; Y_0)$. Ориентация робота в направлении возможного движения по прямой указана стрелкой на его кожухе.

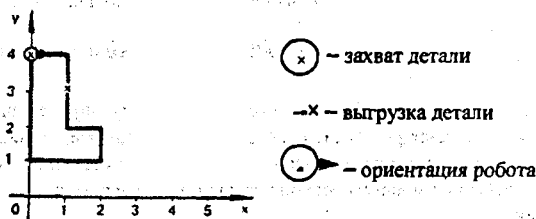


Рисунок 1.5.4 – Схема движения робота...

В нашем случае $(X_0; Y_0) = (0; 4)$ и $(X_1; Y_1) = (1; 3)$. Если робот, сориентированный в направлении Ox , находится в состоянии покоя (C_1) в точке $(0; 4)$ и готов к движению по прямой от точки $(0; 4)$ к точке $(1; 4)$, то нижеследующая программа обеспечит выполнение задачи для наглядности будем писать в скобках соответствующие символы состояния, на которые входное воздействие символов P_i в виде цепочки смены состояний):

$P_3 P_2 P_3 P_3 P_4 P_2 P_3 P_1 P_3 P_3 P_3 P_3 P_1 P_3 P_3 P_1 P_1 P_2 (C_1 C_4 C_2 C_3 C_2 C_5 C_2 C_3 C_3 C_2 P_3 C_3 C_2 C_3 C_2 C_2 C_2 C_1$

Чтение записи происходит на базе таблицы автомата A и рисунка контура. В таблице находим, что при входном воздействии P_3 на состояние C_1 автомат переходит в состояние C_4 (захват детали), затем по воздействию P_2 на C_4 он начинает движение на один шаг по прямой (состояние C_2) и приходит в точку $(1; 4)$, по воздействию P_3 на C_4 обеспечивается

переход в состояние C_3 (поворот на 90° по часовой стрелке), что соответствует ориентации робота для движения вниз по прямой и т. д. В итоге подачи всех сигналов входной последовательности робот, двигаясь по контуру, снова возвратится в точку $(0; 4)$. Его состояние C_1 во второй последовательности выделено особо (над символом C_1 нет никакого нового символа P). Это объясняется тем, что после подачи последнего входного сигнала робот будет находиться в состоянии покоя.

Естественно, что можно начать обход контура и в другую сторону (сначала идти по оси Y), но тогда изменится лишь цепочка входных воздействий. Можно пойти еще дальше и убедиться, что задача решается и для контуров другой конфигурации из избранного 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 к 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 тематической символики для записи различных формул и при доказательстве теорем. Символические записи могут читать люди или машины, которые в состоянии интерпретировать их смысл. Понятно, что последовательность символов не может быть любой, существуют ограничения на порядок записи символов в зависимости от порядка их следования.

Машины Поста и Тьюринга часто используются в различных теоретических построениях для доказательства общих утверждений, касающихся возможностей вычислительных машин при реализации алгоритмов. Однако на этих абстрактных машинах можно вести обучение основам программирования на бумаге школьников и студентов.

1.6. Машина Поста и программирование на ней

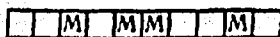
Рассматривая различные подходы к строгому определению алгоритма, мы убедились, что во всех случаях результат использования алгоритма не зависит от того, кто его применяет. Более того, действия человека по выполнению строгих однозначных предписаний накладывают действия машины. Исходя из свойств алгоритмов, легко сформулировать требования к машине:

- 1) во-первых, она должна быть полностью детерминированной и действовать в соответствии с заданной системой правил;
- 2) во-вторых, она должна допускать ввод начальных данных;
- 3) в-третьих, заданная система правил работы машины и класс решаемых задач должны быть согласованы так, чтобы всегда можно было прочесть результат работы машины.

Выдающийся американский математик Э. Пост одновременно с английским математиком А. Тьюрингом в 1936 г. предложил уточненную трактовку понятия алгоритма на основе своей машины, которую позднее стали называть машиной Поста (в отличие от Тьюринга, термин «машина» не использовал). Машина Поста менее популярна, хотя она значительно проще машины Тьюринга. С ее помощью легче решать задачу получения первых навыков составления программ для ЭВМ.

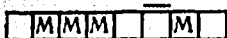
Абстрактная машина Поста состоит из бесконечной ленты, разделенной на равные секции, а также считывающей и записывающей головки.

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



Информация о заполнении метками секций ленты и пустых секций характеризует состояние ленты. Состояние ленты может меняться в процессе работы машины.

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



Говорят, что головка обзрывает эту секцию. Информация о заполнении ленты и местожении головки характеризует состояние машины Поста.

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

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

Команда машины Поста имеет следующую структуру: $x B y$, где x – порядковый номер команды, B характеризует действие, выполняемое головкой, а y указывает номер (адрес) следующей команды, которая подлежит выполнению. Машина Поста имеет шесть команд, которые представлены в таблице 1.5.1.

Таблица 1.5.1 – Система команд машины Поста

И	Вид команды	Условная запись	Описание команды
	Движение вправо	$x \rightarrow y$	Головка сдвигается на одну секцию вправо
	Движение влево	$x \leftarrow y$	Головка сдвигается на одну секцию влево
	Печатание метки	$x M y$	В секцию под головкой вписывается метка М
	Стирание метки	$x C y$	В секции под головкой стирается метка М
	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
	Стоп	$x \text{ СТОП } x$	Остановка машины

Поясним, как происходит реализация каждой команды с отображением соответствующей ситуации на ленте. Состояние головки и ленты будем показывать схематически рядом с командой 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. $x M y$

(головка печатает метку в секцию под ней);

4. $x C y$


(головка стирает метку в секции под ней);

5. $x < \begin{matrix} y_1 \\ y_2 \end{matrix}$ y_1 :

y_2 :


Если головка находится над пустой секцией, то управление передается команде 1 адресу y_1 и головка остается над этой же секцией.





Если же головка находится над секцией с меткой, то управление передается команде 1 адресу y_2 и головка остается над этой же секцией. Смысл этой команды сводится к в следующему адресу для продолжения выполнения программы.

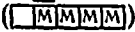
8. x стоп x 





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


Покажем реализацию некоторых типичных элементов программ для машины По иллюстрациями.

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






1 M 2 
 2 → 3 
 3 M 4 
 4 стоп 4 

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

1 C 2 
 2 → 3 
 3 C 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, головка находится над самой крайней меткой справа () . Требуется перевести головку влево до первой пустой позиции.

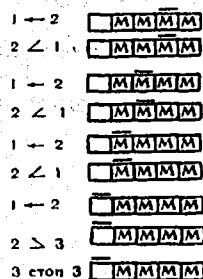
Если решать задачу без использования команды условного перехода, то программа будет следующей:

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 первой пустой секции. Программа будет иметь следующий вид:

1 ← 2
 2 < 3
 3 стоп 3

Ее уже не удастся представить в виде однозначного отображения в картинках после ко-манды. Такую программу можно «показать» только в процессе ее выполнения для ре-тного случая:



Поскольку под головкой есть метка, то управление передается снова команде 1

— «0» —

— «0» —

Поскольку под головкой находится пустая секция, то управление передается команде 3

На этом примере мы видим, что команда условного перехода является одним из средств организации типичных циклических процессов, например, нахождения первой и справа или слева от головки, расположенной над пустой секцией; нахождения слева справа от головки пустой секции, если она расположена над секцией с меткой и т. д. и средством мы будем широко пользоваться при решении различных задач на машине Поста, поскольку таким образом получают на ленте запись произвольного количества меток. грамматической машины Поста называется конечный непустой список ее команд, обладающий следующими свойствами:

- 1) на i -м месте в списке стоит команда с номером i ;
- 2) номер u совпадает с номером некоторой (другой или той же) команды списка, т. е. u обязательно упоминается среди номеров команд с левой стороны от действия B .

Примеры.

- 1) 1 → 2
2 М 3 — программа машины Поста 3 СТОП 3 (в соответствии с определением);
- 2) 2 → 3
3 М 1 — не является программой для 1 СТОП 1 машины Поста, так как нарушено первое условие;
- 3) 1 → 2 — не является программой для 2 М 5 машины Поста, так как нарушено второе условие (5 отсутствует среди левых номеров программы).
3 СТОП 3

Можно условиться, что в начальном состоянии головка всегда находится против пер-пустой секции левее последней левой метки на ленте, если не оговаривается особо, где находится. Таким образом, начальное состояние машины полностью определено состоя-нием ленты. Программа является той инструкцией, на основании которой работает машина.

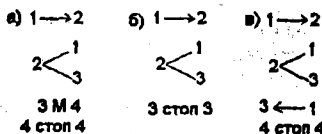
Машина приводится в начальное состояние и приступает к выполнению программы, начиная с первой команды. Каждая команда выполняется за один такт. После выполнения одной команды выполняется команда, адрес которой указан индексом u , или происходит остановка машины.

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

При выполнении программы могут встретиться три случая:

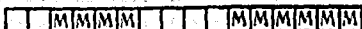
- а) машина дойдет до невыполнимой команды, и произойдет ее безрезультатная остановка;
- б) машина при выполнении программы дойдет до команды «стоп», и произойдет результативная остановка, при которой программа считается выполненной;
- в) машина при выполнении программы не дойдет ни до одной из команд, указанных а) и б), и никогда не остановится.

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



Остановимся на записи чисел на машине Поста и выполнении операций над ними.

Число k представляется на машине Поста путем записи подряд $k+1$ 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 ограничим себя простейшими из них для формирования первичных навыков составления программ.

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

Общими для реальной и абстрактной машин являются следующие моменты:

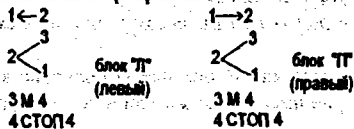
- 1) можно составлять различные программы, приводящие к одной и той же цели;
- 2) если расширяется класс исходных данных, то задача построения программы, как правило, усложняется;
- 3) при построении сложных программ можно использовать в качестве готовых строительных блоков составленные ранее программы 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, как какую-то общую ситуацию для постановки задачи, промоделировать, начиная с перкоманды, действие программы. Например, работу блока «Л» для случая, когда головка одитя над второй меткой числа, можно записать так: 1Л2 1Л2 3М4. Такая запись позво- г легко смоделировать действия машины и проверить конечный результат, глядя на ис- ную позицию головки и запись числа. В самом деле, запись 1Л2 отражает, что головка винулась влево на одну секцию и произошла передача управления команде 2; по команде тается метка М под головкой, и поэтому передается управление команде 1 (запись 21); команде 1 снова выполняется движение влево и передается управление команде 2 (вторая ь 1Л2); после второго смещения влево головка уже окажется над пустой секцией, и по- му команда 2 передаст управление команде 3 (запись 23); по команде 3 произойдет запись ки в пустую секцию и передача управления команде 4 (запись 3М4), и наконец по коман- 4 машина остановится. Такое детальное рассмотрение работы программы на достаточно их (или в крайнем случае типичных) ситуациях называется процессом отладки програм- так как он позволяет обнаруживать ошибки как программирования, так и алгоритма, по- сленного в основу программы.

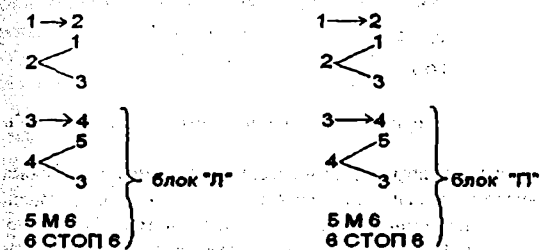
Представим себе, что в команде 2 вместо адреса 3 был бы случайно записан адрес 4, то, выполнив программу, машина остановилась бы, и к заданному числу метка не была бы исана. Конечно, такая ошибка была бы обнаружена при моделировании работы програм- на машине Поста.

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

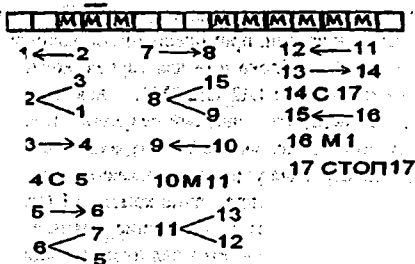
→2
←1
←3
→1
→2
→3
→4
→5
→3
5 М 6
6 СТОП 6

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

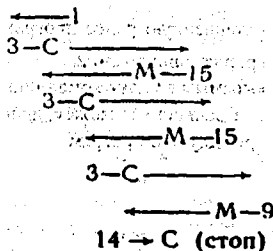
Дело в том, что написанный нами текст программы из двух команд позволяет решить за- ну нахождения первой метки числа на ленте, а дальше мы можем использовать готовый ультат, механически увеличив все номера в предыдущей программе на два:



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



Проведем отладку этой программы для сложения чисел 2 и 4, находящихся друг друга на расстоянии 3 секций, и пусть головка находится над второй секцией числа 2. Реализованная нами выше схема записи текста будет выглядеть так: 1121123П4СП6. Однако, что в этом случае трудно ориентироваться в движении головки вправо, влево, стирании написания меток. Поэтому более практична будет такая схема, когда движение головки в одну сторону отмечается стрелкой, включая записи или стирания (они помечаются буквой M или C под ячейкой ленты); пишется в отдельной строке слева направо (или справа налево) 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 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, потом стирается одна метка, и машина останавливается. Стирание метки обязательно, так как тогда бы результат был на единицу большим. 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

устройства, ПО и т. п.). В более широком смысле – это взаимосвязанная совокупность аппаратных средств вычислительной техники и программного обеспечения, предназначенная для обработки информации.

Одним из наиболее распространенных способов классификации ЭВМ является система Флинна (Flynn), в рамках которой основное внимание при анализе архитектуры ВС уделяется способам взаимодействия последовательностей (потоков) выполняемых команд и обрабатываемых данных. В результате такого подхода различают следующие основные типы систем:

– SISD (Single Instruction, Single Data) – системы, в которых существует одиночный поток команд и одиночный поток данных; к данному типу систем можно отнести обычно последовательные ЭВМ;

– SIMD (Single Instruction, Multiple Data) – системы с одиночным потоком команд и множественным потоком данных; подобный класс составляют многопроцессорные вычислительные системы, в которых в каждый момент времени может выполняться одна и та же команда для обработки нескольких информационных элементов;

– MISD (Multiple Instruction, Single Data) – системы, в которых существует множественный поток команд и одиночный поток данных; относительно данного типа систем единого мнения – ряд специалистов говорят, что примеров конкретных ЭВМ, соответствующих данному типу ВС, не существует, и введение подобного класса предпринимается в полноте системы классификации; другие же относят к данному типу, например, систолические ВС или системы с конвейерной обработкой данных;

– MIMD (Multiple Instruction, Multiple Data) – системы с множественным потоком команд и множественным потоком данных; к подобному классу систем относится большинство параллельных многопроцессорных ВС.

Принципы построения параллельных ВС. Под параллельными вычислениями понимаются процессы обработки данных, в которых одновременно могут выполняться несколько операций компьютерной системы. Достижение параллелизма возможно только при выполнении следующих требований к архитектурным принципам построения вычислительной среды:

– независимость функционирования отдельных устройств ЭВМ – это требование относится ко всем основным компонентам ВС (устройствам ввода-вывода, обрабатывающим процессорам, устройствам памяти);

– избыточность элементов ВС (использование специализированных процессоров для целочисленной и вещественной арифметики, устройств многоуровневой памяти, дублирование устройств ЭВМ путем использования нескольких однотипных обрабатывающих процессоров или нескольких устройств оперативной памяти).

Разновидности вычислительного параллелизма. Различают параллелизм уровня и структур, данных и задач.

Основная идея подхода, основанного на параллелизме данных, заключается в том, что одна операция выполняется сразу над всеми элементами массива данных. Различные фрагменты такого массива обрабатываются на векторном процессоре или на разных процессорах параллельной машины. Распределением данных между процессорами занимается ОС. Векторизация или распараллеливание в этом случае чаще всего выполняется уже на этапе компиляции – перевода исходного текста программы в машинные команды. Роль программиста в этом случае обычно сводится к заданию опций векторной или параллельной оптимизации компилятору, директив параллельной компиляции, использованию специализированных языков для параллельных вычислений.

Стиль программирования, основанный на параллелизме задач, подразумевает, что вычислительная задача разбивается на несколько относительно самостоятельных подзадач, каждый процессор загружается своей собственной подзадачей. Компьютер при этом пре-

ляет собой MIMD-машину, т. е., это компьютер, выполняющий одновременно множество различных операций над множеством, вообще говоря, различных и разнотипных данных.

Распределенные вычисления. Этот термин обычно используют для указания параллельной обработки данных, при которой используется несколько обрабатывающих устройств, достаточно удаленных друг от друга, и в которых передача данных по линиям связи водит к существенным временным задержкам и как результат, эффективная обработка данных при данном способе организации вычислений возможна только для параллельных ритмов с низкой интенсивностью потоков межпроцессорных передач данных.

Дополнительной формой обеспечения параллелизма может служить конвейерная реакция обрабатывающих устройств, при которой выполнение операций в устройствах представляется в виде исполнения последовательности составляющих операцию подкоманд. Как правило, при вычислениях на таких устройствах на разных стадиях обработки могут находиться одновременно несколько различных элементов данных.

Примером параллельных ВС могут служить кластеры — группы компьютеров, объединенных в локальную вычислительную сеть и способных работать в качестве единого вычислительного ресурса.

Основные блоки ЭВМ. ЭВМ, построенная по принципам, определенным Нейманом, состоит из следующих основных блоков (рис. 1.6.1): запоминающего устройства, арифметико-логического устройства (операционного) устройства и устройства управления.

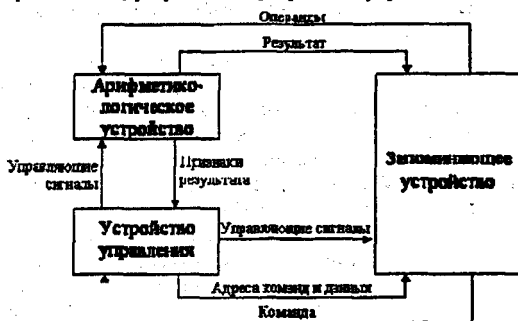


Рисунок 1.6.1 — Структура классической ЭВМ

Любое действие, выполняемое в операционном блоке, описывается некоторой микропрограммой и реализуется за один или несколько тактов. Элементарная функциональная реакция, выполняемая за один тактовый интервал и приводимая в действие управляющим сигналом, называется микрооперацией. Совокупность микроопераций, выполняемых в одном такте, называется микрокомандой. Если все такты должны иметь одну и ту же длину, а именно имеет место при работе компьютера, то она устанавливается по самой продолжительной микрооперации. Микрокоманды, предназначенные для выполнения некоторой функционально законченной последовательности действий, образуют микропрограмму. Например, микропрограмму образует набор микрокоманд для выполнения команды умножения. Устройство управления предназначено для выработки управляющих сигналов, под воздействием которых происходит преобразование информации в арифметико-логическом устройстве, а также операции по записи и чтению информации в/из запоминающего устройства. Они реализуются на жесткой логике, когда для каждой команды, задаваемой кодом операции, строится набор комбинационных схем, которые в нужных тактах вырабатывают необходимые управляющие сигналы, или микропрограммно, когда каждой команде ставится в соответствии микропрограмма.

Литература

[1, 9, 14, 20, 21, 27, 28, 32, 35, 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?
4. В чем заключается особенность комбинаторных методов реализации алгоритмов?
5. Укажите на особенности использования ассоциативных исчислений в процессах казательств и общего описания функционирования ЭВМ.
6. Чем принципиально отличается нормальный алгоритм Маркова от других алгоритмов?
7. Почему модель конечного автомата может использоваться для управления дискретными процессами?
8. По каким причинам машина Поста может рассматриваться как модель ЭВМ и может использоваться для отработки навыков по программированию?
9. Назовите основные типы архитектур вычислительных систем и их особенности.

Глава 2. ФОРМАЛЬНЫЕ ЯЗЫКИ И ОПИСАНИЕ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

2.1. Синтаксис и семантика формальных языков

Формальный и естественный языки. Интерес к применению обычного разговорного языка в диалоговом общении человека с ЭВМ и к подготовке программ объясняется целым рядом причин. Исходя из практики общения между людьми, пользователи ЭВМ, конечно, предпочитают говорить с ней на обычном языке: подавать устные команды голосом, давать текст или, в крайнем случае, делать набор этого текста на клавиатуре.

Во многих языках программирования эта тенденция проявилась в том, что в них включаются слова из естественного языка, чтобы облегчить написание программ. Однако полного отождествления этих языков с естественным не удастся достигнуть, и поэтому языкам приходится учиться.

Профессиональные программисты проходят такое обучение, но вместе с ростом возможностей машин в различных прикладных областях растет и число людей, которые хотели бы пользоваться ЭВМ в своей работе, не проникая в программистские тонкости. Сейчас резко возросло число таких программ для непрофессиональных пользователей, и делаются попытки ввести диалоговые средства для пользователя на естественном языке. Одним из простых средств такого типа является управление работой программ на уровне «меню», т. е. вина дает рецепты человеку, а он выбирает нужный из них для своего случая. Но все же и ряд непреодолимых трудностей по использованию естественных языков на ЭВМ.

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

Однако вместе с необходимостью однозначно описывать различные процессы для полной реализации на ЭВМ вскрылся ряд недостатков в построении естественных языков и пришлось прибегнуть к созданию специальных формальных языков.

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

В естественных языках различают структуру предложений (их форму) и содержание (смысл). Форме конкретного предложения не всегда соответствует единственное содержание. Например, фраза «он встретил ее на поляне с цветами» не является определенной, так как могла бы быть с цветами, поляна была с цветами или он был с цветами).

Фраза «предмет имеет синий цвет» обладает расплывчатым смыслом, так как разные люди из множества допустимых голубых и синих предметов выберут разное их количество по признаку «синий».

Грамматические правила естественного языка могут зависеть от смысла получаемых с помощью предложений. Например, правильно будет «я вижу стол» (а не стола) и «я вижу студента» (а не студент). Чтобы правильно строить такие предложения, нужно знать, 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 противоречия ни истинным, ни ложным).

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

Описание формального языка. 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 дальнейшем, чтобы избежать путаницы, мы будем четко различать метаязык (внешний язык) и язык-объект.

Завершенную конструкцию естественного языка обычно называют предложением. храним этот термин и для формальных языков. Внешний язык должен обладать возможностями для описания, как структуры, так и смысла предложений языка-объекта. Часто 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 построения 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 кодирование. Внешний язык такого формального языка состоит из двух подязыков, 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 подязыка 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 ; 10_6 ; 7_7 ; 7_8 (нижний индекс указывает систему счисления: двоичную, троичную и т. д.). Правило декодирования тогда сведется к алгоритму перевода из кодированной записи в десятичную ($111_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7$; $21_3 = 2 \cdot 3^1 + 1 \cdot 3^0 = 7$).

Металингвистические формулы Бэкуса. В большинстве описаний различных разборов 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 «ф», 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 «ф» и стоят вне угловых скобок. В левой части формулы Бэкуса должен стоять составной метасимвол, за ним следует знак «:=», после которого записывается прав-

формулы. Она может быть строкой либо пустой, либо состоящей из конечного числа символов метасимволов и символов формального языка, либо конечной последовательности таких строк, разделенных знаками «ф». По определению две формулы Бэкуса, имеющие левые и различные правые части, обозначают то же самое, что формула, которая читается, если к правой части любой из первых формул приписать символ «ф», а за ним часть другой из них.

Начало и конец формулы Бэкуса ничем не обозначены, поэтому начинать формулу надо, 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 же 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 определенных 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) (начало) (цифра), т. е. здесь указывается, что (начало) всегда будет числом, которое начинается не с нулевой цифры. Например, 2, 25 и 20 попадают 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 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 разработку смыслов в тексты и обратно.

Формальная грамматика – система правил, описывающая множество конечных последовательностей символов. Конечные последовательности символов (цепочки), входящие в данное 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, позволяющие по любой цепочке, является ли она предложением. Это деление в некоторой степени условно, так как любая распознающая грамматика, по существу, задает способ порождения 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 в теории искусственных языков, в особенности языков программирования, а также и в других случаях (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, и синтаксических правил, каждое из которых имеет вид $\phi \rightarrow \Phi$, где ϕ и Φ — цепочки, состоящие из основных и вспомогательных символов.

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

Синтаксические правила описывают связи между частями предложения. Применение правила $\phi \rightarrow \Phi$ к цепочке, имеющей вид $\alpha\phi\beta$, 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

$$V = \{a, b, c\};$$

$$W = \{A, B\};$$

$$I = A;$$

$$R = \{A \rightarrow aAB, A \rightarrow Bc, B \rightarrow b\}.$$

Одним из выводов будет последовательность: $A, aAB, aBcB, abcB$. Цепочка $abcB$ — предложение. Другой пример вывода: A, Bc, bc .

Основные классы порождающих грамматик выделяются в зависимости от ограничений, налагаемых на вид синтаксических правил.

В бесконтекстной или контекстно-свободной грамматике (КС — грамматике) используется правило вывода $A \rightarrow \Phi$, где A — вспомогательный символ и Φ — непустая цепочка. Язык порождаемый такими грамматиками, называется бесконтекстными языками.

В грамматике непосредственно составляющих (НС-грамматике) или контекстно-грамматике синтаксические правила имеют вид $\nu A w \rightarrow \nu \Phi w$. В НС-грамматике каждый шаг вывода состоит в замене одного вхождения символа A вхождением цепочки Φ , причем замена обусловлена наличием контекстов V и W . На каждом шаге вывода заменяется только один символ, поэтому с каждым выводом предложения ассоциируется так называемое дерево вывода (рис. 2.1.1), строящееся следующим образом.

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

Например, если грамматика содержит следующие правила: $I \rightarrow AAB$; $A \rightarrow a$; $B \rightarrow C$; $C \rightarrow ($ (основные символы — a, b, c ; A, B, C — вспомогательные символы, I — начальный символ), вывод $(I, AAB, aAB, aaB, aaC, aac)$ можно представить в виде иерархической структуры (дерева) так, что I будет корнем; A, A, B — тремя вершинами второго уровня; a, a, C — тремя вершинами третьего уровня; c — вершиной четвертого уровня.

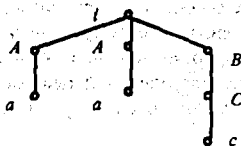


Рисунок 2.1.1 – Дерево вывода

Иногда в определение грамматики еще включают алфавит связей между буквами или ми, а также выделяют операции и формулы, которые используются в построении цепи, являющейся предложением.

В теоретических исследованиях разных авторов имеется сильное взаимное проникновение теории автоматов и математической лингвистики, одним из важных объектов которой являются порождающие грамматики. В некотором смысле к классу порождающих автоматов можно отнести и любой вычислитель (человека, машину), который в процессе вычисления выдает цифры или другие знаки; эти знаки можно рассматривать как элементы, которые он складывает в процессе вычисления.

Математический вывод всегда осуществляется средствами заданной формальной системы. Сначала происходит кодирование входной информации в языке данной формальной системы (или машины), а затем эти коды перерабатываются в соответствии с правилами кодирования рассматриваемой системы (или машины). Процессы вывода в порождающих грамматиках аналогичны преобразованиям слов в ассоциативных исчислениях.

2.2. Алгоритмические языки и операционные системы

Общие подходы к построению алгоритмических языков и трансляторов. Алгоритмический язык является средством точного формулирования вычислительных процессов для 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, программирование 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. Данный этап связан, прежде всего, с анализом классов задач, подлежащих решению. Основная цель анализа – выделить в рассматриваемых задачах устойчивые и наиболее часто встречающиеся структуры объектов языка и выявить над ними, которые после их формализации и введения соответствующей символики являются элементарными объектами и базисными операциями языка.

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

В языке должны быть правила, позволяющие из элементарных операций и объектов кономерно строить более сложные. Для операций таковыми являются в основном при образовании линейных структур с использованием разного рода скобок и введением функциональных обозначений.

Алгоритмы численного анализа находятся в лучшем положении, так как для них существовала сложившаяся веками математическая символика. Кроме того, применен ЭВМ началось с автоматизации арифметических вычислений. Поэтому для них были созданы первые в мире системы автоматизации программирования и первый алгоритмический язык, ставший международным стандартом, — АЛГОЛ-60.

Базисными операциями в языках для алгоритмов численного анализа являются арифметические действия над числами и их отношения, а также элементарные функции математического анализа. Элементарными объектами являются числа и числовые переменные, которые могут объединяться в линейные массивы (векторы, матрицы) с указанием компонент массива при помощи индексов-координат.

Основными средствами описания сложных структур в языках являются: объединение объектов в линейные последовательности, образование скобочных структур и функциональные обозначения. При этом большое место занимают операции поиска и обработки данных, также описания формата и структуры входной информации.

Проблема обработки символической информации возникла только с появлением вычислительных машин. Поэтому авторы первых алгоритмических языков из рассматриваемого класса вводили тот или иной набор операций и объектов как продукт некоторого умозрительного и приорного представления о закономерностях обработки символической информации.

Отбор изобразительных средств языка — противоречивый и сложный процесс, который требует учета последующих стадий разработки системы программирования. Состав и грамматический строй алгоритмического языка всегда являются компромиссом между стремлением к богатству и разнообразию операций и объектов и стремлением к его простоте и лаконичности, которые позволяют обеспечить создание транслятора и упростить его работу.

Описание языка. На этом этапе предполагается, что любой язык представляет собой некоторое множество осмысленных текстов. Описать язык — значит задать правило, которое определяет, какие тексты могут быть текстами языка (синтаксис), и установить, какие процессы и 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 структурирована 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, определять 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, и 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 рамках или в элементах программ (в выражениях, операторах и пр.), обусловленные незнанием программы или элементов программы металингвистическим формулам, называясь синтаксическими ошибками. Программы или какие-либо элементы программы, не являющиеся синтаксическими ошибками, называются синтаксически правильными. Синтаксическая правильность программы не означает, что в программе вообще нет ошибок. Например, с помощью синтаксического контроля нельзя обнаружить ошибку, допущенную автором, синтаксически фразы не противоречат правилам ее написания. Это можно сравнить с таким примером, когда перестановка запятой на другое место решает судьбу человека: «Любить нельзя повесить», хотя автору текста хотелось отдать другой приказ: «Помогать нельзя повесить».

Разработка транслятора. Трансляторы с алгоритмического языка, представляют собой 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 операций 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 снижением производственных задач непрерывно совершенствуются. Часть из них отмирают, рождаются новые языки. Например, причиной рождения новых языков послужило появление многопроцессорных машин, позволяющих одновременно выполнять независимые цели 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 средств трансляции, а также методики их использования.

Алгоритмы, описанные на различных формальных языках, как уже отмечалось выше, могут непосредственно выполняться на цифровой электронной вычислительной машине, как они, как правило, не представлены на языке ее команд. Языки описания алгоритмов часто 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 с ее помощью человеком из-за структурированности программы на уровне крупных блоков 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) 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 программы и более развитую структурную схему для диагностики ошибок синтаксического типа в алгоритме 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 с отклонениями от правил их записи); анализ описания данных и распределение памяти для них для программы; получение и оптимизация текста программы, выдача текста оттранслированной программы. Часть из этих функций в некоторых трансляторах может отсутствовать, например оптимизация.

По характеру работы близки к трансляторам и различные конверторы и эмуляторы. Их можно рассматривать как трансляторы с более узким выполнением функций. Программы эмуляции используются для интерпретации команд одной машины на другой. К их услугам прибегают при моделировании работы проектируемых ЭВМ или же при необходимости обеспечить возможность выполнения программ для ЭВМ старых моделей на новых ЭВМ. Получение текстов программ для их реализации на конкретной ЭВМ выполняется с помощью операционной системы.

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

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

Особая роль среди управляющих программ отводится супервизору, который всегда находится в оперативной памяти. Супервизор ведет текущее обеспечение выполняемых шагов и, ведает такими ресурсами машины, как память, управляет процессами обмена с иными устройствами.

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

Управление данными представляет собой обобщение понятия системы управления вводом, которая организует ввод-вывод при выполнении ВС всех возможных заданий. При возникновении сбоев и неисправностей машины сообщается оператору, и он принимает соответствующие меры по устранению неисправностей или же продолжает решение оставшейся задачи, 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 машины или же на значительном удалении от нее. В этом случае каждый пользователь получает определенное время, 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. Удачным подбором числа дисплеев и задач, согласующихся с быстродействием процессора, для пользователя создается «иллюзия», что он один работает за пультом машины. Схемы организации такой работы могут идти и на приоритетной основе, когда для более важных пользователей кванты времени выделяются чаще, или же кванты времени имеют большую длительность. Еще сложнее обеспечить работу программ, которые используются для управления нерезультативными процессами в реальном масштабе времени. Когда результат их работы должен быть известен не позже наперед заданного момента времени.

ОС с элементами искусственного интеллекта. Кроме того, ОС для конкретной ЭВМ в составе 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 являются фактически довольно сложной автоматизированной системой специального назначения с рядом программ, имитирующих деятельность человека по эксплуатации и отладке программ. Создание таких ОС с элементами искусственного интеллекта за относительно короткий срок с момента появления машин, которые обслуживают одновременно многих пользователей, объясняется тем, что их создатели-программисты должны были оптимизировать свой собственный труд и им не требовалось привлекать знания из других областей деятельности человека.

Вместе с тем сложность современных ОС на больших ЭВМ возросла до такой степени, что пользоваться всеми возможностями, которые они могут предоставить, не всегда в состоянии даже опытный программист. Поэтому операционные системы для персональных и больших ЭВМ для пользователей-непрофессионалов, в программировании стали включать элементы «дружественного» интерфейса, направленного на облегчение диалога человека и машины. ЭВМ взяла на себя функции по обучению и консультированию пользо-

вателя при решении им конкретных задач: на базе тестовых примеров советовать пользе, какой алгоритмический язык ему выбрать; на какие конструкции операторов и специальных средств следует ориентироваться; какими подсказками ЭВМ воспользоваться на ограниченном выборе действий («меню») и т. д.

Операционные системы с каждым днем продолжают совершенствоваться, и степень доступности для пользователей будет упрощаться. 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 программистами-профессионалами стала черновой и неинтересной, так как она была направлена лишь на облегчение освоения ОС пользователями 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 сводится к управлению движением маркера на экране дисплея в нужных областях, к указанию «световым» пером нужного объекта или его элемента, звуковому вводу информации на новое ограниченное 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 отмечал академик А.И. 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.

2.3. Механизмы поиска информации

Информационно-логические задачи. 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.

вую очередь такого рода задачи используются при применении различных автоматических систем обработки информации, при поиске конструкторско-технологической информации в системах автоматизированного проектирования, в оперативном управлении предприятием.

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

Типы систем поиска информации. Фактографические системы поиска информации предлагают 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 образа ключевых слов, выбранных из ряда характерных для данной тематики текстов, является стандартный набор терминов без синонимов со строго фиксированными значениями. Эти термины называются дескрипторами, а полный набор таких терминов называется набором 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 комбинации. 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 перечисления 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 проектирования экономических задач в заданном массиве документов:

Таблица 2.4.1 – Инверсный массив

Дескрипторы	Коды	Номера документов
Наименование		
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, незанятые 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, делом финансовые операции и запирающемся на электронный замок, а также в тексте электронном банке данных научной информации.

Более эффективными являются дескрипторные поисковые системы с грамматикой, в которых дескрипторы, относящиеся к документам, и дескрипторы, входящие в состав запроса, 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 данных» 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 машинно утверждений и дает 3 ответа: «да», «нет», «неизвестно».

Специальные информационные языки для поиска информации. В качестве иллюстрации одного из возможных специальных информационных языков для поиска конструкторской технологической информации на основе порождающей грамматики Γ приведем пример.

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

$\Gamma = \langle V, W, I, R \rangle$,

где V – основной словарь,

W – вспомогательный словарь, $I (I \in W)$ – начальный символ,

R – множество правил подстановки, которые представим в форме Бэкуса с использованием знаков \wedge (и), \vee (или), \neg (нет):

$\langle \text{описание объекта на информационном языке} \rangle ::= \langle \text{логическое произведение сообщений} \rangle$

<логическое произведение сообщений> ::= <сообщение> | <сообщение> Λ <логическое
 зведение 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
 ювой характеристики>
 <знак отношения> ::= > | < | >= | <= | =
 <значение числовой характеристики> ::= <положительное число> | <отрицательное
 ю>

Язык, порождаемый грамматикой Г, будем называть информационным языком, а эле-
 ги множества W – сообщениями на информационном языке. Для осуществления поиска
 ных сведений в этом случае сообщения на языке взаимодействия переводятся на инфор-
 онный язык (кодирование) и после получения ответа на информационном языке прово-
 его декодирование. Сложность поиска для пользователя в этом случае сводится к уме-
 описать требуемый объект, используя его характерные признаки в их логической взаи-
 мизи.

Из известных разработок представляет значительный интерес система фактографиче-
 о типа на ОАО «Пластполимер». В ней хранятся сведения практически обо всех термо-
 тичных фторполимерах. Проводя диалог с ЭВМ, можно получить необходимые сведе-
 о любом из девяти тысяч марок полимерных материалов и их известных свойствах. Ана-
 чная информация имеется в ЭВМ и о свойствах деталей, выпускаемых из полимеров.

Особенность автоматизированной системы состоит в том, что она не только дает поло-
 ьный или отрицательный ответ, но, учитывая предъявляемые к материалу требования,
 мендует, какой полимер выбрать, а в случае отсутствия возможности дать точный ответ,
 алагает рекомендацию о материалах с близкими свойствами по отношению к заданным.

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

В настоящее время страны выработали международные коммуникативные форматы,
 оляющие передавать информацию по каналам связи. Например, информацией на ма-
 ных носителях в области изобретательской деятельности сейчас обменивается ряд стран.
 логичные обмены ведутся и в области реферативной информации по публикуемым ис-
 ишкам.

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

В технических областях знаний запись информации на практически однозначно вос-
 нимаемых машиной подмножествах естественного языка позволяет уже сегодня полу-
 машинные переводы, пригодные для понимания затрагиваемого вопроса специалистом.
 ому информационная система, снабженная соответствующими программами для пере-
 с наиболее употребительных языков, позволяет специалистам, минуя языковой барьер,
 омититься с последней информацией в мире в конкретных областях деятельности.

2.4. Базы данных и знаний. СУБД и языки запросов

Назначение баз данных и знаний. Большинство сложных производственных автоматизированных систем различного назначения требует не только использования специфических процессов для подготовки исходных данных для решения отдельных задач, но и объединения данных с выполняемыми одновременно другими задачами, а при создании интеллектуальных систем и подбора методов для разрешения создавшихся ситуаций. Это привело к созданию систем, близких по духу к операционным, но имеющих 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 баз 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 баз 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. Для баз данных реляционного типа массовое применение получил язык запросов *SQL*.

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 интеллектуального интеллекта, в частности, при создании экспертных систем, т. к. они позволяют получать знания, которые непосредственно не вносятся 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.

Банк данных. 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 фундаментальных концепциях банков данных, позволяют обеспечить многоаспектный доступ к совокупности взаимосвязанных данных, интеграцию и централизацию управления данными, устранение 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 поддерживает 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

му значению в соответствующем столбце. Столбец соответствует некоторому элементу μ – атрибуту. Каждый столбец имеет имя соответствующего элемента данных (атрибу-
Длин или несколько атрибутов, значения которых однозначно идентифицируют строку
таблицы, называют ключом таблицы. Он и является исходным элементом различных опера-
с объектами и для организации получения информации из других таблиц. Табличная
работы с информацией широко используется в различных сферах деятельности; и по-
у такие банки данных облегчают их применение непрофессионалами.

К схемам такого рода относятся и банки данных, позволяющие осуществлять в тек-
ых документах поиск необходимой информации на основе логических конструкций, по-
нных пользователем с применением ключевых слов.

Особую роль интеллектуальной части СУБД играют функции интеграции текстовых,
рых и графических данных и в последние годы и речевых сообщений, особенно в диа-
ных процедурах. К элементам интеллектуализации можно также отнести и протоколы
на документами при создании различных систем, когда при установлении соединения
ентов проверяются их права на доступ к ресурсам и выполнение контроля защиты баз
ых от разрушения, для регистрации пользователей на основе средств искусственного
лекта.

При автоматизации проектирования и принятия решений в различных областях дея-
ности полезно в БД иметь особый подраздел в виде архива готовых прототипов (проек-
ешений). Однако это связано с рядом особенностей введения этого подраздела, так как
ходим экспертный отбор элементов архива, их заверения электронной цифровой подпи-
с выполнением аналогичных условий и при изъятии элементов архива.

Специфика использования знаний в интеллектуальных системах. При обработке
ВМ данные постоянно трансформируются, последовательно проходя от входной инфор-
ри как результата измерений и наблюдений, к данным на материальных носителях ин-
ации в виде таблиц, протоколов, справочников; структур данных в виде диаграмм, гра-
ов, функций; в компьютере на языках описания данных и т. д. Более высокой формой ор-
зации данных является их представление в виде баз знаний.

Знания связаны с данными, основываются на них, но представляют собой результат
пительной деятельности человека, обобщают его опыт, полученный в ходе практической
ельности. Знания – это выявленные закономерности предметной области. При обработке
ВМ знания также трансформируются: от существующих форм в памяти человека как ре-
гат обучения, воспитания, мышления к знаниям, помещенным на материальных носите-
учебниках, инструкциях, методических пособиях, книгах и далее к знаниям, описан-
на языках их представления для занесения в компьютерные базы знаний.

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

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

Исторически первыми были процедурные знания, т. е. знания, представленные в алго-
ах.

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 яв-
я результатом интеллектуальной деятельности древних геометров и представляет собой
и, выражающий площадь круга через его радиус.

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

Производственная модель состоит из трех основных компонентов. Первый из них – 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 задачи 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 пертной 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, хранящимися 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 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, содержимое рабочей памяти пополняется образцом «использовать джип». В итоге правила, которые можно было бы применить, не остается и система останавливается.

В рассмотренном примере приведен прямой вывод – от данных к поиску цели. Однако применяются и обратный вывод – от цели для ее подтверждения к данным. Продемонстрируем этот способ на нашем примере. Допустим, что наряду с исходными данными «намерение – 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 специального 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, 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 потребуются 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 множество носит название *конфликтного набора*, а выбор одного из них называется *решением 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 кроме того, не существует правила, подтверждающего его, данный выбор является немым. 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.

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

Следующей эффективной моделью представления знаний является фреймовая модель. Фреймовая модель использовалась первоначально в психологии и философии для описания содержания абстрактного образа. Например, слово «автомобиль» вызывает у слушающих образ объекта, способного перемещаться, имеющего четыре колеса, салон для шофера и пассажиров, двигатель, руль. Приведенное описание абстрактного образа «автомобиль» является минимальным и из него ничего нельзя убрать без потери его сущности.

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

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

В качестве значения слота может выступать имя другого фрейма. Таким образом фреймы объединяются в сеть. Свойства фреймов наследуются сверху вниз, т. е. от вышестоящих фреймов к нижестоящим через АКО-связи (начальные буквы английских слов «A Kind Of», что можно перевести как «это»). Слот с именем АКО указывает на имя фрейма более 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?» – ответ будет положительным, так как этим свойством обладает вышестоящий фрейм «Человек».

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

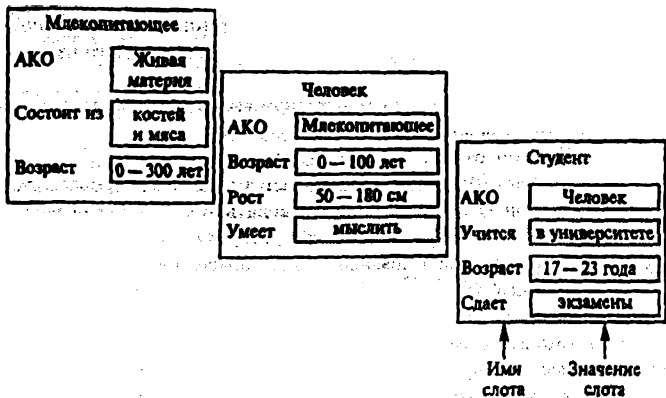


Рисунок 2.5.1 – Сеть фреймов

Основным преимуществом фреймов как способа представления знаний является наглядность 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, что любое знание можно представить в виде совокупности *понятий* (объектов) и *отношений* (связей). Семантическая сеть представляет собой ориентированный граф, вершинами которого являются понятия, а дугами – отношения между ними. Сам термин «семантическая» означает смысловая.

Пример семантической сети приведен на рис. 2.5.2.

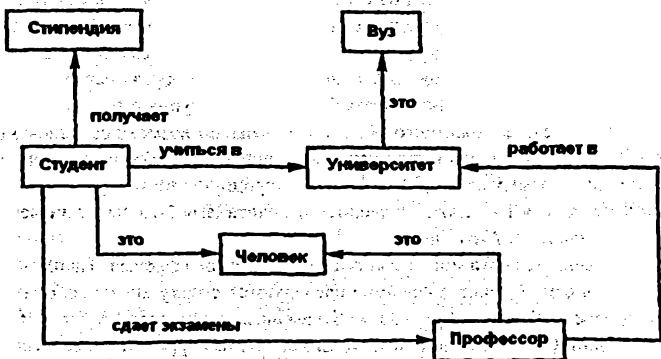


Рисунок 2.5.2 – Семантическая сеть

Основным преимуществом этой модели является наглядность представления знаний, также соответствие современным представлениям об организации долговременной памяти человека. Недостаток – сложность поиска вывода, а также сложность корректировки, т.е. удаления и дополнения сети новыми знаниями.

Литература

[1, 9, 17, 18, 21, 23, 27, 28, 29, 33, 34]

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 алгоритмического кода 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. ПРОБЛЕМЫ СОЗДАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

3.1. Общие подходы к созданию систем искусственного интеллекта

В течение многих столетий человек в основном занимался проблемами автоматизации физического труда. Появление современной вычислительной техники, важнейшей компонентой 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 ряд обнадеживающих результатов прикладного плана при попытках моделирования творческой деятельности человека по управлению различными объектами, проскрининг распознаванию образов, доказательству теорем, игре в шахматы и шашки, сочинению стихов, музыки и т. п.

Развитие робототехники в последние годы породило еще ряд сложных проблем, связанных с организацией поведения роботов. Они не возникали, 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, который развил 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 на вопрос о возможности машинного мышления будет отрицательным. Если же стать на позицию оценки «деятельности» машины по конечным результатам, то ответ будет положительным, что хорошо иллюстрируется следующим опытом (рис. 3.1.1).

Имеется три комнаты. В первой комнате находится человек (экспериментатор), во второй — машина, в третьей — человек. Экспериментатор точно не знает, в какой комнате находится человек, а в какой — машина. Ему предлагается задать конечную серию задач

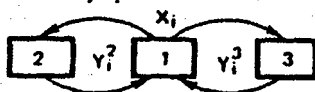


Рисунок 3.1.1 — Схема эксперимента Тьюринга

(вопросов) и определить по ответам, в какой комнате принимаются решения человеком, а в какой — машиной. Естественно признать, что машина может «мыслить», если экспериментатор однозначно не может указать, где находится машина. Простейшими примерами для эксперимента могут послужить решение шахматных этюдов или задач, проектирование режущего инструмента и т. д.

3.2. Попытки создания общих решателей задач

Все пять видов процедур в той или иной степени были использованы впервые в США в годах А. Ньюэллом, Дж. Шоу и Г. Саймоном в программе «Общий решатель задач», назначенной для решения широкого класса задач (доказательство теорем, интегрирование с использованием табличных интегралов, игровые задачи и др.). Они осознали необходимость развития новой теории, опирающейся на имитацию процедур универсального характера, с помощью которых порождались бы конкретные процедуры, направленные на решение определенных задач — задача интеллектуального типа.

В качестве основной ими была взята процедура, имитирующая поиск по лабиринту ожных альтернатив. Авторы программы выдвинули тезис, что решение любой задачи веком состоит в поиске пути, приводящего от начальной площадки лабиринта, соответствующей исходному описанию ситуации, к некоторой конечной площадке, соответствующей решению этой задачи. На каждом шаге поиска пути используются локальные критерии, дающие возможность выбора очередного коридора лабиринта. Авторы вначале были уверены, что с помощью найденной ими процедуры удастся решить все интеллектуальные задачи. Первыми успешными экспериментами в этом направлении были доказательство теоремы исчисления высказываний и игровые программы. Но проверка универсальности процедуры поиска по лабиринту на шахматной программе показала ее авторам, что, основываясь на ней, практически невозможно решить сколько-нибудь серьезную задачу.

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

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

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

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

Дискуссия ученых о том, какие системы считать интеллектуальными, не привела кированию единой точки зрения. Ряд ученых уходит от прямого определения и относит кым системы, играющие в шахматы и шашки, сочиняющие стихи и музыку, распознаю-речь и образы, выполняющие автоматический перевод, доказывающие теоремы и др. ко все существующие конструктивные определения распадается на две группы: в пер-группе перечисляются свойства интеллектуальной системы, а во второй — минимальный ень выполнения некоторого списка функций.

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

Не останавливаясь детально на философских аспектах этой проблемы, будем считать, процесс получения любого результата всегда расчленяется на несколько этапов, каждый торых может полностью или частично выполняться вычислительной машинкой. Если гичные результаты, полученные человеком на каком-то этапе или на нескольких авто-вированных этапах, квалифицируются как творческие, то будем и машинный результат сить к разряду творческих. При невозможности автоматизировать весь процесс, можно

остановиться на автоматизации лишь его рутинных этапов. Однако во всех случаях машина будет получать либо творческий результат, либо использование машины поможет человеку прийти к этому результату. На наш взгляд, в практике не существует резкой границы 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 приемлемым, так как многие процессы могут выполняться программно и аппаратно. 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 человеком и машиной. От активного вмешательства человека в процесс выполнения машинной 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, 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.

3.3. Диалоговые методы решения задач 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 (дисплеи, клавиатуру, звуковой ввод информации и т. п.) и специальное ПО, 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 нелегко 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.

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. С одной стороны, они стремятся детально изучить функции и свойства элементарной ячейки нервной системы — нейрона, а с другой — научиться «собирать» из этих ячеек сложные конструкции структуры человеческого мозга. На этом пути пока не удалось достигнуть существенных в прикладном отношении результатов, так как имеются трудности, как с описанием самой элементарной ячейки, так и с поиском закономерностей объединения их в сети. Имитировать же работу мозга с помощью случайно скоммутированной сети маловероятно. Другой путь — с моделированием работы мозга на уровне информационных процессов, добиваясь, чтобы функции модели были по возможности неотличимы от функции моделируемого объекта.

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

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

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

В целом ситуационное управление дает с точки зрения языка описания единообразный подход как для построения модели объекта, так и для управления им. Однако решение задач этим методом из-за отсутствия специального математического обеспечения достаточно трудно, а качество получаемых решений определяется квалификацией специалистов, привлекаемых для формирования моделей объекта и управления им.

Мы подробнее остановимся на той ветви второго направления работ по искусственному интеллекту, которое широко известно как *эвристическое программирование*, и остановимся на его прикладных аспектах, так как в недрах эвристического программирования зародились идеи создания универсальной программы для решения любых задач.

По вопросам создания искусственного интеллекта насчитывается сотни тысяч публикаций. Тем не менее, при решении конкретных прикладных проблем не всегда удается эффективно использовать эту информацию. Обычно статьи содержат общие рекомендации по проектированию программ и не имеют описаний конкретных приемов, использованных данным автором.

Современное эвристическое программирование занимается исследованием типовых приемов, полезных в процессах решения проблем. Эвристика строится на основе наблюдения за тем, как люди решают задачи, что есть общего в решении самых различных проблем.

До начала решения задачи надо быть уверенным, что она поставлена. В современной практике рассматриваются лишь некоторые методы решения задач, но не исследуются вообще, связанные с постановкой проблемы. Методы эвристического программирования в основном направлены на ограничение области поиска решения за счет использования опыта решения данного класса задач или выдвижения некоторых гипотез.

Прежде чем приступить к решению, надо обладать *методом распознавания* удовлетворительного ответа. Иными словами, за конечное число шагов требуется установить, является ли полученный ответ удовлетворительным. С этой точки зрения, крайне желательно иметь активный алгоритм, гарантирующий получение решения, если оно существует в пределах разумных затрат времени.

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

Процедура распознавания образов позволяет не исследовать все возможности для организации поиска, чтобы сразу применить к конкретной проблеме наиболее эффективный метод. В связи с этим возникает задача классификации методов и распознавания соответствующих им ситуаций. Наиболее простой путь в этом отношении – сравнение неизвестного объекта с рядом эталонов. На практике оказывается, что число эталонов довольно велико. Поэтому пытаются истолковать образ как систему в некотором смысле подобных объектов. Возникает задача выявления хорошей системы признаков, описывающих образы.

При наличии такой системы подпрограмма распознавания подвергает исходные данные последовательным испытаниям на присутствие различных признаков. После чего решает вопрос о принадлежности исследуемого объекта к одному из образов в соответствии с выявленными признаками. Поэтому основной задачей многих эвристических программ является отыскание хорошей системы признаков, инвариантных относительно различных преобразований.

Обучение позволяет при решении новых задач использовать методы, оказавшиеся эффективными при решении аналогичных проблем. Такую эвристику обычно реализуют на основе моделей, повышающих ее приоритет при успешном применении. Любая обучающая система должна использовать результаты прошлого опыта как основу для более общих предположений. Простейшим способом обобщения внутри множества признаков является построение «типичного» члена данного множества (т. е. усреднение). Все же возможности системы с простым повышением приоритета ограничены ее зависимостью от «учителя» (тренировочной последовательности задач). Один из путей преодоления этой трудности состоит в разработке процедуры обобщения действий «учителя». Тогда при решении задач машина будет сама повышать приоритет эвристик в процессе работы.

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

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

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

исту или же к специализированной автоматизированной «экспертной» системе. Решение задач в реальном масштабе времени должно опираться на эффективные средства связи людей, так и ЭВМ.

Диалоговый режим открывает новые возможности и в осуществлении моделирования (испытательного эксперимента). Некоторый объект можно представить в виде ряда элементарных формальное описание которых позволяет имитировать их поведение на ЭВМ. В результате имитации поведения среды можно проверить качество будущей конструкции без ее физической реализации и тем самым экономить средства, затрачиваемые на неудачные экспериментальные образцы. Наличие моделей для различных элементов позволяет инженеру в диалоговом режиме путем подбора построить модель объекта из нужного набора элементов, дающую искомыми свойствами.

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

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

Использование при решении задач проектирования больших банков данных, в которых используются типовые решения и элементы, поможет обеспечить получение новых конструкций и элементов, а также быстро учитывать в создаваемых конструкциях новинки, разработанные в других организациях и записанные в общий банк данных.

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

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

Возможности машинного эксперимента по сравнению с натурным очень большие. Он дешевле, позволяет воспроизвести условия, которые трудно создать в лаборатории, и дает полную информацию о процессах, протекающих в модели.

Например, проблемы загрязнения среды, развития животного и растительного мира, изменения и обводнения, развития городов и поселков разумно решать с участием многих специалистов и путем широкого машинного эксперимента.

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

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

Взаимодействие процессов, протекающих в сетях из разнородных ЭВМ; кроме обмена информацией, порождает еще массу проблем, связанных с достоверностью переданных и их защитой от постороннего вмешательства.

Для рядового пользователя ЭВМ — специалиста только в своей области — ступок во этих специфических вопросах постепенно стал непреодолимой стеной на пути использования вычислительной техники.

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

ЭВМ, соединенные в сети и хранящие информацию и тексты различных программ в улах сети, придают новое качество функциональным возможностям специалиста — оперативность в решении сложных задач. Он не заменяется техникой, а лишь его усилия перемещаются от рутинной работы в более творческие области благодаря первичной обработке и пересылке информации без его участия.

На схему работы человека существенно должно повлиять внедрение персональных микро-ЭВМ, которые допускают как автономное решение задач, так и использование их компонентов в локальных сетях ЭВМ. Такие ЭВМ используются в автоматизации подготовки данных и как «интеллектуальные» терминалы для пересылки информации, и как средства обращения за помощью в решении задач к другим ЭВМ сети.

Такие сети ЭВМ могут использоваться для безбумажной обработки информации благодаря простым средствам обмена ею в «электронной» форме между людьми, обладающими персональными ЭВМ. Более того, развиваются идеи о проведении телеконференций на основе сетей ЭВМ и о перенесении рабочих мест сотрудников из учреждений в домашние условия. Таким образом, диалоговые методы решения задач по мере развития научно-технического прогресса получают новые эффективные возможности для обработки, хранения и обмена информацией. В частности, хорошие результаты в ряде специальных областей деятельности человека получили с использованием экспертных систем, построенных с использованием знаний высококвалифицированных специалистов.

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

Существуют неформализуемые знания, которые вообще не попадают в книги и руководства в связи с их неконкретностью, субъективностью, приблизительностью. Знания этого рода являются результатом многолетних наблюдений, опыта работы, интуиции. Они обычно представляют собой множество эмпирических и эвристических приемов и правил. Такие знания передаются из поколения в поколение в виде определенных навыков, ноу-хау, секретов ремесла. Есть также знания, которые не могут быть выражены ни в математическом виде, ни в терминах обычного человеческого языка.

Класс задач, относящихся к неформализуемым и плохо формализуемым знаниям, значительно больше класса задач, для которых знания формализуемы. Этим объясняется особая популярность и широкое практическое применение экспертных систем, которые открывают 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 знания: управления производством, медицине и т. д.

Экспертная система должна содержать существенную часть знаний эксперта-человека конкретной области и использовать накопленную информацию подобно человеку. Напри

экспертная система в области медицины должна уметь ставить диагноз на основе анализов симптомов пациента. Из такого класса программ наиболее интересна в прикладном плане система МИЦИН, созданная в Станфордском университете в США, и ее последующая версия ГЕЙРЕСИАС. Их успех у практикующих медиков в том, что они помогают врачу поставить правильный диагноз на основании рассуждений типа: «Если..., то можно предположить, ...». Программа дает врачу численную оценку вероятности каждого из заключений. За словом «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 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 записей, 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, опираясь глубже на основе точных анализов сложившуюся ситуацию.

Рассмотрим одну из возможных структур экспертной системы (рис. 3.3.1).

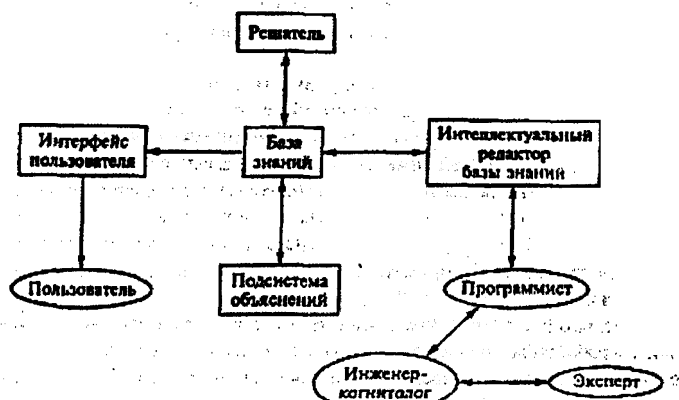


Рисунок 3.3.1 — Типичная блок-схема экспертной системы

Обычно в ее состав входят следующие взаимосвязанные между собой модули:

- база знаний — ядро экспертной системы, совокупность знаний предметной области, хранящаяся 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, подсказочный 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, реализующая 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.

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, извлекать 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 задачи 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 надежности, важности, четкости. В этом случае они снабжаются некоторыми весовыми коэффициентами, которые называют коэффициентами доверия. Такие знания обрабатываются с помощью алгоритмов нечеткой математики.

В процессе опытной эксплуатации коэффициенты доверия могут подвергаться корректировке. В этом случае говорят, что происходит обучение экспертной системы. Процесс обучения экспертной системы может производиться автоматически с помощью обучающего алгоритма либо путем вмешательства инженера-когнитолога, выполняющего роль учителя.

В процессе разработки экспертные системы проходят определенные стадии, в результате которых создаются различные версии, называемые прототипами.

Демонстрационный прототип — экспертная система, которая решает часть требуемых задач, демонстрируя жизнеспособность метода инженерии знаний. Работает, имея в базе знаний всего 50-100 правил.

Исследовательский прототип — экспертная система, которая решает все требуемые задачи, но неустойчива в работе и не полностью проверена. База знаний содержит 200-500 правил.

Действующий прототип — надежно решает все задачи, но для решения сложных задач может потребоваться много времени и памяти. База знаний содержит 500-1000 правил.

Промышленная экспертная система — обеспечивает высокое качество решения всех задач при минимуме времени и памяти, что достигается переписыванием программ с использованием более совершенных инструментальных средств и языков низкого уровня. База знаний содержит 1000-1500 правил.

Коммерческая экспертная система — отличается от промышленной тем, что помимо собственного использования она может продаваться различным потребителям. База знаний содержит 1500-3000 правил.

В настоящее время уже сложилась определенная технология разработки экспертных систем (рис. 3.3.2):

1. *Идентификация (постановка задачи).* На этапе устанавливаются задачи, которые подлежат решению, выявляются цели разработки, требования к экспертной системе, ресурсы, используемые понятия и их взаимосвязи, определяются методы решения задач. Цель этапа — сформулировать задачу, охарактеризовать поддерживающую ее базу знаний и таким образом обеспечить начальный импульс для развития базы знаний.

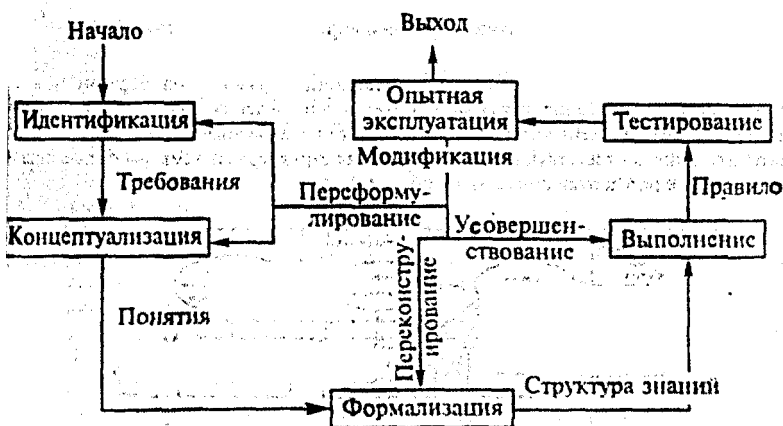


Рисунок 3.3.2 – Технология разработки экспертной системы

2. **Концептуализация.** Проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

3. **Формализация.** Определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, оценивается адекватность целям системы зафиксированных понятий, методов решения, средств представления и манипулирования знаниями.

4. **Выполнение.** Осуществляется наполнение экспертом базы знаний. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечение эффективную работу системы, и представление знаний в виде, понятном эксперт-системе. Из-за эвристического характера знаний их приобретение является весьма трудным.

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

6. **Опытная эксплуатация.** Проверяется пригодность экспертной системы для конечных пользователей. По результатам этого этапа может потребоваться модификация эксперт-системы.

7. **Модификация.** В ходе создания экспертной системы почти постоянно производится модификация: переформулирование понятий и требований, переконструирование представления знаний и усовершенствование прототипа.

Усовершенствование прототипа осуществляется в процессе циклического прохождения этапов выполнения и тестирования для отладки правил и процедур вывода.

Переконструирование выбранного ранее способа представления знаний предполагает с этапа тестирования на этап формализации.

Если возникшие проблемы еще более серьезны, то после неудачи на этапе тестирования может потребоваться возврат на этап концептуализации и идентификация. В этом случае идет о переформулировании понятий, используемых в системе, т. е. перепроектировании системы заново.

3.4. Распознавание образов и обработка изображений

Общая характеристика задач распознавания образов. Под образом понимается структурированное описание изучаемого объекта или явления, представленное вектор признаков, каждый элемент которого представляет числовое значение одного из признаков характеризующих соответствующий объект. Общая структура системы распознавания и этапы в процессе ее разработки показаны на рис. 3.4.1.

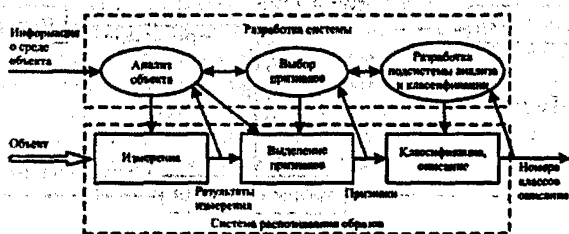


Рисунок 3.4.1 – Структура и этапы разработки системы распознавания

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

Задачи распознавания имеют следующие характерные черты.

1. Решение, как правило, состоит из двух этапов: а) приведение исходных данных к виду, удобному для распознавания; б) собственно распознавание (указание принадлежности объекта определенному классу).

2. Можно вводить понятие аналогии или подобия объектов и формулировать понятие близости объектов в качестве основания для зачисления объектов в один и тот же класс и разные классы.

3. Можно оперировать набором прецедентов-примеров, классификация которых известна и которые в виде формализованных описаний могут быть предъявлены алгоритму распознавания для настройки на задачу в процессе обучения.

4. Трудно строить формальные теории и применять классические математические методы (часто недоступна информация для точной математической модели или выигрыш от использования модели и математических методов не соизмерим с затратами).

5. В этих задачах возможна «плохая» информация (информация с пропусками, разнородная, косвенная, нечеткая, неоднозначная, вероятностная).

Целесообразно выделить следующие типы задач распознавания.

1. Задача распознавания – отнесение предъявленного объекта по его описанию к одному из заданных классов (обучение с учителем).

2. Задача автоматической классификации – разбиение множества объектов (ситуаций по их описаниям на систему непересекающихся классов (таксономия, кластерный анализ, обучение без учителя).

3. Задача выбора информативного набора признаков при распознавании.

4. Задача приведения исходных данных к виду, удобному для распознавания.

5. Динамическое распознавание и динамическая классификация – задачи типов 1 и 2 для динамических объектов.

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

Основными задачами обработки изображений является обнаружение (определение границ) и локализация (определение местоположения) объектов. Основные требования к алгоритмам выделения объектов: высокая скорость при обработке больших изображений и не-

коэффициент ошибок при работе с зашумленными изображениями, которые к тому же могут иметь разные уровни яркости.

Идентификация и классификация проводится для уже локализованных на изображении объектов. Каждый из объектов характеризуется набором информативных признаков: цветом, формой и размером. Для идентификации объекта весь набор информативных признаков должен удовлетворять заданным условиям. Задача распознавания сводится к отысканию некой функции, отображающей множество образов (изображений) во множество, элементами которого являются классы образов.

Процесс идентификации и распознавания проводится, как правило, в три этапа.

Предварительная обработка. Заданное изображение $f(x)$ преобразуется в одно или несколько новых изображений $f_1(x, y), \dots, f_n(x, y)$ с помощью некоторого набора или последовательности определенных операций.

Выделение признаков. Функции $f_i(x, y)$ подвергаются функциональному преобразованию F_1, \dots, F_m , определяющему признаки, в результате чего изображение кодируется действительными числами.

Классификация. В результате выполнения первых двух этапов появляется набор чисел, которые можно считать признаками исходного изображения $f(x, y)$; этот набор можно рассматривать как точку в n -мерном пространстве. Если указаны области, занимаемые тем или иным классом в этом пространстве, либо на нем задана плотность вероятности для каждого класса, то на основании геометрической близости и максимальной вероятности, данное изображение можно отнести к определенному классу, т. е. «классифицировать».

Выполнение каждого из названных этапов с помощью ЭВМ представляет определенную трудность, которые связаны с названными выше особенностями изображений. В связи с этим приобретают большое значение различные методы фильтрации изображений, нормировки, получения инвариантов и формирования признаков, а также сегментации изображений и определение контуров для последующего выделения объектов.

Возможен и иной подход к обнаружению и локализации объектов на изображениях. Он заключается в проведении классификации отдельных пикселей или групп пикселей, после чего проводится локализация объектов, например, на основании близости к пикселям той или иной группы.

Пусть $f(x, y)$ — дискретная функция, описывающая анализируемое изображение; X — множество точек плоскости, на котором определена функция $f(x, y)$; $S = \{S_1, S_2, \dots, S_k\}$ — разбиение X на k непустых связных областей S_i , где $i = 1, 2, \dots, k$; L_p — предикат, заданный на множестве S и принимающий истинные значения тогда и только тогда, когда любая пара пикселей (x, y) из каждого подмножества S_i удовлетворяет некоторому критерию однородности подмножества. Сегментацией изображения $f(x, y)$ по предикату L_p называется разбиение X на k областей S_i , удовлетворяющее условиям:

$$1) \bigcup_{i=1}^k S_i = X;$$

$$2) S_i \cap S_j = \emptyset \text{ для любых } i \neq j;$$

$$3) L_p(S_i) \text{ принимает истинные значения для любого } i;$$

$$4) L_p(S_i \cup S_j) \text{ принимает ложные значения для любых } i \neq j.$$

Отметим, что k областей, полученных в результате сегментации часто различаются по размеру и группируются в m классов, где $2 \leq m \leq k$.

Большинство известных методов сегментации и выделения контуров, можно разделить на две основные группы: областно-ориентированные, когда непосредственно строятся S_i , и контурно-ориентированные, когда определяются границы между S_i . В методах первой группы критерием однородности при задании L_p выступают общие свойства изображения: значительность (цветности) пикселя, тип текстуры, спектральные свойства изображения

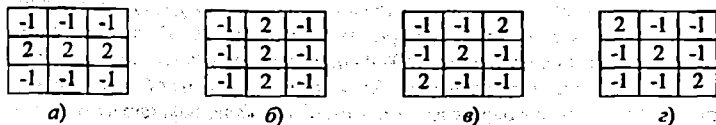
и т. п. Сюда относят методы порогового разделения, метод разрастания (увеличения) областей, метод деления областей. В методах второй группы реализуется отслеживание контура и при определении L_p используется характеристика изменения интенсивности – разность величин и направления градиента, т. е. два пикселя (x, y) и (x', y') подобны, если имеет место соотношение

$$L_p(S_i) = \left| \|\nabla f(x, y)\| - \|\nabla f(x', y')\| \right| \leq T; \quad (3.4)$$

$$\left| \varphi \nabla f(x, y) - \varphi \nabla f(x', y') \right| \leq A,$$

где ∇ – градиент функции; φ – направление градиента; T, A – пороговые значения.

Для вычисления градиента используются операторы локального анализа: Собель, Бергса, Кириша и т. д. В свою очередь, операторы свертки позволяют непосредственно брать пиксели, которые, например, могут принадлежать прямым определенной ориентации ширины (рис. 3.4.2).



ориентация прямых: а – 0°; б – 90°; в – 45°; г – 135°

Рисунок 3.4.2 – Операторы свертки для выделения прямых, толщиной в один пиксель

Не существует универсального критерия эффективности названных методов для обработки сильно зашумленных изображений, особенно если требуется выделять мелкие объекты. Выбор того или иного метода зависит от конечной цели всего процесса обработки изображений, типа обрабатываемых изображений и имеющихся вычислительных мощностей.

Найденные контура обычно представлены в ортогональной решетке, которые затем проксимируются прямыми линиями. При этом делается попытка максимально упростить изображение контура объекта, т. е. представить его минимальным числом прямых линий. Аппроксимация базируется на использовании клеточной логики, примитивов, преобразования Хафа.

Распознавание объектов топологии можно отнести к слабоструктурированным задачам и принятию решений с недостаточно формализованной предметной областью, в частности это одна из задач распознавания по прецедентности, для которых характерно наличие конечного множества возможных решений (прецедентов, классы заданы перечислением объектов) и механизма выбора лучшего из них.

Используются три типа моделей для представления информации о классах: детерминистская – в виде эталонов с фиксированными значениями признаков; нечеткая (квази- и постонечеткая, квазидетерминистская) – в виде эталонов, задаваемых с точностью до неопределенных параметров или группы непрерывных; статистическая – в виде статистических распределений множества параметров первичного описания (дискретных отсчетов, коэффициентов Фурье и т. п.). Выбор модели определяется особенностями классов распознаваемых изображений, а также соображениями достоверности правильной классификации и простоты реализации. Так, например, в случае детерминистской и квазидетерминистской моделей необходимо сначала определить близость распознаваемого изображения f до эталонных f_i

этого формируются некоторые функционалы – меры близости $\rho(f, f_{i,m})$. Наиболее часто используются следующие меры:

$$\begin{aligned} \rho_1^1 &= \max |f(x, y) - f_{i,m}(x, y)|; \\ \rho_1^2 &= \left\{ \frac{1}{S_D} \iint_{(D)} [f(x, y) - f_{i,m}(x, y)]^2 dx dy \right\}^{\frac{1}{2}}; \\ \rho_1^3 &= \frac{\left[\iint_{(D)} f(x, y) f_{i,m}(x, y) dx, dy \right]^2}{\iint_{(D)} f(x, y) dx dy \iint_{(D)} f_{i,m}^2(x, y) dx, dy}. \end{aligned} \quad (3.4.2)$$

Меры ρ_1^1 и ρ_1^2 характеризуют отклонение от эталона, а мера ρ_1^3 является коэффициентом взаимной корреляции. Если используются нечеткие модели, то в качестве мер близости естественно задавать минимальные значения ρ_1^1 и ρ_1^2 , но максимальное значение ρ_1^3 , достигая путем вариации этих параметров.

Алгоритмы распознавания реализуют принцип похожести новых объектов с заданными по прецедентности. Основным недостатком таких алгоритмов является их эвристический характер (в основу положено то или иное дополнительное интуитивное предположение – эвристика). При использовании таких алгоритмов налагаются дополнительные условия: число проб отказов должно быть минимальным и не превышать заданного числа при определенных ограничениях на распознаваемые объекты.

Каждый из используемых методов имеет свои достоинства и недостатки. Так, корреляционные методы обладают относительно высокой вычислительной сложностью, к тому же их эффективность существенно снижается при наличии различных уровней яркости и увеличении уровня шумов на анализируемом изображении. Алгоритмы, основанные на использовании метрических признаков, менее чувствительны к вариациям яркости, однако очень чувствительны к уровню шумов. Использование моментных инвариантов осложняется высокой чувствительностью этих признаков даже к небольшим вариациям формы выделяемых объектов относительно эталонов.

Как правило, эффективные методы удается получить с учетом специфики изображения. Поэтому предлагаемые методы существенно учитывают особенности изображений, а алгоритмы обработки должны быть параметрически настраиваемыми, причем шаги каждого алгоритма также могут варьироваться.

Построение множества информативных признаков. Под выделением признаков подразумевают переход от исходных измерений X к новому выбору более информативных в рассматриваемой задаче переменных $y = [y_1, y_2, \dots, y_M]^T$. При этом $y_i = F_i(x_1, x_2, \dots, x_N) = F_i(\mathbf{x})$, $i = 1, \dots, M$, где F_i – функции преобразования переменных или признаков операторы. Эти функции (операторы) могут осуществлять просто отбор части исходных измерений ($M < N$), масштабирование $y_i = \alpha_i x_i$, $i = 1, \dots, M$, глобальные (локальные) преобразования и т. д.

Для получения признаков могут быть использованы коэффициенты разложения по полнортогональным системам функций. К ним следует отнести тригонометрические полиномы, полиномы Лежандра, полиномы Чебышева, коэффициенты обобщенного ряда Фурье, а также функции Хаара, Уолша и некоторые другие. Заметим, что также могут быть использованы для формирования признаков квазиортогональные системы функций, например, двумерные случайные функции. По сравнению с функциями Уолша, Хаара и другими, такие

функции обладают с точки зрения формирования признаков одним существенным преимуществом – увеличением частоты элементов структуры с ростом номера реализации функции, т. е. изображение становится все более и более мелкоструктурным.

Главным требованием при получении признаков является обеспечение инвариантности метода идентификации и распознавания к размещению объекта, его угловой ориентации, масштабу, допустимым топологическим преобразованиям и прочим атрибутам. Использование инвариантных методов позволяет повысить достоверность распознавания.

Процедуры выделения признаков можно разделить на следующие три типа:

- глобальные преобразования и разложение в ряд;
- признаки, выводимые из статистических распределений точек изображения;
- геометрические и топологические признаки.

Информативные признаки, выделенные из статистических распределений пикселей изображения, включают обработку и анализ характерных зон, моменты, k -уровневые фильтры, расстановки. Они обеспечивают как высокую скорость, так и невысокую сложность вычисления на ЭВМ. Тем не менее, конструирование масок для этого типа признаков является довольно трудной задачей.

Методы геометрического и топологического анализа признаков могут представлять и глобальные, так и локальные свойства распознаваемых объектов. Тем не менее, процесс выделения признаков этого типа весьма сложен, и столь же сложно конструирование масок для них.

Методы на основе глобальных преобразований и коэффициентов разложения в ряд позволяют понизить размерность вектора признаков и в ряде случаев дают возможность подбирать признаки, инвариантные к некоторым глобальным деформациям изображения, такие как сдвиг или поворот. Глубокие теоретические разработки в области линейных операторов одно- и двумерных преобразований, цифровой фильтрации, а также значительный прогресс области вычислительной техники, появление доступных высокопроизводительных систем способствуют их внедрению в область распознавания.

Приведем описания нескольких подходов каждого типа, которые были апробованы при решении задач распознавания объектов топологии интегральных схем.

3.5. Нейронные сети как инструмент решения сложных задач

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

Интерес к нейроинтеллекту возник еще на заре развития вычислительной техники. Его основе лежит нейронная организация искусственных систем, которая имеет биологические предпосылки. Способность биологических систем к обучению, самоорганизации и адаптации имеет большое преимущество по сравнению с современными вычислительными системами. Достоинством компьютерных систем является большая скорость распространения информации и возможность учета большого объема знаний, накопленных человечеством.

и области. Разработка искусственных разумных систем, которые соединяют преимуществ биологических существ и современной вычислительной техники, создаст потенциальные предпосылки для перехода к качественно новому этапу эволюции в вычислительной сфере.

Нейронные сети имеют различные периоды своего развития, связанные с оптимистичным и пессимистичным взглядом на эволюцию данной технологии в контексте развития искусственного интеллекта. Первые шаги в области искусственных нейронных сетей были в 1943 г. В. Мак-Калох (W. McCulloch) и В. Питс (W. Pitts). Они показали, что при определенных пороговых нейронных элементах можно реализовать исчисление любых логических функций. В 1949 г. Д. Хебб предложил правило обучения, которое стало математической основой для обучения ряда нейронных сетей. В 1957-1962 гг. Ф. Розенблатт предложил и назвал модель нейронной сети, которую он назвал перцептроном. Результаты исследования обобщил в книге «Принципы нейродинамики», которая имеет большое значение для теории нейронных сетей. В 1959 г. В. Видроу (W. Widrow) и М. Хофф (M. Hoff) предложили процедуру обучения для линейного адаптивного элемента, который они назвали «ADALINE». Процедура обучения получила название «дельта правило». В 1969 г. М. Минский (M. Minsky) и С. Паперт (S. Papert) опубликовали монографию «Перцептрон», в которой осуществили математический анализ перцептрона и показали ограничения, присущие ему из-за способности решать линейно неразделимые задачи. Хотя, как будет показано далее, однопольный перцептрон с сигнальной функцией активации способен решить задачу «исключает или». Выводы их были довольно пессимистичными, и это сыграло негативную роль в дальнейшем развитии исследований в области нейронных сетей. Работы в области перцептронных сетей были практически приостановлены. В 70-е годы появился ряд работ в области ассоциативной памяти. Так, Андерсон (Anderson) предложил в 1977 г. модель линейно-ассоциативной памяти. В этом направлении продолжил исследования Т. Кохонен (T. Kohonen), который предложил модель оптимальной линейной ассоциативной памяти. В 1976 г. 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. Д. Хопфилд (J. Hopfield) в 1982 г. произвел анализ устойчивости нейронных сетей с обратными связями и предложил использовать их для решения задач оптимизации. Г. Кохонен разработал и исследовал самоорганизующиеся нейронные сети. Ряд авторов (Elman, Hinton, Williams) в 1986 г. предложили алгоритм обратного распространения ошибки, который стал мощным средством для обучения многослойных нейронных сетей. В том же научной среде вплоть до 2006 г. господствовала парадигма, что не имеет смысла использовать более двух скрытых слоев в перцептроне. Это было связано с тем, что алгоритм обратного распространения ошибки не давал никакого выигрыша в решении задач при обучении перцептрона с более чем двумя скрытыми слоями. Также данная парадигма базировалась на теореме об универсальной аппроксимации перцептроном с одним или двумя скрытыми слоями любой функции со сколь угодно заданной точностью. В результате использования перцептронов с малым количеством скрытых слоев сложилось мнение, что его возможности в основном исчерпаны. Тем более, что появились машины опорных векторов (SVM), которые показывали часто большую эффективность на операциях распознавания объектов по сравнению с перцептроном. Поэтому интерес к многослойным перцептронам стал постепенно спадать. И только, начиная с 2006 г., благодаря работам Хинтона (G. Hinton), произошел обратный отсчет в развитии многослойных перцептронов под новым названием «глубокие нейронные сети глубокого доверия (deep belief neural networks)». Такие сети в общем случае представляют собой дальнейшее развитие многослойных перцептронов и интегрируют различные парадигмы обучения нейронных сетей. Такие сети, благодаря многослойной архитектуре, позволяют обрабатывать и анализировать большой объем данных, а также моделировать когнитивные процессы в различных областях. В настоящее время большинство высокотехнологических компаний в США (Microsoft, Google и т. д.) используют нейронные сети

глубокого доверия для проектирования различных интеллектуальных систем. По версии ученых Массачусетского технологического института (США), нейронные сети глубокого доверия (deep belief neural networks), входят в список 10 наиболее прорывных высоких технологий, способных в недалеком будущем в значительной степени преобразить повседневную жизнь большинства людей на нашей планете. В настоящее время нейронные сети глубокого доверия считаются революционным шагом в области интеллектуальной обработки данных.

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.

Формальный нейрон. Основным элементом нейронной сети (НС) 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 $X = (x_1, \dots, x_n)^T$ – вектор входного сигнала;

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

Схема нейронного элемента изображена на рис. 3.4.1 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.

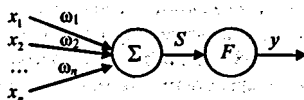


Рисунок 3.4.1 – Нейронный элемент

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 , α – угол между векторами W и X .

Длины весового и входного векторов определяются через их координаты:

$$|W| = \sqrt{\omega_1^2 + \omega_2^2 + \dots + \omega_n^2},$$

$$|X| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}.$$

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 w_i 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 решаемой задачи. Цель ее настройки — решение избранного класса задач. Настройка сети обычно сопровождается изменением весовых коэффициентов w_{ij} при многократной подаче образов до тех пор, пока не будет гарантирован предельный уровень ошибки в работе сети.

Однослойный перцептрон. В настоящее время существует устойчивое мнение, что однослойный перцептрон не может решить задачу «исключающее или». Однако это справедливо только для перцептрона с пороговой или монотонно возрастающей функцией активации. При использовании сигнальной или унимодальной функции активации однослойный перцептрон, как будет показано далее, способен решить проблему «исключающее или».

Однослойный перцептрон принято изображать в виде двухслойной нейронной сети, где первый слой нейронных элементов является распределительным, а второй — обрабатывающим. Распределительный слой передает входные сигналы на обрабатывающий слой нейронных элементов, который преобразует входную информацию в соответствии с синаптическими связями и функцией активации (рис. 3.4.2). При этом каждый нейрон распределительного слоя имеет синаптические связи со всеми нейронами обрабатывающего слоя.

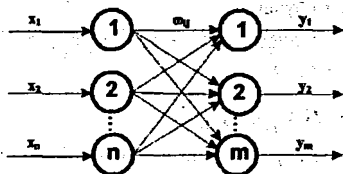


Рисунок 3.4.2 — Однослойный перцептрон

Выходное значение j -го нейронного элемента второго слоя можно представить как

$$y_j = F(S_j) = F\left(\sum_{i=1}^n a_{ij} x_i - T_j\right),$$

где T_j — порог j -го нейронного элемента выходного слоя; a_{ij} — сила синаптической связи между i -м нейроном распределительного слоя и j -м нейроном обрабатывающего слоя, F — оператор нелинейного преобразования или функция активации нейронных элементов.

Рассмотрим правило обучения однослойного персептрона, который имеет линейную функцию активации. Такое правило обучения называется дельта правилом (delta rule) и может использоваться как для обучения линейного персептрона, так и для персептрона с пороговой функцией активации. Дельта правило было предложено в 60-х годах Ф. Розенблаттом [22] для персептрона с пороговой функцией активации, а для для линейной сети его предложили Видроу (Widrow) и Хофф (Hoff).

Выходное значение линейного однослойного персептрона определяется, как

$$y_j = \sum_{i=1}^n \omega_{ij} x_i - T_j$$

Дельта правило предполагает минимизацию суммарной среднеквадратичной ошибки нейронной сети, которая для L входных образов определяется следующим образом:

$$E_j = \sum_{k=1}^L E(k) = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^m (y_j^k - e_j^k)^2,$$

где $E(k)$ – среднеквадратичная ошибка сети для k -го образа; y_j^k и e_j^k – соответственно выходное и эталонное значение нейронной сети для k -го образа.

Для минимизации суммарной квадратичной ошибки используется метод градиентного спуска. Существует два основных подхода к обучению однослойного персептрона: последовательное обучение (online learning) и групповое обучение (batch learning). При последовательном обучении модификация синаптических связей происходит после подачи каждого образа из обучающей выборки на нейронную сеть. В этом случае в методе градиентного спуска используется квадратичная ошибка нейронной сети для одного входного образа:

$$E = \frac{1}{2} \sum_{j=1}^m (y_j - e_j)^2.$$

Тогда в соответствии с методом градиентного спуска, весовые коэффициенты и пороги нейронной сети необходимо изменять с течением времени по следующим выражениям:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \frac{\partial E}{\partial \omega_{ij}(t)},$$

$$T_j(t+1) = T_j(t) - \alpha \frac{\partial E}{\partial T_j(t)},$$

где α – скорость или шаг обучения.

Отсюда можно получить *дельта-правило* для последовательного обучения:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha (y_j - e_j) x_i,$$

$$T_j(t+1) = T_j(t) + \alpha (y_j - e_j).$$

Величина адаптивного шага при последовательном обучении равна

$$\alpha(t) = \frac{1}{1 + \sum_i x_i^2(t)}.$$

При использовании группового обучения модификация синаптических связей будет происходить после подачи на вход сети L образов. В этом случае дельта правило можно представить следующим образом:

$$\omega_{ij}(L) = \omega_{ij}(0) - \alpha(t) \sum_{k=1}^L (y_j^k - e_j^k) \cdot x_i^k,$$

$$T_j(L) = T_j(0) + \alpha(t) \sum_{k=1}^L (y_j^k - e_j^k).$$

Величина адаптивного шага при групповом обучении определяется соотношениями:

$$\alpha(t) = \frac{\sum_k \sum_j (y_j^k - e_j^k) a_j^k}{\sum_k \sum_j (a_j^k)^2},$$

$$a_j^k = \sum_p (y_j^p - e_j^p) (1 + \sum_i x_i^p x_i^k).$$

Однослойный перцептрон с пороговой функцией активации формирует линейную разделяющую поверхность с точки зрения классификации образов и поэтому не может решить задачу «исключающее или». Это в свое время показали американские ученые Минский и Перл, которые сделали пессимистические выводы насчет дальнейшего развития нейронных сетей. Однако последнее утверждение справедливо только для

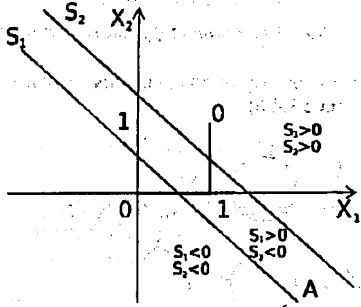


Рисунок 3.4.3 – Решение задачи «исключающее или»

однослойного перцептрона с пороговой или монотонной непрерывной функцией активации (например, сигмоидная). При использовании функции активации Гаусса или сигнальной функции однослойный перцептрон может решить задачу «исключающее или». Покажем это на примере сигнальной функции активации. В этом случае, необходимо выделить область единичных значений при помощи двух прямых (рис. 3.4.3). Область А, которая определяет класс единичных значений характеризуется следующим условием:

$$(S_1 > 0) \& (S_2 < 0).$$

Уравнения прямых можно представить следующим образом:

$$S_1 = X_2 + X_1 - 0,5,$$

$$S_2 = X_2 + X_1 - 1,5.$$

Отсюда можно получить, что $S_2 = S_1 - 1$ (рис.3.4.4).

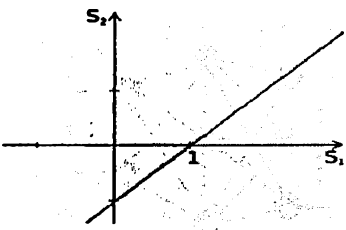


Рисунок 3.4.4 – Зависимость S2 от S1

Тогда, если $S_1 > 0$ и $S_2 < 0$, то $0 < S_1 < 1$, и можно получить следующую функцию активации (рис.3.4.5):

$$y = F(S) = F(S_1) = \begin{cases} 1, & 0 < S_1 < 1 \\ 0, & \text{иначе} \end{cases}$$

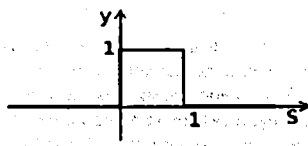


Рисунок 3.4.5 – Сигнальная функция активации

В результате однослойный перцептрон для решения задачи «исключающее или» будет иметь следующие веса и порог (рис.3.4.6):

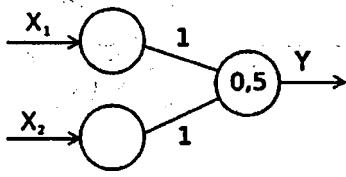


Рисунок 3.4.6 – Однослойный перцептрон для решения задачи «исключающее или»

Таким образом, при использовании сигнальной функции активации однослойный перцептрон может решить задачу «исключающее или», так как она разбивает входное пространство образов на классы при помощи двух прямых. При использовании однослойного перцептрона с радиально базисной функцией активации для решения данной задачи необходимо принимать выходное значение перцептрона равным единице, если оно больше определенного значения.

Многослойный перцептрон. В 1986 рядом авторов (Rumelhart, Hinton, Williams) независимо друг от друга был предложен алгоритм обратного распространения ошибки (backpropagation algorithm), который стал эффективным средством обучения многослойных нейронных сетей. Перцептрон с одним скрытым слоем изображен на рис. 3.4.7.

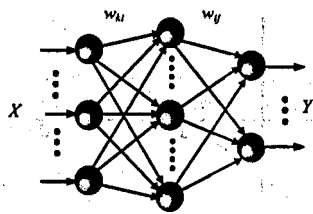


Рисунок.3.4.7 – Перцептрон с одним скрытым слоем

Выходное значение j -го нейрона последнего слоя определяется как

$$y_j = F(S_j),$$

$$S_j = \sum_i \omega_{ij} y_i - T_j.$$

Аналогичным образом для скрытого слоя имеем:

$$y_i = F(S_i),$$

$$S_i = \sum_k \omega_{ik} x_k - T_i,$$

где S_j — взвешенная сумма j -го нейрона, ω_{ij} — весовой коэффициент от i -го к j -му нейрону — порог j -го нейрона последнего слоя.

Целью обучения является минимизация суммарной квадратичной ошибки сети:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_j (y_j^k - e_j^k)^2.$$

где L — размерность обучающей выборки, y_j^k и e_j^k — соответственно выходное и эталонное значение нейронной сети для k -го образа.

Для минимизации квадратичной ошибки сети весовые коэффициенты и пороги нейронных элементов должны изменяться с течением времени следующим образом:

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha \gamma_j F'(S_j) y_i,$$

$$T_j(t+1) = T_j(t) + \alpha \gamma_j F'(S_j),$$

где $\frac{\partial E}{\partial S_j} = F'(S_j)$, γ_j — ошибка j -го нейрона.

Для выходного слоя $\gamma_j = y_j - e_j$.

Для любого скрытого слоя i ошибка i -го нейронного элемента определяется рекурсивно через ошибки нейронов следующего слоя j :

$$\gamma_i = \sum_{j=1}^m \gamma_j F'(S_j) \omega_{ij},$$

где m — количество нейронов следующего слоя по отношению к слою i ; ω_{ij} — синаптическая связь между i -м и j -м нейроном различных слоев; S_j — взвешенная сумма j -го нейрона.

Данное правило обучения многослойных нейронных сетей называется *обобщенным та-правилом*.

Для группового обучения, когда модификация синаптических связей происходит после L образов на нейронную сеть:

$$\omega_{ij}(L) = \omega_{ij}(0) - \alpha(t) \cdot \sum_k \gamma_j^k F'(s_j^k) y_i^k,$$

$$T_j(L) = T_j(0) + \alpha(t) \cdot \sum_k \gamma_j^k F'(s_j^k).$$

Рассмотрим возможности многослойного персептрона в зависимости от количества слоев. В литературе существует много ошибок по данному вопросу. Так, например, манн (R. Lippmann) утверждал в работе, что персептрон с одним скрытым слоем может аппроксимировать только выпуклую разделяющую поверхность с точки зрения классификации. Позже было показано, что такой персептрон может формировать произвольную

разделяющую поверхность, однако данная ошибка попала и попадает во многие учебники. Прежде всего, следует отметить, что возможности персептрона с одним скрытым слоем различаются в зависимости от используемой функции активации. В 1989 г. G. Cybenko, доказав теорему об универсальной аппроксимации:

Теорема 1. Любая функция может быть аппроксимирована персептроном с одним скрытым слоем и сигмоидной функцией активации нейронных элементов с любой степенью точности при достаточном количестве нейронов в скрытом слое.

Данная теорема является основной для аппроксимации функций при помощи многослойных нейронных сетей. При этом точность аппроксимации зависит от количества нейронов в скрытом слое. Чем больше количество нейронных элементов в скрытом слое, тем больше точность. Однако при слишком большой размерности скрытого слоя может наступить явление, которое называется перетренировкой сети, когда сеть имеет плохую обобщающую способность. Таким образом, нейронная сеть с одним скрытым слоем позволяет осуществить любое отображение входных сигналов в выходные при использовании сигмоидной или любой ограниченной, монотонно возрастающей функции активации. Поэтому такая сеть является также универсальным классификатором, т. е. она способна формировать произвольную разделяющую поверхность в пространстве решений.

Рассмотрим теперь другой случай, когда в качестве функции активации нейронных элементов используется пороговая функция. Здесь ситуация меняется. Персептрон с одним скрытым слоем не является универсальным классификатором. В данном случае, только персептрон с двумя скрытыми слоями является универсальным классификатором. Покажем это на примере решения простой задачи бинарной классификации (рис. 3.4.8).

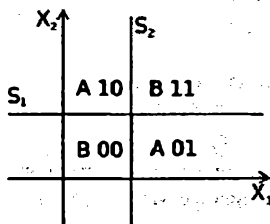


Рисунок 3.4.8 – Задача бинарной классификации двух классов A и B

Как видно из рисунка классы разделяются двумя прямыми S_1 и S_2 . Уравнения прямых будут следующие: $S_1 = X_2 - 1$ и $S_2 = X_1 - 1$. В результате, как следует из рис. 3.4.8, данная задача сводится к проблеме «исключающее или». Поэтому персептрон для решения данной задачи будет состоять из двух скрытых слоев, где первый скрытый слой формирует разделяющие линии S_1 и S_2 (рис.3.4.9).

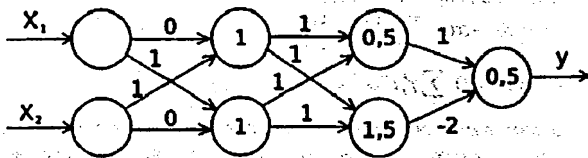


Рисунок 3.4.9 – Персептрон с двумя скрытыми слоями для решения задачи бинарной классификации

Рассмотрим еще один случай, когда используется сигнальная функция активации в от-
 -ых слоях нейронных элементов. Тогда для решения приведенной выше задачи класси-
 -ия достаточно персептрона с одним скрытым слоем и сигнальной функцией активации
 -одним нейроне, как это показано на рис.3.4.10. Здесь нейроны скрытого слоя формиру-
 -критичные линии S_1 и S_2 , а выходной нейронный элемент с сигнальной функцией
 -и формирует окончательное решение задачи.

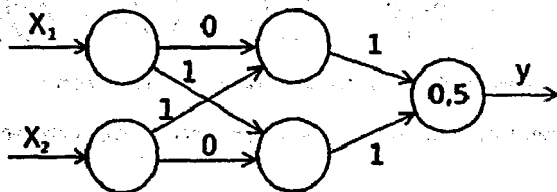


Рисунок 3.4.10 – Персептрон с одним скрытым слоем для решения задачи бинарной классификации

Таким образом, основная научная парадигма до 2006 г. состояла в том, что персептрон
 или максимум двумя скрытыми слоями является достаточным для решения различ-
 -задач. Использование персептрона с более чем двумя скрытыми слоями не имеет боль-
 -о смысла.

Нейронные сети глубокого доверия. В общем случае нейронные сети глубокого до-
 -я (deep belief neural networks) представляют собой персептрон с множеством слоев ней-
 -ных элементов. До 2006 года в научной среде была приоритетной парадигма, что много-
 -ый персептрон с одним, максимум двумя скрытыми слоями является более эффектив-
 - для нелинейного преобразования входного пространства образов в выходное по сравне-
 - с персептроном с большим количеством скрытых слоев. Считалось, что персептрон с
 -е чем двумя скрытыми слоями не имеет смысла применять. Данная парадигма базирова-
 - на теореме, что персептрон с одним скрытым слоем является универсальным аппрок-
 -атором. Другой аспект этой проблемы заключается в том, что все попытки использовать
 -ритм обратного распространения ошибки (backpropagation algorithm) для обучения пер-
 -трона с тремя и более скрытыми слоями не приводили к улучшению решения различных
 -ч. Это связано с тем, что алгоритм обратного распространения ошибки является неэф-
 -тивным для обучения персептронов с тремя и более слоями. В 2006 Хинтон (Hinton)
 -ложил «жадный» алгоритм послонного обучения (greedy layer-wise algorithm), который
 - эффективным средством обучения нейронных сетей глубокого доверия, которые как
 -отмечалось, представляют собой персептрон с большим количеством слоев. Было пока-
 -но, что нейронная сеть глубокого доверия имеет большую эффективность нелинейного
 -образования и представления данных по сравнению с традиционным персептроном. Та-
 -сеть осуществляет глубокое иерархическое преобразование входного пространства обра-
 -В результате первый скрытый слой выделяет низкоуровневое пространство признаков
 -ных данных, второй слой детектирует пространство признаков более высокого уровня
 -ракции (иерархии) и т. д.

Как уже отмечалось, нейронная сеть глубокого доверия содержит множество скрытых
 -в (рис. 3.4.11) и осуществляет глубокое иерархическое преобразование входного про-
 -ства образов.

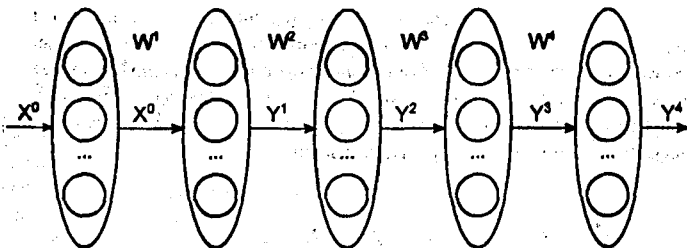


Рисунок 3.4.11 – Нейронная сеть глубокого доверия

Выходное значение j -го нейрона k -го слоя определяется следующим образом:

$$y_j^k = F(S_j^k),$$

$$S_j^k = \sum_{i=1}^n w_{ij}^k y_i^{k-1} + T_j^k,$$

где F – функция активации нейронного элемента, S_j^k – взвешенная сумма j -го нейрона k -слоя, w_{ij}^k – весовой коэффициент между i -ым нейроном $(k-1)$ -го слоя и j -м нейроном k -го слоя, T_j^k – пороговое значение j -го нейрона k -го слоя.

Для первого (распределительного) слоя

$$y_i^0 = x_i.$$

В матричном виде выходной вектор k -го слоя

$$Y^k = F(S^k) = F(W^k Y^{k-1} + T^k),$$

где W – матрица весовых коэффициентов, Y^{k-1} – выходной вектор $(k-1)$ -го слоя, T^k – вектор пороговых значений нейронов k -го слоя.

Если нейронная сеть глубокого доверия используется для классификации образов, то выходные значения сети часто определяются на основе функции активации softmax:

$$y_j^F = \text{softmax}(S_j) = \frac{e^{S_j}}{\sum_i e^{S_i}}.$$

Процесс обучения нейронных сетей глубокого доверия в общем случае состоит из двух этапов:

1. Предобучение нейронной сети методом послойного обучения, начиная с первого слоя (pre-training). Данное обучение осуществляется без учителя.

2. Настройка синаптических связей всей сети (fine-tuning) при помощи алгоритма обратного распространения ошибки или алгоритма «бодрствования и сна» (wake-sleep algorithm).

Важным этапом обучения нейронных сетей глубокого доверия является предобучение слоев нейронной сети. Существует два основных подхода к предварительному обучению слоев нейронных сетей глубокого доверия. Первый подход называется автоэнкодерным и базируется на представлении каждого слоя в виде автоассоциативной нейронной сети (рециркуляционная или PCA нейронная сеть). Второй подход базируется на представлении каждого слоя нейронной сети в виде ограниченной машины Больцмана (RBM).

В случае автоэнкодерного метода, вначале обучается первый слой как автоассоциативная нейронная сеть с целью минимизации суммарной ошибки реконструкции информации.

второй и так далее. Для обучения каждого слоя можно использовать алгоритм обрат-аспространения ошибки. После этого осуществляется точная настройка синаптических весов всей сети (fine tuning), используя алгоритм обратного распространения ошибки.

Рассмотрим персептрон с тремя скрытыми слоями (рис.3.4.12). Тогда в соответствии с кодерным методом, прежде всего, берутся первые два слоя нейронной сети (1 и 2), и на их основе строится автоассоциативная (рециркуляционная) нейронная сеть (1-2-1), управляемая восстанавливающим слоем (рис.3.4.13). Затем происходит обучение такой сети (например при помощи алгоритма обратного распространения ошибки, с целью минимизации ошибки реконструкции информации). Продолжительность обучения обычно составляет больше, чем 100 эпох.

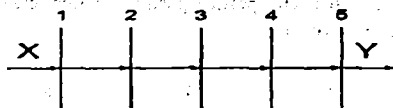


Рисунок 3.4.12 – Персептрон с тремя скрытыми слоями

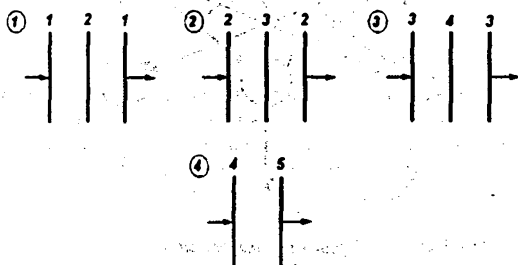


Рисунок 3.4.13 – Автоэнкодерный метод обучения

После этого отбрасывается восстанавливающий слой (последний слой), фиксируются веса скрытого слоя, и строится автоассоциативная сеть из следующих двух слоев нейронной сети (2-3-2), которая обучается на основе данных поступающих с предыдущего (2-го) слоя.

Процесс продолжается до последнего или предпоследнего слоя, как это схематично показано на рис. 3.4.13. В результате послойного обучения получается предварительно обученная нейронная сеть. Далее осуществляется точная настройка (fine tuning) посредством метода обратного распространения ошибки с учителем.

Данный процесс можно представить в виде следующего алгоритма:

1. Строится автоассоциативная сеть с входным слоем X , скрытым Y и выходным X .
2. Обучается автоассоциативная сеть, например при помощи алгоритма обратного распространения ошибки (как правило не более 100 эпох) и фиксируются синаптические связи второго слоя W^1 .
3. Берется следующий слой и формируется автоассоциативная сеть аналогичным образом.
4. Используя настроенные синаптические связи предыдущего слоя W^1 , подаем входные данные на вторую автоассоциативную сеть и обучаем ее аналогичным образом. В результате получаются весовые коэффициенты второго слоя W^2 .
5. Процесс продолжается до последнего слоя нейронной сети.

6. Берется последний слой нейронной сети и обучается с учителем.

7. Обучается вся сеть для точной настройки параметров при помощи алгоритма обратного распространения ошибки.

Второй подход к обучению нейронных сетей глубокого доверия базируется на представлении каждого слоя нейронной сети в виде ограниченной машины Больцмана (RBM). Ограниченная машина Больцмана состоит из двух слоев стохастических бинарных нейронных элементов, которые соединены между собой двунаправленными симметричными связями (рис.3.4.14). Входной слой нейронных элементов называется видимым (слой X), а второй слой называется скрытым (слой Y). Нейронную сеть глубокого доверия можно представить как совокупность ограниченных машин Больцмана. Ограниченная машина Больцмана может генерировать (представить) любое дискретное распределение, если используется достаточное количество нейронов скрытого слоя.

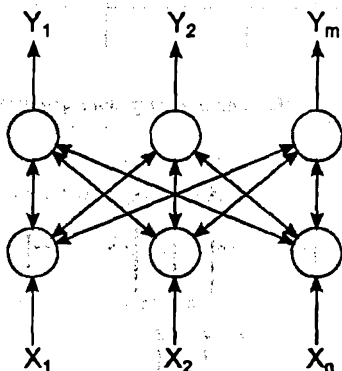


Рис.3.4.14 – Ограниченная машина Больцмана

Данная сеть является стохастической нейронной сетью, в которой состояния видимых скрытых нейронов меняются в соответствии с вероятностной версией сигмоидной функции активации (состояния видимых и скрытых нейронных элементов принимаются независимыми):

$$P(y_j | x) = \frac{1}{1 + e^{-S_j}}, \quad S_j = \sum_i w_{ij} x_i + T_j,$$

$$P(x_i | y) = \frac{1}{1 + e^{-S_i}}, \quad S_i = \sum_j w_{ij} y_j + T_i,$$

$$P(x | y) = \prod_{i=1}^n P(x_i | y)$$

$$P(y | x) = \prod_{j=1}^m P(y_j | x)$$

Таким образом, состояния всех нейронных элементов ограниченной машины Больцмана определяются через распределение вероятностей. В сети RBM нейроны скрытого слоя являются детекторами признаков, которые сохраняют закономерности входных данных.

Задача обучения состоит в воспроизведении распределения входных данных на основании нейронов скрытого слоя как можно точнее. Это эквивалентно максимизации правдоподобия путем модификации синаптических связей нейронной сети. Рассмотрим это подробнее.

Вероятность нахождения видимого и скрытого нейрона в состоянии (x, y) определяется новым распределением Гиббса:

$$P(x, y) = \frac{e^{-E(x, y)}}{Z}$$

где $E(x, y)$ – энергия системы в состоянии (x, y) , Z – параметр, который определяет усложнение нормализации вероятностей, т. е., чтобы сумма вероятностей равнялась единице. Данный параметр определяется следующим образом:

$$Z = \sum_{x, y} e^{-E(x, y)}$$

Вероятность нахождения видимых нейронов в определенном состоянии равняется сумме вероятностей конфигураций $P(x, y)$ по состояниям скрытых нейронов:

$$P(x) = \sum_y P(x, y) = \sum_y \frac{e^{-E(x, y)}}{Z} = \frac{\sum_y e^{-E(x, y)}}{\sum_{x, y} e^{-E(x, y)}}$$

Для нахождения правила модификации синаптических связей необходимо максимизировать вероятность воспроизведения состояний видимых нейронов $P(x)$ ограниченной максимальной Больцмана. Для того, чтобы определить максимум функции правдоподобия распределения данных $P(x)$ будем использовать метод градиентного спуска в пространстве весовых коэффициентов и пороговых значений сети, где в качестве градиента применим функцию арифметического правдоподобия:

$$\ln P(x) = \ln \sum_y e^{-E(x, y)} - \ln \sum_{x, y} e^{-E(x, y)}$$

Тогда градиент равен

$$\frac{\partial \ln(P(x))}{\partial w_j} = \frac{\partial}{\partial w_j} (\ln \sum_y e^{-E(x, y)}) - \frac{\partial}{\partial w_j} (\ln \sum_{x, y} e^{-E(x, y)})$$

Преобразуя последнее выражение, получим

$$\frac{\partial \ln(P(x))}{\partial w_j} = \frac{1}{\sum_y e^{-E(x, y)}} \sum_y e^{-E(x, y)} \frac{\partial E(x, y)}{\partial w_j} + \frac{1}{\sum_{x, y} e^{-E(x, y)}} \sum_{x, y} e^{-E(x, y)} \frac{\partial E(x, y)}{\partial w_j}$$

Так как

$$P(x, y) = P(y/x)P(x),$$

то

$$P(y/x) = \frac{P(x, y)}{P(x)} = \frac{(1/Z)e^{-E(x, y)}}{(1/Z)\sum_y e^{-E(x, y)}} = \frac{e^{-E(x, y)}}{\sum_y e^{-E(x, y)}}$$

В результате можно получить следующее выражение:

$$\frac{\partial \ln(P(x))}{\partial \omega_{ij}} = - \sum_y P(y/x) \frac{\partial E(x,y)}{\partial \omega_{ij}} + \sum_{x,y} P(x,y) \frac{\partial E(x,y)}{\partial \omega_{ij}}$$

В данном выражении первое слагаемое определяет позитивную фазу работы машины Больцмана, когда сеть работает на основе образов из обучающей выборки. Второе слагаемое характеризует негативную фазу функционирования, когда сеть работает в свободном режиме независимо от окружающей среды.

Рассмотрим энергию сети RBM. С точки зрения энергии сети задача обучения состоит в том, чтобы на основе входных данных найти конфигурацию выходных переменных с минимальной энергией. В результате на обучающем множестве сеть будет иметь меньшую энергию по сравнению с другими состояниями. Функция энергии бинарного состояния (x, y) определяется аналогично сети Хопфилда:

$$E(x, y) = - \sum_i x_i T_i - \sum_j y_j T_j - \sum_{i,j} x_i y_j w_{ij}$$

В этом случае

$$\frac{\partial E(x, y)}{\partial \omega_{ij}} = -x_i y_j$$

и

$$\frac{\partial \ln(P(x))}{\partial \omega_{ij}} = \sum_y P(y/x) x_i y_j - \sum_{x,y} P(x,y) x_i y_j$$

Так как математическое ожидание равняется:

$$E(x) = \sum_i x_i P_i$$

то

$$\frac{\partial \ln(P(x))}{\partial \omega_{ij}} = E[x_i y_j]_{data} - E[x_i y_j]_{model}$$

Аналогичным образом можно получить градиенты для пороговых значений:

$$\frac{\partial \ln(P(x))}{\partial T_i} = E[x_i]_{data} - E[x_i]_{model}$$

$$\frac{\partial \ln(P(x))}{\partial T_j} = E[y_j]_{data} - E[y_j]_{model}$$

Как следует из последних выражений, первое слагаемое характеризует работу сети на основе данных из обучающей выборки, а второе слагаемое характеризует работу сети на основе данных модели (данные генерируемые сетью), т. е. в свободном режиме независимо от окружающей среды.

Так как вычисление математического ожидания на основе RBM сети является очень медленным, Хинтон предложил использовать аппроксимацию данных слагаемых, которую он назвал контрастным расхождением (contrastive divergence (CD)).

Такая аппроксимация основывается на дискретизаторе Гиббса (Gibbs sampling). В этом случае первые слагаемые в выражениях для градиента характеризуют распределение данных в момент времени $t = 0$, а вторые слагаемые характеризуют реконструированные или генерируемые моделью состояния в момент времени $t = k$. Исходя из этого, CD-к процедура может быть представлена следующим образом:

$$x(0) \rightarrow y(0) \rightarrow x(1) \rightarrow y(1) \rightarrow \dots \rightarrow x(k) \rightarrow y(k).$$

В результате можно получить следующие правила для обучения RBM сети. В случае изменения CD-1, $k = 1$, и учитывая, что в соответствии с методом градиентного спуска

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha \frac{\partial \text{Ln} P(y)}{\partial \omega_{ij}(t)},$$

можно получить, что

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(1)y_j(1))$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(1)),$$

$$T_j(t+1) = T_j(t) + \alpha(y_j(0) - y_j(1)).$$

Аналогичным образом, для алгоритма CD-к имеем:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha(x_i(0)y_j(0) - x_i(k)y_j(k)),$$

$$T_i(t+1) = T_i(t) + \alpha(y_j(0) - y_j(k)),$$

$$T_i(t+1) = T_i(t) + \alpha(x_i(0) - x_i(k)).$$

Из последних выражений видно, что правила обучения ограниченной машины Больцмана минимизируют разницу между оригинальными данными и данными, генерируемыми моделью. Генерируемые моделью данные получаются при помощи сэмплирования Гиббса.

Обучение нейронной сети глубокого доверия происходит на основе «каждого» алгоритма послыонного обучения (greedy layer-wise algorithm): В соответствии с ним вначале обучается первый слой сети как RBM-машина. Для этого входные данные поступают на видимый слой нейронных элементов и используя CD-к процедуру вычисляются состояния скрытых $p(y|x)$ и видимых нейронов $p(x|y)$. В процессе выполнения данной процедуры (не более 100 эпох) изменяются весовые коэффициенты и пороговые значения RBM-сети, которые тем фиксируются. Затем берется второй слой нейронной сети и конструируется RBM-машина. Входными данными для нее являются данные с предыдущего слоя. Происходит обучение и процесс продолжается для всех слоев нейронной сети, как показано на рис.

15. В результате такого обучения без учителя можно получить подходящую начальную инициализацию настраиваемых параметров сети глубокого доверия. На заключительном этапе осуществляется точная настройка параметров всей сети при помощи алгоритма обратного распространения ошибки или алгоритма «бодрствования и сна» (wake-sleep algorithm).

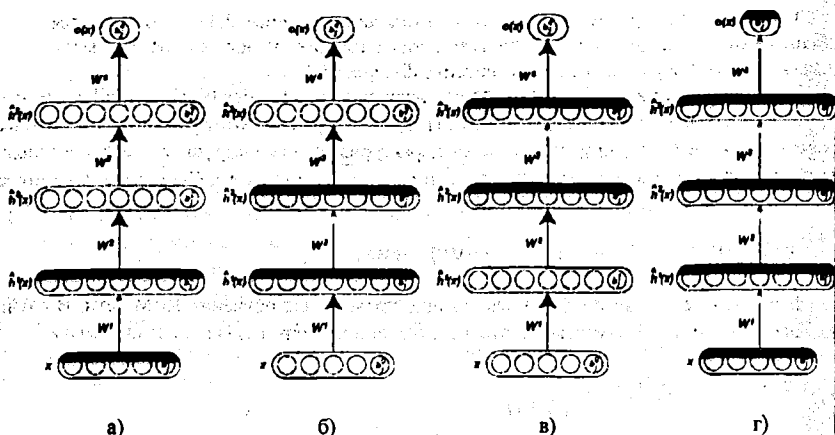


Рисунок 3.4.15 – “Жадный” алгоритм постоянной обучения: а), б), в) – обучение первого, второго и третьего слоев соответственно; г) – обучение всей сети [Larochelle, H. Exploring Strategies for Training Deep Neural Networks / H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin // Journal of Machine Learning Research. – 2009. – Vol. 10. – p. 1-4]

Теоретически число слоев 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.

Задачи. 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, и т.д.) где нейросетевые подходы дали хорошие результаты.

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

Задача аппроксимации функций, когда по набору экспериментальных данных $\{(x_1, y_1), \dots, (x_n, y_n)\}$, представляющему значения y , неизвестной функции от аргумента x , требуется найти функцию, аппроксимирующую неизвестную и удовлетворяющую некоторым критериям.

Задача важна для сложных систем управления динамическими объектами.

Задача предсказания по набору $\{y(t_1), \dots, y(t_n)\}$ – предсказать поведение системы в момент $t_n + 1$, например, при управлении запасами, при принятии решений.

Задача оптимизации – найти решение, удовлетворяющее ряду ограничений и оптимальное значение целевой функции (например, задача коммивояжера).

Литература

- [1, 5, 7, 10, 15, 16, 17, 22, 25, 27, 28, 29, 31, 33, 34, 40]

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 эксперимента 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. Дайте общую характеристику задач по распознаванию образов и их типов.
7. Как решается задача отнесения рассматриваемого образа к одному из выделенных классов?
8. Какие базисные функции используются в обработке изображений?
9. Сравните функции биологического и формального нейронов.
10. Назовите характерные черты задачи распознавания.
11. Назовите типы задач распознавания их этапы.
12. Какие преимущества даёт применение процессов обучения и самообучения НС при классификации и распознавании образов?

Глава 4. ОБЩИЕ ПОДХОДЫ К МОДЕЛИРОВАНИЮ ПРОЦЕССОВ И ОБЪЕКТОВ

4.1. Математическое и компьютерное моделирование

В последние годы резко возрос интерес к подготовке специалистов в области разработки математических и компьютерных моделей, в частности, для сферы автоматизации проектирования, научных исследований, информационных компьютерных технологий, обучения и т. д. Внезапно для практики потребовалось значительное количество современных специалистов, способных разрабатывать:

– математические и алгоритмические модели научно-технических информационных процессов;

– математические модели организационных и технических процессов;

– базы данных и базы знаний;

– программные системы и комплексы для различных сфер применения;

– системы с использованием сетей ЭВМ и средств связи;

– системы поддержки административного управления;

– системы трехмерного моделирования;

– системы с использованием средств искусственного интеллекта в принятии решений и др.

Компьютерное моделирование – метод решения задачи анализа или синтеза сложной системы на основе использования ее компьютерной модели. Его суть – получение качественных и количественных результатов, стоящих перед исследователем в соответствии с постановкой задачи. Сложность вопроса состоит в том, что степень полезности результатов многом зависит от возможностей информационного наполнения модели, когда часть данных еще не получена по тем или иным причинам (еще не завершены научные исследования, эксперименты, данные закрыты от исследователя и т. д.).

Однако по компьютерной модели проще и удобнее проводить вычислительные эксперименты, часть из которых в натуре иногда слишком дорога или неосуществима вообще (ураган, взрыв, потеря объекта и т. п.). На модели легче проверить реакцию изучаемой системы на изменение ее параметров.

К основным этапам компьютерного моделирования относят:

– постановка задачи (иногда более половины успеха дела);

– определение границ объекта моделирования;

– определение процессов взаимодействия между частями модели;

– формализация элементов модели;

– разработка алгоритма воспроизведения процесса на модели;

– написание программы (подбор подходящей готовой системы и ее адаптация);

– проведение экспериментов и фиксация их результатов;

– анализ и интерпретация итогов экспериментов.

Иногда полезно различать аналитическое и имитационное моделирование. Под аналитическим моделированием понимают разработку модели реального объекта путем описания его средствами математики (на основе подбора различных функций и их параметров, связей между ними и т. д.). Имитационное моделирование обычно сводится к построению значительного количества функций для описания отдельных элементов процесса. Его результатом является многократное воспроизведение взаимодействий между элементами модели с последующим получением результата на основе усредненных показателей изучаемого явления в целом.

При создании моделей на практике необходимо соблюдать ряд наиболее важных принципов:

– гарантия осуществимости реализации модели с позиции достаточности информации для ее функционирования;

– возможность достижения поставленной цели за заданное время;

– допустимость модификации модели по мере уточнения информации и результатов;

– использование максимально возможного набора стандартных средств, функций, программ.

Математические модели всего объекта и его частей выражают некоторые его существенные черты на языке уравнений и формул. Современные компьютеры при математическом моделировании используются не только для численных расчетов, но и для построения графиков, воспроизведения звуков и аналитических расчетов. Например, Mathcad и другие аналогичные системы являются хорошими готовыми средствами для таких целей. Эти новые возможности открывают пути к более широкому пониманию и использованию современных компьютерных технологий. В частности, использование графики и речи позволяет воспроизвести более сложные процессы. Например, имитацию работы технических устройств с участием человека, генерацию моделей на основе интеллектуальных банков данных.

Наиболее широкое продвижение математического моделирования нашло себя в систематизированного проектирования (САПР) в области радиоэлектроники, так как эта область системы в значительной мере влияют на технический прогресс в развитии вычислительной техники. Для этих целей было сделано большое количество разработок в создании математического обеспечения процессов проектирования в микроэлектронике, языков проектирования схем, специальных баз данных с типовыми решениями в области схемотехники, Д и др. Здесь впервые стало уделяться внимание комплексному решению задачи проектирования и производства радиоэлектронных схем с целью создания единой системы от проектирования микроэлектронных схем до их выпуска на оборудовании с числовым программным управлением, что в современных условиях позволяет в течение нескольких месяцев перейти к модернизации производства выпускаемых радиоэлектронных изделий (например, телефонов) и резко сократить сроки производства новинок. Создаются системы проектирования креативного типа, когда они допускают изменение хода процесса проектирования за использования опыта и интеллекта инженеров. Многие в этом направлении сделано в решении задачи размещения компонентов и трассировки печатных плат и создании монолитных кристаллов для чипов.

Эти работы послужили толчком к развитию компьютерной графики, развитию нейровых методов решения задач и созданию специальных типов компьютеров (нанокомпьютеров, биокомпьютеров, нейрокомпьютеров и др.), а также методов и программных средств 3D моделирования. Компьютерная модель является программной реализацией математической модели (абстрактной знаковой модели), которая частично отражает исходный объект моделирования, исходя из целей его исследования, т. е. математическая модель ограничена отражением только необходимых свойств объекта, которые влияют достаточно адекватно на исследуемый процесс. В этом смысле знаковые модели могут быть довольно схематичными (план цеха в виде чертежа для решения задачи размещения оборудования и т. п.), упрощает моделирование задаваемого процесса. Процесс исследования на модели заключается получением параметров соответствующих заданному критерию с необходимой точностью. Его обеспечение предполагает выполнение некоторых стандартных шагов:

- выделение существенных свойств для объекта моделирования с позиций предположений для решения задач и построение математической модели;
- построение и отладка компьютерной модели;
- оценка соответствия модели для решения класса задач;
- адаптация модели на предмет обеспечения полноты класса решаемых задач.

4.2. Модели оптимизации многомерных функций и принятие решений

Принятие решений в научных, технических, экономических и других областях деятельности часто сводится к поиску оптимальных (рациональных) решений с позиций лица, принимающего решения (ЛПР). Во многих случаях при наличии программных продуктов поддержки принятия решений ЛПР мотивируют правильность своего решения, ссылаясь на обстоятельство. Поэтому особо подчеркнем, что ЭВМ лишь инструмент для принятия решения, за которое несет ответственность пользователь. Существует много методов поиска оптимальных значений функций методами классической высшей математики для непрерывных и

дискретных значений их переменных. Поэтому выбор конкретной системы оптимизации оставлен за пользователем. Мы остановимся лишь на некоторых приемах преодоления трудностей в решении многомерных дискретных и непрерывных задач, когда функция вычислима.

Решение дискретных сложных задач адаптивными приближенными методами опирается на знания из теории оптимизации функций одной переменной. К сложным дискретным задачам обычно относят те, которые в худшем случае точно решаются при полном переборе всех вариантов решений. Их часто называют *NP*-полными. Такие задачи пытаются решать различными методами, включая использование также и нейросетевых моделей.

При принятии решений часто цель управления выражается в виде функции многих переменных (дискретных и/или непрерывных), пределы изменения которых известны, что позволяет строить стратегию выбора рационального или оптимального результата. Основная трудность в решении этих задач состоит в выборе приемлемого набора переменных (x_1, x_2, \dots, x_n) с позиций практики. Универсальный характер носит метод покоординатного спуска. Его применение можно свести к последовательному поиску экстремумов задач от одной переменной x_i , когда все оставшиеся переменные фиксируются на время поиска. Полученное значение x_i делается фиксированным, а x_{i+1} становится следующей переменной и т. д. Остановка наступает, когда за весь цикл фиксируется достигнутый максимум (минимум). Можно делить несколько циклов подряд в зависимости от ограничения по времени. Критерий остановки выбирается ЛПР. Желательно также указывать шаг изменения свой для каждой переменной в течение цикла, корректировать области их изменения и точность решения, т. е. в этом случае окажется полезным диалоговый метод управления по ходу решения проблемы.

По близкой по характеру схеме решается задача градиентным методом оптимизации для непрерывных дифференцируемых функций от многих переменных. Ее также желательно решать в диалоговом режиме. Общность подходов наблюдается в выборе с учетом ограничений исходной точки начала итерационного процесса (интуитивно, случайным механизмом по некоторому закону и т. п.). Аналогично решается и задача выбора длины шага на каждой итерации для рассматриваемой переменной x_i . Описание таких механизмов и их реализации приведены в лабораторной работе № 7.

Остановимся более подробно на реализации случайных механизмов и их роли в решении сложных задач. Когда о поведении функции недостаточно информации, можно с равной вероятностью выбирать любое значение для назначения координаты начальной точки дискретной переменной или выбирать её из заданного отрезка для непрерывной переменной. Такой прием позволяет в какой-то мере при старте из нескольких попыток повысить шансы избежать попадания в локальный экстремум. Вероятностные приемы могут быть и более сложными.

Вероятностное моделирование на ЭВМ обычно опирается на использование генерации случайных чисел равномерно распределенных в диапазоне (0, 1) или (0, а), который со временем встречается в большинстве вычислительных систем в виде стандартной процедуры. На его основе обычно строятся случайные процессы с использованием стандартных процедур для возмущения элементов моделей в виде конкретных случайных законов (экспоненциальный, нормальный и др.).

Второй путь использования генератора случайных чисел состоит в выборе случайных условий или путей продолжения процессов, когда требуется исключить влияние исследуемого на выбор текущего продолжения процесса, например, стартовых точек для различных итерационных процессов вычислений для многомерных функций и т. д.

4.3. Методы защиты информации и электронная подпись документов

Защита информации и создание безбумажных систем документирования стали тесно переплетаться на основе общих средств используемого математического аппарата.

С каждым годом повышается роль средств, обеспечивающих информационную безопасность различных технологий, использующих компьютеры. Ранее информационная безопасность ограничивалась сферами разведки, дипломатии, иногда коммерческих опера-

С приходится обеспечивать информационную безопасность практически для всего населения страны. Это объясняется, прежде всего, выполнением дистанционных денежных 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 разных электронных картотек (медицинских, налоговых, правоохранительных и т. п.). По мере привело к необходимости использовать сложнейшие информационные технологии в виде 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 замков и сейфов, охрану объекта людьми, хитроумные системы на основе лазерной оптики 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 букв в тексте: вместо истинной буквы записывалась другая, отстоящая от нее в алфавите на заданный шаг в закольцованном алфавите. Например, в алфавите {A, B, C, D, E, F, R}, истинное сообщение ARFA, закодированное с шагом 3, будет — DСVD.

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, используемых 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 денонесения 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 Standard, США). Он опирался на открытый для всех алгоритм, 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 $18 \cdot 10^{16}$ комбинаций удалось осуществить взлом контрольной шифровки. В связи с этим на смену DES пришел AES (Advanced Encryption Standard — 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 шло в направлении создания двухключевых систем: их первый ключ публикуется для применения всеми пользователями при расшифровании данных, а для их шифрования имеет секретный второй ключ, который нельзя получить из первого, что сделало сам зашифрованный текст равнозначным подписанному. Суть математических преобразований для этих систем опирается на теорию функций с «лазейкой», когда $y = f(x)$ легко вычисляется, а обратная функция $x = f(y)$ невычислима без знания дополнительной информации (лазейки). Простейшим примером 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, ...) Неизвестно 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 Хеллман предложили использовать функцию $f(x) = a^x \pmod{p}$, где p – очень большое простое число – целое от 1 до $(p - 1)$ число; a – целое число, степени которого в некотором порядке принимают 1, 2, 3, ... $(p - 1)$ по модулю p . Предполагается, что всем пользователям закрытой системы известны a и p (сообщает администратор системы при регистрации). Пользователь случайным образом выбирает целое число x_i ($1 < x_i < (p - 1)$) и держит его в секрете. Пользователь вычисляет $y_i = a^{x_i} \pmod{p}$ 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 $Z_{ij} = (y_j)^{x_i} \pmod{p}$. 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), $y_i = 3^3 \pmod{7} = 6$ (передается в справочник);

$X_j = 5$ (секретное число пользователя j), $y_j = 3^5 \pmod{7} = 5$;

$Z_{ij} = 5^3 \pmod{7} = 125 \pmod{7} = 6$; $Z_{ji} = 6^5 \pmod{7} = 7776 \pmod{7} = 6$.

Предположим, цифра 6 будет означать страницу в каком-то справочнике или книге, содержащая функции для шифрования текстов. (Администратор может выдавать им этот справочник и время от времени его менять; важно, что их ключ (6) остается секретным и администратора сети, так как «6» не передается по открытому каналу).

Решение задач, связанных с признанием электронной подписи, идентификации и аутентификации, тесно связаны с системой шифрования RSA, ХЭШ-функциями и исследованиями по развитию такого рода механизмов.

Алгоритм RSA – компонента ряда криптосистем для цифровой подписи. Алгоритм обеспечивает высокую степень защиты, поскольку в нем в качестве модуля используется произведение двух больших простых целых чисел. В такой системе каждый пользователь имеет свой ключ шифрования и дешифрования. Ключи дешифрования известны всем пользователям системы, а шифрующий ключ держит в секрете его владелец. Криптографические системы типа RSA подходят для реализации цифровой подписи, применяемой в системах электронных платежей и при передаче сообщений с помощью устройств телесвязи.

К недостаткам системы RSA и аналогичных ей относят ее существенно более низкую скорость действия и потребность в более длинных ключах. Наиболее эффективные реализации RSA характеризуются скоростью шифрования порядка нескольких тысяч бит в секунду, тогда как аналогичные реализации более простых систем шифруют несколько миллионов бит в секунду. В связи с этим наиболее целесообразным применением RSA считается организация обмена секретными ключами, необходимыми для обеспечения безопасности в сетях связи.

Основная проблема для системы RSA – генерация соответствующей пары ключей. Для шифрования используется следующая процедура:

1. Выбрать 2 простых числа P и Q ;
2. Найти произведение $N = PQ$ и число $L = (P - 1)(Q - 1)$;
3. Выбрать случайное число D такое, что оно должно быть взаимно простым с числом L . Числа называются взаимно простыми, если они не имеют общего делителя;
4. Определить другое число E такое, что $(ED) \bmod L = 1$;
5. Как только все числа найдены, мы имеем: секретный ключ – E ; открытый ключ – пара чисел D и N .

Тогда при шифровании сообщения его разбивают на блоки M . В результате шифрования для каждого блока M получим число

$$C = (M^E) \bmod N.$$

При дешифровании получаем:

$$M^* = (C^D) \bmod N.$$

Рассмотрим это на примере алфавита из букв $\{Л, О, Я\} = \{1, 2, 3\}$ для передачи текста «ЛОЯ» (или 2, 1, 3). Цифровые обозначения букв или блоков обязательны, так как метод основывается на обработке натуральных чисел.

1. Выберем $P = 3$ и $Q = 11$.
2. Найдем $N = PQ$, $N = 33$; $L = (P - 1)(Q - 1)$; $L = 20$.
3. Выберем D взаимно простое с L : $D = 3$.
4. Выберем E такое, что $(ED) \bmod L = 1$, например, $E = 7$: действительно, $(3 \cdot 7) \bmod 20 = 1$.

5. Тогда открытый ключ $\left. \begin{array}{l} D = 3 \\ N = 33 \end{array} \right\}$, секретный $E = 7$.

Производим шифрование:

$$C_3 = (3^7) \bmod 33 = 9,$$

$$C_1 = (1^7) \bmod 33 = 1,$$

$$C_2 = (2^7) \bmod 33 = 29.$$

Зашифрованный текст получается (29, 1, 9).

Расшифровка:

$$M_1^* = (C_1^D) \bmod N: (29^3) \bmod 33 = 2,$$

$$M_2^* = (C_2^D) \bmod N: (1^3) \bmod 33 = 1,$$

$$M_3^* = (C_3^D) \bmod N: (9^3) \bmod 33 = 3.$$

В результате мы получили исходный текст: 2, 1, 3.

Остается только добавить, что для получения достаточно стойкой шифровки необходимо брать очень большие простые числа.

Выполнение соотношения $(ED) \bmod L = 1$ позволяет использовать этот факт для проверки подлинности подписи без знания секретного ключа E с помощью аппарата Хэш-функций.

В практической работе необходимо идентифицировать автора электронного документа: предприятие не по особенностям подписи и печати (например, по образцам подписей и печей в банковской карточке клиента), а по наличию у него электронного ключа для подписывания документов. В этом случае конкретное число-подпись под данным документом в зашифрованное время, может сделать только законный обладатель ключа (E).

Процедура электронной подписи включает в себя два этапа: первый – подписывание (численные параметры подписи, зависящих от текста конкретного документа, один из которых (E) хранится в секрете); второй – проверка получателем с помощью несекретных параметров (D, N) подлинности сообщения (подписи).

Сообщение шифруется по алгоритму RSA, где E подбирается и известно только отправителю, а D, N знает и получатель. Получатель должен иметь возможность с помощью открытого ключа проверить подлинность сообщения. Для этой цели в сообщение добавляется еще одно число, которое является результатом вычисления хэш-функции $h(T)$, зависящей от текста T .

К хэш-функции предъявляется ряд требований:

- невозможность (или возможность за очень длительное время) найти по значению $h(T)$ само T (т. е. требуется построить практически необратимую функцию);
- для заданного T нельзя найти такое T' , чтобы $h(T) = h(T')$;
- вообще нельзя найти пару различных слов T и T' такую, что $h(T) = h(T')$;
- сообщение T (например, текст договора, платежного поручения и т. п.) по заданной функции сжимается в целое число $m = h(T)$, причем $1 < h(T) < N$.

Число m позволяет с помощью открытого ключа констатировать подлинность документа.

С этой целью автор документа с помощью своего секретного ключа E получает второй параметр подписи $S = (m^E) \bmod N$. Параметры m и s вставляются в текст сообщения на место подписи и печати. Все сообщение по телекоммуникационным каналам передается получателю. Он проверяет правильность цифровых параметров m и S , исходя из знания функции $h(T)$, полученного символического объема зашифрованного сообщения (T_1) и «лазейки» для вычисления $h(T)$, $h(T_1)$.

Проверка параметра S производится путем идентификации условия: $(S^D) = m \bmod N$.

Математически доказано, что результат проверки m и s будет положительным в том случае, когда в их формировании использовался секретный ключ E , соответствующий открытому ключу D . Вероятность расшифровки секретного ключа E по открытым параметрам S, m, D и N считается ничтожно малой из-за затрат времени на решение задачи взлома системы по сравнению со временем полезного действия сообщения.

Покажем в упрощенном варианте проверку подписи сообщения $T = (\text{ОЛЯ})$, с дополнением его параметрами m и S . В качестве функции хеширования $h(T)$ возьмем произведение сумм из двух элементов для каждого шифруемого знака: его кода по ходу текста с простым числом (2,3,5,7,...) по порядку следования. В нашем случае их позиции $O = 1, L = 2, Я = 3$; коды $L = 1, O = 2, Я = 3$, т. е. $P = (2 + 2)(1 + 3)(3 + 5) = 128$, чтобы получить m вычисляем так: $m = 128 \bmod 33 = 29$. Можно убедиться, что эта функция удовлетворяет требованиям не по крайней мере для любого сообщения из трех разных букв. Это обнаруживается при вычислении всех возможных 6 разнобуквенных сообщений T : (ОЛЯ) – 29, (ОЯЛ) – 12, (ЛОЯ) – 21, (ЛЮО) (27), (ЯОЛ) (18), (ЯЛО) (8), т. е. не равных $h(T)$.

Следует заметить, что с ростом длины сообщения может оказаться, что такая функция удовлетворяет требованию, когда два сообщения разной длины имеют одинаковое значение, т. е. $h(T) = h(T')$. Чтобы избежать такого явления можно разбивать сообщения на блоки заранее определенной длины. Поэтому выбор хорошей функции $h(T)$ является очень трудной задачей и ее решением занимаются специалисты, которые разрабатывают стандарты шифрования. В описанном примере ограничимся лишь демонстрацией процедуры признания подписи.

Текст $T = \text{ЛОЯ}$, который получает партнер позволяет проверить подлинность подписи по параметрам m, S и $D = 3$ (известно как открытая часть ключа), s приходит с текстом отправителя ($S = m^E \bmod N = 21^3 \bmod 33 = 21$; $m = 21 < 33$). Проверка подлинности сообщения:

$S^D = m \bmod N$; $21^3 = 21 \bmod 33$, т. е. результат проверки положителен и подпись подлинна. Кроме того, если получатель тоже знает как вычислить функцию хеширования от текста, то по прочитанному тексту он может ее вычислить.

В настоящее время в Республике Беларусь выпущен предварительный стандарт СТБ 134, 101.25-2008 для электронной подписи, в котором алгоритм RSA один из трех рекомендуемых для применения. Предприятия могут пользоваться различными системами шифрования, но в этом случае стороны обязаны подписывать договор об использовании избранного алгоритма подписи для своей корпоративной сети. Например, белорусские банки имеют такого типа договоры

4.4. Защита от несанкционированного доступа

Защита от несанкционированного доступа к информации, циркулирующей или хранящейся в различных частях удаленных друг от друга узлов сети, является одной из самых сложных. Это связано с тем, что необходимо решать не только задачу о допуске удаленного пользователя к разрешенным ему объектам для доступа в рамках его полномочий, но и опрени личности пользователя, чтобы вместо него не работал за терминалом сети или самоподключенным терминалом злоумышленник. Для решения такого рода задач созданы специальные инструменты – сетевые протоколы.

Их составной частью являются элементы для идентификации и аутентификации пользователя. Идентификация пользователя, как правило, сводится к определению его полномочий на доступ к данным и устройствам по фамилии и паролю. Аналогичная задача может решаться и по отношению к удаленному терминалу на предмет его законного подключения к сети.

Аутентификация является более сильной формой опознания личности при совершении ответственных операций, так как в современных системах требуется предъявлять физические характеристики пользователя (палец, ладонь, произношение некоторых из заданных слов и т. п.).

Информационная безопасность вычислительных сетей различного ранга – от локальных (притягивая) 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 защитить информацию достоянием, 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 ключ (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.).

4.5. Безбумажные системы ведения документации

Особую роль в создании таких систем играют стандарты различных уровней (предприятия, отрасли, страны, международные). С их помощью достигается качество функционирования систем и их совместимость с другими системами.

Система стандартов менеджмента качества разработана Техническим комитетом Международной Организации по Стандартизации (ISO, International Organization for Standardization).

Стандарты серии ISO 9000, принятые более чем 90 странами мира в качестве национальных, применимы к любым предприятиям, независимо от их численности, объема выпускаемой продукции.

При создании системы безбумажного документооборота на предприятии важную роль играют программные средства формирования электронной подписи и шифрования документов. Эффективность этих средств по реализации электронной подписи состоит в том, что они должны быть сертифицированы на государственном уровне, так как только в этом случае электронный документ признается судом в качестве эквивалента бумажному. Организация системы электронной почты между всеми структурными подразделениями и доступ к различным базам данных может также базироваться на электронной подписи, системе шифрования и стандартных средствах доступа, аналогичных имеющимся для пользователей Интернета.

Комплексность характера защиты документооборота достигается одновременным применением физических средств (замки, пропуска, сигнализация, видеокамеры), аппаратных средств (борьба с перехватом сообщений, шифрование, обнаружение «жучки» и т. п.), организационные средства (инструкции, приказы и т. п.), наиболее специфическим средством защиты является шифрование, позволяющее сохранить тайну от злоумышленника даже украденную информацию.

Юридической основой для ведения «бесбумажного» документооборота являются законы РФ «Об информации» и «Об электронном документе». Законом устанавливается обязательная структура электронного документа (ЭД), он имеет силу в юридическом плане аналогичную обычным заверенным документам (например, в судах).

Общая часть ЭД должна иметь структуру и реквизиты, установленные законами стандартами для обычных документов (дата создания, утверждение, наименование, сведения от его создателя), а также сведения о защите ЭД и средствах выполнения электронной цифровой подписи (ЭЦП), сведения о технических и программных средствах, необходимых для воспроизведения ЭД и его составе. ЭЦП должна выполнять две функции, что отличает её от обычной подписи. Во-первых, она должна удостоверить информацию об общей (содержательной части) ЭД и выполнять роль печати и подписи. Во-вторых, ЭЦП предназначена для подтверждения подлинности и целостности ЭД, что необычно для обычного бумажного документа. Эта функция пресекает возможность изменить ЭД, не оставив никаких следов вмешательства. Это свойство обеспечивает безупречность выполнения дистанционных платежей в электронной форме.

Проблемой в развитии новых подходов к управлению предприятиями является оценка затрат на реализацию информационных технологий, так как трудно оценивать затраты на платные информационные услуги в различных сетях ЭВМ.

Из зарубежных баз данных можно получать достаточно объективные сведения из отчетов предприятий, описаний патентов, новинках науки, биржевых операциях, электронных изданиях в виде книг, рекламе и т. п.

В собственных базах данных имеется достаточно много полезной информации по различным аспектам деятельности предприятия за ряд лет. Сопоставив информацию из этих баз данных, можно более уверенно строить разумные стратегии поведения в бизнесе.

Литература

[2, 3, 4, 5, 8, 9, 13, 14, 15, 19, 24, 25, 27, 34, 35, 36, 37, 40]

Контрольные вопросы

1. Назовите основные этапы при реализации компьютерного моделирования.
2. В чём различие между аналитическим и имитационным моделированием?
3. От чего зависит характер основных свойств, отражаемых моделью?
4. Назовите общие подходы в оптимизации многомерных целевых функций.
5. В чём выражается общность подходов при оптимизации дискретных и непрерывных многомерных функций?
6. Зачем используются генераторы случайных чисел в процессах оптимизации?
7. В чём выражается общность защиты и подписи документов?
8. Зачем вводится государственная сертификация систем защиты информации?
9. В чём различие между одноключевыми и многоключевыми системами шифрования?
10. Назовите свойства функций с «лазейкой».
11. Как делается проверка подлинности ЭЦП?
12. Назовите функции механизма стеганографии.
13. Перечислите обязательные элементы в структуре электронного документа.
14. Сравните процедуры идентификации и аутентификации по их функциям.

5.1. Основные понятия автоматизированного проектирования

Системный подход к проектированию. Ручная разработка ВС обычно порождала следующие проблемы:

- неадекватная спецификация требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

Перечисленные факторы способствовали появлению программно-технологических средств специального класса – CAD- и CASE-средств, которые частично устранили названные выше проблемы. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь ПО, в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ВС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ВС, включая анализ и формулировку требований, проектирование структурного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ВС.

Проектирование, при котором все проектные решения или их часть получают путем взаимодействия человека и ЭВМ, называют автоматизированным, в отличие от ручного (без использования ЭВМ) или автоматического (без участия человека). Система, реализующая автоматизированное проектирование, называется САПР (CAD System – Computer Aided Design System). В основном используется автоматическое проектирование, а автоматическое проектирование возможно лишь в отдельных случаях для сравнительно несложных объектов.

Проектирование сложных объектов основано на применении принципа системного подхода, суть которого заключается в рассмотрении частей явления или сложной системы с учетом их взаимодействия. Системный подход включает в себя выявление структуры системы, типизацию частей, определение атрибутов, анализа влияния внешней среды. Интерпретация и конкретизация системного подхода имеют место в ряде известных подходов с другими названиями. Таковы структурный, блочно-иерархический, объектно-ориентированный подходы.

При *структурном подходе*, как разновидности системного, требуется синтезировать варианты системы из компонентов (блоков) и оценивать варианты при их частичном переборе с предварительным прогнозированием характеристик компонентов.

Блочно-иерархический подход к проектированию использует идеи композиции сложных описаний объектов и соответственно средств их создания на иерархические уровни и спектры, вводит понятие стиля проектирования (восходящее и нисходящее), устанавливает связь между параметрами соседних иерархических уровней.

Ряд важных структурных принципов, используемых при разработке информационных систем и прежде всего их ПО, выражен в *объектно-ориентированном подходе* к проектированию. Такой подход имеет следующие преимущества в решении проблем управления сложностью и интеграции ПО: вносит в модели приложений большую структурную определенность, распределяя представленные в приложении данные и процедуры между классами объектов; сокращает объем спецификаций, благодаря введению в описания иерархии объектов и отношений наследования между свойствами объектов разных уровней иерархии; уменьшает вероятность искажения данных вследствие ошибочных действий за счет ограничения доступа к определенным категориям данных в объектах. Описание в каждом классе объектов доступных обращений к ним и принятых форматов сообщений облегчает согласование и интеграцию ПО.

Наиболее распространенной методологией проектирования ПО на основе структурного подхода является INFRUN, разработанная фирмой SilverRun. Жизненный цикл ПО декомпозируется на стадии, которые связываются с результатами выполнения основных процессов. Технологии поддерживаются в различных CASE-средствах. Тем не менее, структурный подход не позволяет выделить абстракции и обеспечить ограничение доступа к данным; он также не предоставляет достаточных средств для организации параллелизма и он, как правило, неэффективен в объектных и объектно-ориентированных языках программирования.

Метод потоков данных предполагает рассмотрение программной системы в качестве преобразователя входных потоков в выходные. Метод потоков данных, как и структурный метод, с успехом применялся при решении ряда сложных задач, в частности, в системах информационного обеспечения, где существуют прямые связи между входными и выходными потоками системы, и где не требуется уделять особого внимания быстрдействию.

В основе объектно-ориентированного подхода лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причём классы образуют иерархию. Этот подход отражает топологию новейших языков высокого уровня, таких как Smalltalk, Object Pascal, C++, CLOS и Ada. Основными являются следующие методологии: ОМТ (Object Modelling Technique), которая состоит в построении диаграммы потоков данных для описания выполняемых функций и создания модели классов. RUP+UML (Rational Unified Process + Unified Modelling Language), которая является де-факто стандартом ПО-инженерии, покрывает все этапы жизненного цикла ПО, используя в частности UML.

Для всех подходов к проектированию сложных систем характерны следующие особенности:

1. Структуризация процесса проектирования, выражаемая декомпозицией проектных задач и документации, выделением стадий, этапов, проектных процедур. Эта структуризация является сущностью блочно-иерархического подхода к проектированию.

2. Итерационный характер проектирования.

3. Типизация и унификация проектных решений и средств проектирования.

Таким образом, при системном подходе основными являются:

- построение иерархической структуры систем, организация их проектирования;
- анализ и моделирование систем;
- синтез и оптимизация систем.

Моделирование имеет две четко различимые задачи: создание моделей сложных систем (modelling) и анализ свойств систем на основе исследования их моделей (simulation).

Синтез также подразделяют на две задачи: синтез структуры проектируемых систем (*структурный синтез*) и выбор численных значений параметров элементов систем (*параметрический синтез*). Эти задачи относятся к области принятия решений.

Моделирование и оптимизацию желательно выполнять с учетом статистической природы систем. Детерминированность – лишь частный случай. При проектировании характерны нехватка достоверных данных, неопределенность условий принятия решений. Учет статистического характера данных при моделировании в значительной мере основан на методе статистических испытаний (методе Монте-Карло), а принятие решений – на использовании нечетких множеств, экспертных систем, эволюционных вычислений.

Иерархическая структура системы. При блочно-иерархическом проектировании представление о проектируемой системе расчлняют на *иерархические уровни*. По сути, используется декомпозиционный подход, который основан на разбиении сложной задачи большой размерности на последовательно и (или) параллельно решаемые задачи меньшей размерности, что сокращает требования к используемым вычислительным и временным ресурсам. На верхнем уровне используют наименее детализированное представление, охватывающее только самые общие черты и особенности проектируемой системы. На последующих уровнях степень подробности описания возрастает, при этом рассматривают уже отдельные блоки системы, но с учетом воздействий на каждый из них его соседей.

Можно говорить не только об иерархических уровнях спецификаций, но и об *иерархических уровнях проектирования*, понимая под каждым из них совокупность спецификаций того иерархического уровня совместно с постановками задач, методами получения решений и решениями возникающих проектных задач. Для большинства приложений характерно следующее наиболее крупное выделение уровней:

- *системный уровень*, на котором решают наиболее общие задачи проектирования ВС процессов; результаты проектирования представляют в виде структурных схем, схем разбора оборудования, диаграмм потоков данных и т. п.;
- *макроуровень*, на котором проектируют отдельные устройства, узлы машин и приборов; результаты представляют в виде функциональных, принципиальных и кинематических схем, сборочных чертежей и т. п.;
- *микроуровень*, на котором проектируются отдельные детали и элементы машин и приборов.

В каждом приложении число выделяемых уровней и их наименования могут быть разными. В радиоэлектронике микроуровень часто называют компонентным, макроуровень — элементным, между ними вводят уровень, называемый функционально-логическим. Результаты уровней подразделяют на уровни проектирования ВС и вычислительных сетей.

В зависимости от последовательности решения задач иерархических уровней различают нисходящее («сверху-вниз», от верхних уровней к нижним), восходящее («снизу-вверх», от нижних уровней к верхним) и смешанное проектирование (имеются элементы как восходящего, так и нисходящего стиля проектирования). В большинстве случаев для сложных систем предпочитают нисходящее проектирование. Отметим, однако, что при наличии заранее спроектированных блоков (устройств) можно говорить о смешанном проектировании.

Неопределенность и нечеткость исходных данных при восходящем проектировании (так как еще не спроектированы компоненты) или исходных требований при нисходящем проектировании (поскольку техническое задание имеется на всю систему, а не на ее части) обуславливают необходимость прогнозирования недостающих данных с последующим их уточнением, т. е. последовательного приближения к окончательному решению (*итерационность* проектирования).

Наряду с декомпозицией описаний на иерархические уровни применяют разделение представлений о проектируемых объектах на аспекты.

Аспект описания (страта) — описание системы или ее части с некоторой оговоренной точки зрения, определяемой функциональными, физическими или иного типа отношениями между свойствами и элементами.

Различают аспекты функциональный, информационный, структурный и поведенческий (процессный). *Функциональное* описание относят к функциям системы и чаще всего представляют его функциональными схемами. Информационное описание включает в себя основные понятия предметной области (сущности), словесное пояснение или числовые значения характеристик (атрибутов) используемых объектов, а также описание связей между этими понятиями и характеристиками. Информационные модели можно представлять графически (графы, диаграммы «сущность-отношение»); в виде таблиц или списков. *Структурное* описание относится к морфологии системы, характеризует процессы функционирования (алгоритмы) системы и (или) технологические процессы создания системы. Иногда аспекты описаний связывают с подсистемами, функционирование которых основано на различных физических процессах.

Отметим, что в общем случае выделение страт может быть неоднозначным. Так, помимо указанного подхода, очевидна целесообразность выделения таких аспектов, как *функциональное* (разработка принципов действия, структурных, функциональных, принципиальных схем), *конструкторское* (определение форм и пространственного расположения компонентов изделия), *алгоритмическое* (разработка алгоритмов и ПО) и *технологическое* (разработка технологических процессов) проектирование систем. Примерами страт в случае САПР могут служить также виды ПО автоматизированного проектирования.

5.2. Языки описания вычислительных систем

Структурное и поведенческое описания. *Поведенческая модель* показывает реакцию цифрового устройства на изменение входных сигналов с учетом задержки реакции во времени. Эта модель не содержит детального описания аппаратной реализации устройства. Уровень абстракции зависит от уровня детализации описания поведенческой модели. Например, на высшем уровне абстракции поведенческая модель может описывать процессор, выполняющий абстрактный алгоритм, с точки зрения нагрева корпуса и т. п., а на низшем уровне это может быть модель процессора с детализацией системы команд (множества инструкций и алгоритмов их выполнения). Точность детализации входных и выходных данных поведенческой модели зависит от уровня абстракции модели.

Функциональная модель устройства описывает его функции без определения способа реализации этих функций. Данная модель позволяет определять реакцию системы или ее компонента без учета временного фактора (определяет значение выхода устройства, но не время его установки). Уровень абстракции зависит от степени детализации модели устройства. Уровень детализации входных и выходных сигналов зависит от уровня абстракции модели в целом.

Структурные модели представляют компоненты устройств с точки зрения их иерархической взаимосвязей между ними. Структурная модель может отвечать физической иерархии элементов в описываемом цифровом устройстве. Иерархия, в свою очередь, определяется физической организацией конкретной реализации разрабатываемого проекта. Структурная модель описывает физическую структуру конкретной реализации путем определения компонентов и топологии взаимосвязей. Компоненты структурной модели могут быть описаны на структурном, функциональном или поведенческом уровне. Имитационное моделирование структурной модели цифровых устройств требует наличия поведенческих моделей всех нижних уровней иерархии. Степень детализации аспектов модельного времени, значений объектов данных и функциональности структурной модели зависит от степени детализации моделей компонентов.

Модель производительности – данный тип моделей позволяет моделировать временные аспекты работы устройств, т. е. определять скорость реакции модели устройства или компонента на изменение входного сигнала без вычисления значения выходного сигнала.

Модель интерфейса может содержать детализацию всех аспектов обмена информацией между проектируемым устройством и внешней средой, включая функциональность, временные характеристики, значение данных и т. п. Такая модель не содержит информации о внутренней структуре устройства.

Языки логического управления. В системах логического управления традиционно используются *булевы функции и системы булевых функций*, задаваемые в форме табличности для полностью определенных функций и таблиц решений для неполностью определенных функций. При этом таблицы истинности, описывающие автоматы с памятью, сданы названия кодированных таблиц переходов, или кодированных таблиц переходов и выходов. Применение таблиц истинности ограничивается задачами небольшой размерности. Применение таблиц решений – в основном автоматами без памяти – комбинационными схемами. Табличное задание автоматов с памятью ненаглядно.

Аналитической формой представления булевых функций являются булевы формулы. Система булевых формул, которые позволяют описывать как комбинационные схемы, так и автоматы с памятью большой размерности. Системы булевых формул могут быть изоморфно реализованы лестничными или функциональными схемами. Иногда используются также другие аналитические формы представления булевых функций, например, пороговые, триальные или арифметические. Основным ограничением на применение систем булевых формул для автоматов с памятью является их низкая наглядность.

Широко используются *функциональные схемы* и разного типа *схемы алгоритмов*, *Петри* и *графы операций*, рассмотренные в разделе 1.2. На этапе перехода от автоматического описания к тексту программ целесообразно применять алгоритмические языки.

Язык ПРАЛУ параллельных алгоритмов логического управления. Язык ПРАЛУ можно использовать для описания систем, характеризующихся сложным взаимодействием, хронологичностью и параллелизмом. Он объединяет возможности моделей «если-то» с возможностями сетей Петри и обладает средствами для представления последовательности тех состояний ВС. С помощью языка ПРАЛУ возможно описание временной упорядоченности событий, возникающих при реализации протокола обмена сообщениями, абстрагировавшись от всех деталей, кроме тех, что выражаются причинно-следственными и временными отношениями. Алгоритмы на ПРАЛУ представляются в виде причинно-временных зависимостей между событиями.

Основными операциями языка ПРАЛУ являются операции ожидания и действия. Операция ожидания «- p_i » сводится к ожиданию наступления некоторого события p_i , представленного конъюнкцией логических переменных, и ограничивается проверкой условия его истинности, завершаясь после ее выполнения. Операция действия «- A » приводит к наступлению некоторого события, представленного также конъюнкцией логических переменных, в некотором объекте (каким-то изменением его состояния) и выполняется в течение некоторого промежутка времени после ее инициализации.

Алгоритм управления представляется неупорядоченной совокупностью предложений. Каждое предложение состоит из одной или нескольких одинаково помеченных цепочек « $\mu_i \rightarrow v_i$ »; через μ_i обозначен некоторый линейный алгоритм, составленный из операций языка; μ_i и v_i – начальная и конечная метки, которыми служат подмножества из множества $M = \{1, 2, \dots, m\}$ целых чисел.

Допускается, что $v_i = \emptyset$, что обозначается как «- \rightarrow » и является концом реализации алгоритма.

Порядок выполнения цепочек алгоритма управления в процессе его реализации определяется множеством N запуска, его текущие значения $N_i \subset M$. Среди предложений алгоритма выделяется одно – начальное, его метка заносится в N перед реализацией алгоритма.

В процессе реализации алгоритма управления цепочки запускаются независимо друг от друга. Если в некоторый момент времени для некоторой цепочки « $\mu_i \rightarrow v_i$ » выполняется условие $\mu_i \subset N_i$ и реализуется событие p_i , с ожидания которого начинается цепочка μ_i , то она запускается. При этом N_i заменяется на $N_i \setminus \mu_i$, а после завершения цепочки N_i становится новым $(N_i \setminus \mu_i) \cup v_i$. Предложенный механизм достаточен для отображения альтернативного ветвления и распараллеливания процессов.

Синтаксически параллельный алгоритм характеризуется наличием меток $|\mu_i| > 1$, $|v_i| > 1$. Альтернативное ветвление обеспечивается ограничением $(i \neq j) \wedge (\mu_i \cap \mu_j \neq \emptyset) \rightarrow (p_i \wedge p_j = 0)$.

Язык ПРАЛУ поддерживает иерархическое описание алгоритмов, которое является особенно важным в случае описания сложных систем. Для обеспечения реакции устройства управления на некоторые особые события, происходящие в системе, в язык введена операция ожидания в трех модификациях: «- $*$ », «- $*\gamma$ » и «- $*\gamma'$ », где $\gamma \subseteq M$. Ее действие заключается в прекращении реализации всех или некоторых (из множества v или $\gamma' = N_i \setminus v$) активных цепочек алгоритма. Наряду с булевыми в ПРАЛУ допустимо использование и арифметических переменных, в частности введены операции: задержки «- n » – выдержки n единиц времени; счета событий: «- $(x = n)$ » – присвоения многозначной переменной x натурального значения n ; «- $(x +)$ » и «- $(x -)$ » – единичного положительного и отрицательного «- $(x -)$ » уменьшения значения; «- $(x = n)$ » – ожидания наступления события: значение x равно n .

ГСА и ПРАЛУ-описание равносильны. Операторным вершинам ставится в соответствие действие, цепочкам условных вершин – операции ожидания. Кроме того, помечаются входы вершин ГСА, соединенные с выходами нескольких вершин, и выходы условных вершин, соединенные с выходами операторных вершин, если данные условные вершины не являются начальными. Полученные таким образом метки соответствуют меткам ПРАЛУ-описания.

Автомат, реализующий заданную ГСА, может быть типа Мили или Мура. Рассмотрим для примера интерпретацию ГСА автоматом Мура. Для этого операторные вершины исходной ГСА отождествляются с состояниями автомата, а условия переходов между состояниями определяются последовательностями условных вершин, связывающими соответствующие операторные вершины. Каждый переход полученного автомата в ПРАЛУ-описании представляется цепочкой вида « $\mu: p \rightarrow a \rightarrow \nu$ », где μ и ν — соответственно начальное и конечное состояния перехода, p — условие перехода, a — выходной сигнал.

Фрагмент ГСА, изображенный на рис. 5.2.1, представится частью таблицы задания автомата с абстрактным состоянием в виде табл. 5.2.1. Здесь первую строку можно совместить с остальными, поскольку объединение условий переходов в последних трех строках представляет собой тождественно выполняемое условие, что является типичным для ГСА. Это подтверждает независимость выходных переменных от входных для автомата Мура. Таким образом, рассматриваемый фрагмент примет вид табл. 5.2.2, что соответствует следующему ПРАЛУ-описанию:

1 : $\neg x_1 \rightarrow y_1, y_2 \rightarrow 1$
 $\neg x_1, x_2 \rightarrow y_1, y_2 \rightarrow 3$
 $\neg x_1, \neg x_2 \rightarrow y_1, y_2 \rightarrow 2$

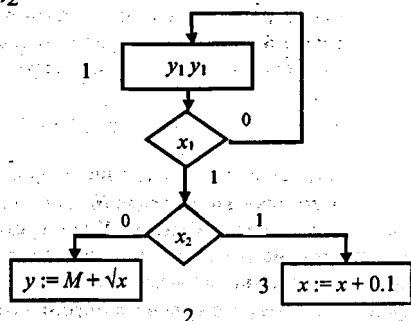


Рисунок 5.2.1 – Фрагмент ГСА

Таблица 5.2.1 – Автомат с абстрактными состояниями

Условие перехода	Начальное состояние	Конечное состояние	Выходной сигнал
-	1	-	y_1, y_2
$\neg x_1$	1	1	-
$x_1 \wedge \neg x_2$	1	2	-
$x_1 \wedge x_2$	1	3	-

Таблица 5.2.2 – Эквивалентное задание автомата с абстрактными состояниями

Условие перехода	Начальное состояние	Конечное состояние	Выходной сигнал
$\neg x_1$	1	1	y_1, y_2
$x_1 \wedge \neg x_2$	1	2	y_1, y_2
$x_1 \wedge x_2$	1	3	y_1, y_2

Графовые модели электронных изделий. Для решения задач автоматизированного проектирования сложных ВС и СБИС используют различные математические модели. Графовые эквиваленты коммутационных схем соединений отдельных частей объекта (му-

и гиперграфы, ультраграфы); графы пересечений для отображения взаимного расположения соединений; неориентированный топологический граф решетки для задания метрического пространства. Использование графотеоретических моделей схем соединений сохраняет ясность и содержательность описываемых объектов и позволяет строить формальные алгоритмы обработки этих моделей, которые легко обрабатываются на ЭВМ.

Любую функциональную или принципиальную схему объекта можно представить собой из множества $E = \{e_1, e_2, \dots, e_n\}$ элементов, множества $\{c_i\}$ выводов (контактов) каждого элемента e_i и множества цепей (эквипотенциальных контактов), соединяющих элементы схемы.

В зависимости от целей проектирования решаемых задач можно использовать различную информацию об исследуемой схеме. Получаемые при этом формальные представления могут иметь разный содержательный смысл. Рассмотрим наиболее употребительные в практике автоматизированного проектирования графовые эквиваленты схем, использующие различные разновидности графов.

Так, при задании схемы в виде мультиграфа $G(X, U)$, множеству X вершин графа ставится в соответствие множество E элементов схемы, а ребра U указывают на наличие соединений между элементами. Данная модель использует минимальные данные о схеме, но имеет широкое практическое значение: мультиграфы коммутационных схем применяются при решении задач декомпозиции и оптимального размещения элементов в пространстве.

Для построения мультиграфа коммутационной схемы необходимо внешние контакты схемы (входные и выходные контакты, соединяющие данную схему с другими) объединить в дополнительный элемент, соответствующий разьему платы или внешним выводам исходной микросхемы и поставить в соответствие каждому элементу схемы вершину мультиграфа. Ребро $u_{ij} = (x_i, x_j)$ мультиграфа схемы $G(X, U)$ будет указывать на наличие связей между вершинами x_i, x_j , соответствующими элементам схемы.

На рис. 5.2.2 приведен условный фрагмент коммутационной схемы, состоящий из пяти элементов $E = \{e_1, e_2, \dots, e_5\}$ и шести цепей $V = \{v_1, v_2, \dots, v_6\}$. Контакты $\{C_i\}$ каждого элемента имеют сквозную нумерацию. Данному фрагменту схемы соответствует математическая модель в виде мультиграфа, представленного на рис. 5.2.3.

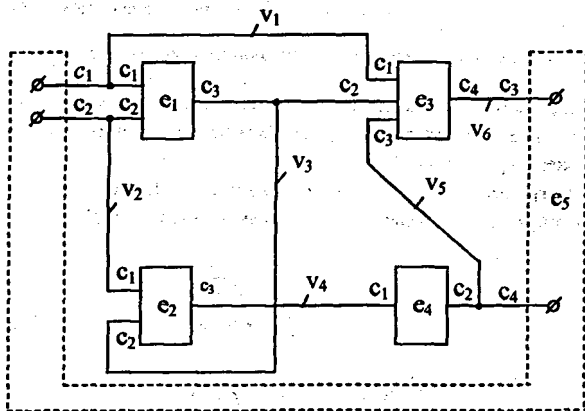


Рисунок 5.2.2 – Фрагмент коммутационной схемы

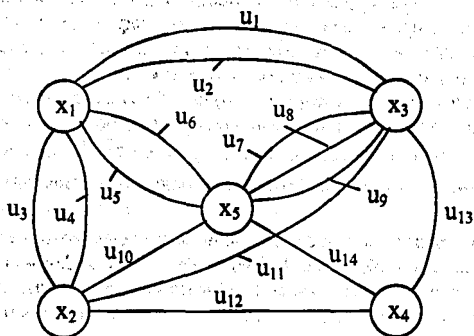


Рисунок 5.2.3 – Мультиграф коммутационной схемы

Для целей автоматизированной обработки используют матричные эквиваленты мультиграфа: матрицу смежности $A = \|a_{ij}\|_{n \times n}$ и матрицу инцидентности $S = \|s_{ij}\|_{n \times x}$, где $|X| = n$:

Временные диаграммы и циклограммы. Достоинство таких форм представления алгоритмов состоит в изображении динамики процессов, а их недостаток – в практической невозможности отражения всех допустимых значений выходных (а тем более внутренних) переменных при всех возможных изменениях значений входных переменных даже для заданной сравнительно небольшой размерности. Поэтому на практике такие диаграммы строят обычно для описания "основного" режима, а алгоритм в целом отражается лишь в программе, которая по указанной причине строится по таким диаграммам во многом неформально.

Автомат задается в виде последовательности S множеств $S(n) = \{t(n), U(n), q(n), V(n)\}$, где n – номер кванта времени, $n = 1, 2, \dots$;

$t(n)$ – время пребывания автомата в текущем состоянии, определяемом квантом времени n ;

$U(n)$ – множество входных сигналов, поступающих к моменту $t_n = \sum_{i=1}^n t(i)$;

$q(n)$ – состояние автомата, в которое он переключается в момент времени t_n ;

$V(n)$ – множество выходных сигналов, вырабатываемое автоматом в течение времени $t(n)$, начиная с момента времени t_n .

Например, диаграмма автомата, показанная на рис. 5.2.4, представляется в виде последовательности, которую для наглядности представим в виде таблицы 5.2.3.

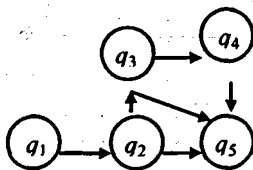


Рисунок 5.2.4 – Диаграмма автомата

Таблица 5.2.3 – Временная диаграмма

n	$t(n)$	$U(n)$	$q(n)$	$V(n)$
1	3	$\wedge u_1$	q_1	-
2	2	u_1	q_2	v_1
3	2	u_1	q_3	v_2
4	2	u_2	q_5	v_1
5	4	-	q_2	-
6	2	u_1	q_3	v_1
7	2	u_1	q_4	-
8	2	-	q_5	v_2
9	2	-	q_2	-

Строка 1 описывает цикл (1,1), строка 2 – линейный участок (1,2), строки 3-5 – цикл (3, 5, 2), строки 6-9 – цикл (2, 3, 4, 5, 2).

Язык *Verilog* был разработан фирмой *Gateway Design Automaton* в 1995 г. как внутренний язык симуляции ВС. К этому времени уже успел получить широкое распространение другой язык высокого уровня для описания принципиальных схем – *VHDL* (*Very high-speed Hardware Description Language*), появившийся еще в 1987 г. Несмотря на похожие названия, *Verilog HDL* и *VHDL* – различные языки. *Verilog* – достаточно простой язык, сходный с языком программирования *C* как по синтаксису, так и по «идеологии». Малое количество «жестких» слов и простота основных конструкций упрощают изучение и позволяют использовать *Verilog* в целях обучения. Но в то же время это эффективный и специализированный язык.

Язык *Verilog* поддерживает четыре уровня абстракции моделей:

- вентиляльный уровень;
- логический уровень;
- поведенческий, или алгоритмический, уровень;
- уровень потоков данных.

Поведенческий, или алгоритмический, уровень – это наивысший допустимый в *Verilog* уровень. На этом уровне модель описывается в виде алгоритма работы устройства, без привязки его аппаратной реализации. Разработка моделей на этом уровне подобна традиционному программированию на языке *C*.

Уровень потоков данных – предполагает построение функциональной модели разрабатываемого устройства на основе предварительно формируемого графа потоков данных. Механизм задержек, включенный в языковой стандарт *Verilog*, позволяет устанавливать время передачи сигнала по ребрам графа и, таким образом, моделировать временные параметры устройства.

Логический уровень – этот уровень, как видно из его названия, предполагает формирование функционально-структурной модели устройства в виде цепей из стандартных логических элементов (например, И, ИЛИ и Исключающее ИЛИ и т. п.). Такие элементы реализованы в виде стандартных модулей в языке *Verilog* и могут быть включены в любую пользовательскую программу.

Вентильный уровень – это наиболее низкий из возможных в языке *Verilog* уровней абстракции. При разработке моделей на этом уровне проектировщик оперирует такими понятиями, как МОП-транзистор, КМОП-транзистор и т. п. Средства синтеза, разработанные для интегральных микросхем *FPGA*-типа, не позволяют синтезировать логические цепи на уровне модели вентиляльного уровня. В то же время такие модели полезны для понимания и исследования физических процессов, протекающих в цифровых цепях.

Отдельно выделяют также уровень регистровых передач (RTL, Register transfer level) представляющий собой комбинацию первых трех уровней абстракции. Verilog-модели, разработанные на уровне регистровых передач, включают только поддерживаемые средствами синтеза операторы языка.

Наиболее распространенная методология проектирования цифровых электронных устройств «сверху-вниз» или «от сложного к простому» предполагает структурную декомпозицию устройства верхнего уровня, или, другими словами, разбиение его на составляющие компоненты, выделяемые по функциональным или топологическим соображениям. Полученные структурные компоненты, в свою очередь, могут быть разделены на более мелкие составляющие либо реализованы операторами языка Verilog на поведенческом (алгоритмическом) уровне. Следует понимать, что «на дне» любой структурной иерархии находятся операторы Verilog, описывающие алгоритм функционирования объекта. Именно последнее определяет основное отличие между поведенческим подходом к проектированию и схемотехническим подходом. При схемотехническом проектировании на нижнем уровне структурной иерархии находится модель устройства, представленная в виде структурной схемы включающей логические вентили, триггеры, регистры, мультиплексоры и другие стандартные цифровые устройства.

Язык VHDL (Very high speed integrated circuits Hardware Description Language) обладает большей универсальностью и может быть использован не только для описания модели цифровых электронных схем, но и для других моделей (например, модели экосистемы). Однако из-за своих расширенных возможностей VHDL проигрывает в эффективности и простоте, т. е. на описание одной и той же конструкции в Verilog потребуются в 3–4 раза меньше символов (ASCII), чем в VHDL. Наряду с языком Verilog, он является базовым языком при разработке аппаратуры современных ВС. Появились пакеты VHDL для аналогового моделирования, моделирования многозначной логики. Кроме того, симуляторы VHDL были гораздо дешевле симуляторов Verilog. Все современные САПР микроэлектроники имеют компиляторы как с Verilog, так и с VHDL. Программист, освоивший VHDL, без особого труда может перейти к программированию на языке Verilog.

VHDL может быть использован на всех этапах разработки электронных систем – алгоритмическом, структурном, регистровых передач (RTL) и потоков данных (dataflow), логическом, аналоговых схем. Вследствие того, что язык легко воспринимается как машиной, так и человеком он может использоваться на этапах проектирования, верификации, синтеза и тестирования аппаратуры также как и для передачи данных о проекте, модификации и сопровождения. VHDL предназначен для описания функции и логической организации цифровой системы. Функция системы определяется, как преобразование значений на входах в значения на выходах. Причем время в этом преобразовании задается явно. Организация системы задается перечнем связанных компонентов.

Проекта VHDL – объединение структуры ВС и алгоритма его функционирования.

Для ВС, описанной на VHDL, обязательно выполнять проверку правильности ее функционирования, например, путем его макетирования. Чтобы определить, правильно ли ВС выполняет заданный алгоритм, достаточно его VHDL-программу запустить на исполнение в симуляторе VHDL. Соответствующие САПР преобразуют VHDL-описание в комплект документации для изготовления работоспособного устройства.

Синтаксический анализ, моделирование и компиляция в логическую схему быстро выявляют ошибки проекта.

Далее, разработанный однажды вычислительный блок может быть использован во многих других проектах. При этом многие структурные и функциональные параметры блока могут быть настраиваемыми (параметры разрядности, объема памяти, элементная база, структура и структура межсоединений).

Проект ВС на VHDL может быть повторно использован через несколько лет. Хорошее техническое решение (например, изобретение), описанное на VHDL, может быть востребованным в течение десятилетий.

Из четырех конструкций верхнего уровня языка VHDL: Entity, архитектуры (architecture body), пакета и конфигурации на любой стадии проектирования обязательно используются две: Entity и архитектура. Entity содержит описание интерфейса между проектируемым устройством и внешним миром. Архитектура содержит описание алгоритма работы проектируемого устройства. Так как для выполнения одной и той же функции могут быть предложены различные алгоритмы, а кроме того, один алгоритм может быть записан различными способами, одному Entity может соответствовать несколько архитектур.

Стили описания архитектур. При описании архитектуры средствами языка VHDL могут использоваться 3 стиля, которые обеспечиваются разными конструкциями языка:

- 1) поведенческий;
- 2) потоковый;
- 3) структурный.

При поведенческом описании алгоритм записывается последовательными операторами (аналогичные которым имеются в большинстве языков программирования).

При потоковом описании архитектура представляется в виде множества параллельных регистровых операций.

При структурном описании проект представляется в виде множества компонент, каждому из которых соответствует своя пара Entity-архитектура. При этом должны быть указаны все связи между компонентами, т. е. для любого входа любого компонента (Entity) должно быть указано, с какого выхода какого компонента поступает сигнал на этот вход.

При этом для описания архитектуры компонента (проекта нижнего уровня) может использоваться любой из трех вышеперечисленных стилей и их комбинация. Таким образом, может быть построено целое дерево компонент, на нижнем уровне которого используются же известные (библиотечные) элементы (пример такого описания приведен ниже).

Поведенческий стиль описания является основным для пользователей САПР, так как именно он используется для создания VHDL-описания, поступающего на вход САПР.

Рассмотрим следующий алгоритм с параллельным выполнением алгоритмов 2 и 3 (рис. 5.2.5, см. также фрагмент параллельной ГСА, изображенной на рис. 1.1.6).

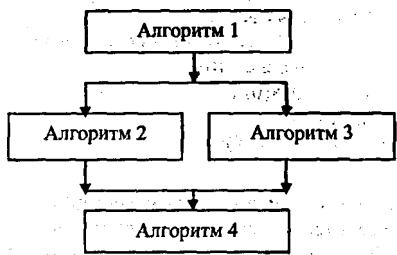


Рисунок 5.2.5 – Параллельный алгоритм

На языке VHDL его можно описать следующим образом:

```
entity PAR is
end PAR;
architecture ar_PAR of PAR is
signal IND1,IND2: bit:=0';
begin
M1: process begin
```

```

    <алгоритм 1>
    IND1<= not IND1;
    <алгоритм 2>
    wait on IND2;
    <алгоритм 4>
    wait on <входные сигналы схемы>;
end process M1;
M2: process (IND1) begin
    <алгоритм 3>
    IND2<= not IND2;
end process M2;
end PAR;

```

Процесс M1 не содержит списка чувствительности, и он стартует первым. После выполнения алгоритма 1 изменится значение сигнала IND1. Так как этот сигнал входит в список чувствительности процесса M2, то после изменения его значения стартует процесс M2. В процессе M1 в это время будет выполняться алгоритм 2. После его выполнения процесс M1 остановится до изменения значения сигнала IND2. Но этот сигнал изменится только после выполнения алгоритма 3 в процессе M2. Таким образом, после выполнения алгоритмов 2 и 3 начнется выполнение алгоритма 4. После выполнения алгоритма 4 работа приостанавливается до изменения значений входных сигналов схемы. Когда изменится какой-либо из входных сигналов, снова стартует процесс M1 и весь цикл повторится.

В качестве иллюстрации различных стилей описания архитектур рассмотрим схему подсчета числа единиц во входном битовом векторе A, имеющем длину, равную 3. Описание ее интерфейса имеет вид:

```

entity ONES_CNT is
    port (A: in bit_vector (0 to 2);
          C: out bit_vector (0 to 1));
end ONES_CNT;

```

Потоковое описание этой схемы использует параллельные операторы назначения сигнала, которые, как и остальные операторы в теле архитектуры, выполняются параллельно:

```

architecture POTOK of ONES_CNT is begin
    C(1)<=( A(1) and A(0) ) or (A(2) and A(0) ) or (A(2) and A(1) );
    C(0)<=( A(2) and not A(1) and not A(0) )
    or (not A(2) and not A(1) and A(0) )
    or (A(2) and A(1) and A(0) )
    or (not A(2) and A(1) and not A(0) );
end POTOK;

```

Поведенческое описание является основным способом задания входной информации для САПР. В этом случае архитектура содержит один процесс или несколько процессов, выполняемых параллельно. В свою очередь каждый процесс содержит операторы, выполняемые последовательно. Поведенческое описание данной схемы может иметь, в частности, следующий вид:

```

architecture ALG1 of ONES_CNT is
begin process (A)
    variable NUM: integer range 0 to 3;
    begin NUM:=0;
    for I in 0 to 2 loop
        if A(I)='1' then NIM:=NUM+1; end if;
    end loop;

```

```

case NUM is
  when 0 => C<=B"00";
  when 1 => C<=B"01";
  when 2 => C<=B"10";
  when 3 => C<=B"11";
end case;
end process;
end ALG1;

```

Естественно существует много различных поведенческих описаний одной и той же мы. В частности, по потоковому описанию можно построить аналогичное поведенческое:

```

architecture ALG2 of ONES_CNT is begin
process (A)
begin
  C(1)<=( A(1) and A(0) ) or (A(2) and A(0) )
  or (A(2) and A(1) );
  C(0)<=( A(2) and not A(1) and not A(0) )
  or (not A(2) and not A(1) and A(0) )
  or (A(2) and A(1) and A(0) )
  or (not A(2) and A(1) and not A(0) );
end process;
end ALG2;

```

Описание автоматов. Описания автоматов на языке VHDL покажем на примере автомата Милли, имеющего три состояния z_1, z_2, z_3 , два входа x_1, x_2 и два выхода y_1, y_2 , заданного основной таблицей переходов и выходов (табл. 5.2.4, где в ячейке (x_i, z_j) указано новое (ячейное) состояние / выход, см. табл. 1.5.2).

Таблица 5.2.4 – Автомат Милли с абстрактными состояниями

Состояние \ Вход	z_1	z_2	z_3
x_1	z_2 / y_1	z_2 / y_2	z_1 / y_1
x_2	z_3 / y_2	z_3 / y_2	z_3 / y_1

В приведенном ниже описании входы, выходы и состояния представлены переменными x, y и z , принимающими целые значения.

```

entity MEALY is
port (x : in integer range 1 to 2; -- входы
      y : out integer range 1 to 2); -- выходы
end MEALY;

architecture AR of MEALY is
constant period : time:=10ns; -- синхронизация
signal clk : bit:=0';
signal z : integer range 1 to 3:=1;
begin
  -- z - состояния
  CLOCK : process begin
    clk <= not clk after period;
  end process CLOCK;
  MAIN : process (clk) begin
    case z is -- переходы

```

```

when 1 => case x is
  when 1 => z <= 2;   y <= 1;
  when 2 => z <= 3;   y <= 2;
end case;
when 2 => case x is
  when 1 => z <= 2;   y <= 2;
  when 2 => z <= 3;   y <= 2;
end case;
when 3 => case x is
  when 1 => z <= 1;   y <= 1;
  when 2 => z <= 3;   y <= 1;
end case;
end case; -- z
end process MAIN;
end AR;

```

5.3. Технологии производства вычислительных систем

Стадии проектирования. Проектирование СБИС базируется на ее иерархической композиции, в основу которой положена концепция абстракции, при этом возможно разделение в двух направлениях: снизу-вверх и сверху-вниз. В зависимости от уровня абстракции СБИС представляется на системном, регистровом, вентилярном и физическом (библиотека элементов кристалла) уровнях. Переход на очередной уровень абстракции выполняется в сле ряда итераций (рис. 5.3.1). Перепроектирование – подход, позволяющий выполнить оптимизацию схем путем ее реконфигурирования.

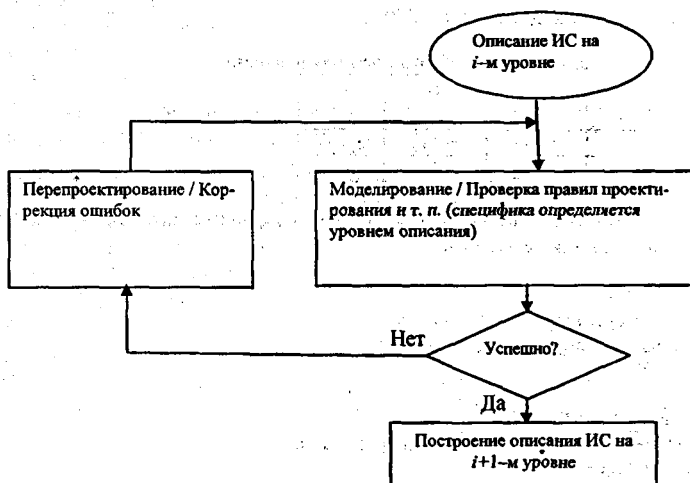


Рисунок 5.3.1 – Схема перепроектирования на i -м уровне абстракции

В традиционной технологии проектирования проект продвигается поэтапно от одного уровня к другому и не происходит возврат на предыдущие уровни. Недостаток этой методологии проектирования в том, что с увеличением сложности проекта увеличивается опасность возникновения ошибок и затрудняется процесс их поиска. Ошибки, обнаруженные в конце той или иной стадии проектирования, ведут к повторному ее выполнению, что в ряде случаев

и неоднократный выпуск прототипов СБИС, приводит к значительному замедлению выполнения проекта и резко повышает его стоимость. Более того, для схем, изготовляемых по субмикронной технологии, такой маршрут вообще не будет работать, поскольку особенности физической реализации должны учитываться уже на логическом уровне проектирования. Поэтому в последнее время используется методология «Specify-Explore-Refine» («определил – опробовал – доделал»). После этапа постановки задачи (спецификации исходных данных), на стадии ее выполнения, происходит оценка различных элементов системы реализации функциональных возможностей в пределах указанных конструкторско-технологических ограничений (КТО). Технические требования модифицируются на стадии проектирования в соответствии с решениями, выполненными на стадии реализации. Маршрут проектирования при этом представляется в виде спиралевидной модели, согласно которой проектирование выполняется одновременно по четырем направлениям: разработка RTL-кода, логический синтез, физический синтез. В процессе работы группы разработчиков обмениваются результатами проектирования. Существенным является то, что после возврата на предыдущие стадии проектирования и корректировки результатов.

Широко используемая технология обратного проектирования СБИС интегрирует в себе различные подходы с тем лишь отличием, что процессу проектирования предшествует восстановление описания ИС на физическом уровне.

Под обратным проектированием понимается процесс анализа системы для идентификации ее компонент и их взаимосвязей и создание описания в другой форме или на более высоком уровне абстракции. Применительно для СБИС этот процесс используется для верификации корректности проекта путем получения изображения топологии и построения транзисторной схемы СБИС. Затем по этой схеме представляют СБИС на вентиляльном и функциональном уровнях. Полученная функциональная модель снова верифицируется и сравнивается с исходным проектом. Обратное проектирование применяется, когда частично или полностью неизвестно поведение, логическая структура, топологическая структура или процесс реализации. В таких случаях может быть интересно проверить неизвестные аспекты и спроектировать лучшую схему. Другая причина заключается в более точной проверке функционирования СБИС и диагностики ее неисправностей. Особенно это актуально сейчас, когда при проектировании используется VHDL и САПР высокого уровня и описание современных СБИС на транзисторном уровне отсутствует.

Методология обратного проектирования представлена на рис. 5.3.2.

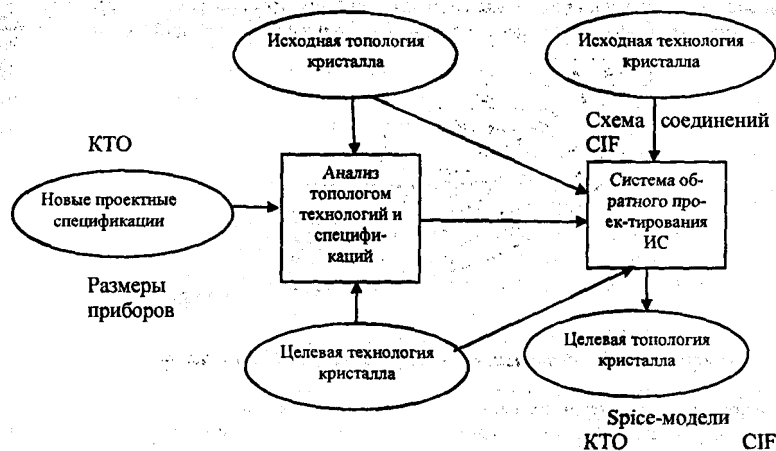


Рисунок 5.3.2 – Информационные потоки при обратном проектировании СБИС

Стадии проектирования разделяют на составные части, называемые проектными процедурами. Примерами проектных процедур может быть моделирование, оптимизация параметров и другие проектные задачи. В свою очередь, проектные процедуры можно разделить на более мелкие компоненты, называемые проектными операциями, например, представление моделирования в графической и текстовой формах.

Типовые проектные процедуры. Создать проект объекта (изделия или процесса) означает выбрать структуру объекта, определить значения всех его параметров и представить результаты в установленной форме. Результаты (проектная документация) могут быть выражены в виде чертежей, схем, пояснительных записок, программ для программно-управляемого оборудования и других документов на бумаге или на машинных носителях информации.

Разработка или выбор структуры объекта есть проектная процедура, называемая *структурным синтезом*, а расчет или выбор значений параметров элементов – процедура *параметрического синтеза*.

Задача структурного синтеза формулируется в системотехнике как *задача принятия решения* (ЗПР). Суть ее заключается в определении цели, множества возможных решений и ограничивающих условий.

Классификация ЗПР осуществляется по ряду признаков. По числу критериев различают задачи одно- и многокритериальные, по степени неопределенности – детерминированные и ЗПР в условиях неопределенности (при наличии в формулировке случайных параметров при неполноте и недостоверности исходной информации).

Реальные задачи проектирования, как правило, являются многокритериальными. Одной из основных проблем постановки многокритериальных задач – установление правил предпочтения вариантов. Способы сведения многокритериальных задач к однокритериальным последующие пути решения изучаются в дисциплинах, посвященных методам оптимизации и математическому программированию.

CASE-технология представляет собой методологию проектирования ВС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ВС и разрабатывать приложения в соответствии с информационными потребностями пользователя. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между модулями системы, динамики поведения системы и архитектуры программных средств.

В результате реальный процесс создания ПО принимал следующий вид (рис. 5.3.3):

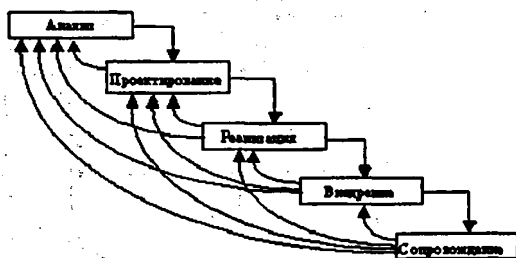


Рисунок 5.3.3 – Реальный процесс разработки ПО по каскадной схеме

Основным недостатком каскадного подхода является существенное запаздывание с получением результатов. Для преодоления перечисленных проблем была предложена спиральная модель жизненного цикла, делающая упор на начальные этапы жизненного цикла: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем

прототипов. Каждый виток спирали соответствует созданию фрагмента или версии, в нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом, углубляются и последовательно детализируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации. Основная проблема спирального цикла – определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом, если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Технология разработки систем на кристалле. Система на кристалле (СНК; *System-Chip, SoC*) представляет собой электронную схему, выполняющую функции целого устройства (например, ЭВМ) и размещенную на одной интегральной схеме. ПО для функционирования СНК не менее важно, чем аппаратное обеспечение. Разработка, как правило, ведется параллельно. Аппаратная часть собирается из стандартных отлаженных блоков, для программной части используются готовые подпрограммы, настройки соответствующих блоков, реализующие необходимые процедуры и функции, (драйвера).

Наиболее трудоемкими и ответственными этапами разработки СНК выступают этапы структурного проектирования и верификации соответствия ВС заданным алгоритмам функционирования. Поэтому эффективность САПРов микросхем и производительность разработчиков, выполняющих проектирование на уровне регистровых передач, постоянно растет приблизительно на 20% в год. Но начиная с середины 90-х годов, производительность разработчиков стала заметно отставать от роста сложности СНК.

Первым направлением улучшения технологии разработки СНК, направленным на уменьшение зазора между ростом производительности проектирования на уровне регистровых передач и ростом сложности СНК, является применение крупных библиотечных вычислительных модулей (*Intellectual Property Cores, IP Cores*). Эти модули должны быть надежно вторгаемыми и настраиваемыми под решаемые задачи в ряде проектов СНК. Повторное применение таких модулей (*IP Core reuse*), которые можно назвать вычислительными заголовками за их функциональную и технологическую адаптируемость, позволяет уменьшить удельные затраты и сроки проектирования СНК.

Второе направление – это разработка САПР совместного проектирования аппаратно-программного обеспечения (*Hardware-Software Codesign*). Архитектура СНК, как правило, включает в себя микропроцессорное ядро с периферийными устройствами в различном сочетании. Обычно процесс разработки ВУ с такой архитектурой состоит из трех последовательных этапов: разработки математического обеспечения микропроцессора, проектирования электрической схемы аппаратуры и стыковки математического обеспечения с аппаратурой. Для ускорения проектирования разрабатывают САПР, которая не только обеспечивает совместное выполнение этих этапов, но и моделирование работы СНК и ее верификацию в комплексе.

Наибольшее ускорение разработки СНК может дать внедрение САПР непосредственно отображения алгоритмов в аппаратуру, т. е. САПР системного проектирования. Например, такая САПР может включать в себя трансляцию программы с языка высокого уровня, например, C++, с автоматическим разделением вычислительных задач между микропроцессорным ядром и спецпроцессорами и другими периферийными устройствами.

В крупных фирмах, долгие годы занимающихся разработкой СБИС, а теперь и СНК, работают большие библиотеки стандартных модулей, как-то: ОЗУ, АЛУ, периферийные устройства. В новых проектах СНК некоторые блоки приходится разрабатывать заново, а остальные – берутся из библиотеки. При этом если модуль неясно описан, не имеет хорошего интерфейса, документации, комментариев, испытательного стенда с надежными тестами, то повторно применяться не будет. Если такой модуль изначально оформлен в виде вычис-

лительной заготовки, то он будет без лишних проблем вставляться в любой новый проект. Более того, лицензию на него можно предлагать другим фирмам-разработчикам СНК. Вычислительные заготовки различаются по степени гибкости своей настройки под условия потребителя как:

- гибкие (описанные языком описания аппаратуры, таком как VHDL, на уровне регистровых передач);

- жесткие (логическая схема, EDIF-файл);

- твердые (маски под определенную технологию, прошивки ПЛИС).

Гибкие заготовки обычно подстраиваются к условиям нового проекта в широких пределах и независимы от его технологии (серия ПЛИС, технология СБИС). Минимизация аппаратных затрат вычислительных заготовок обеспечивает не только уменьшение стоимости СНК, но и уменьшение его энергопотребления, что является важным фактором для портативных и энергонезависимых приложений.

В сегодняшних условиях, чтобы быстрее перейти от идеи к «железу», эффективно провести проектирование новой СНК, необходимо эту СНК «собрать» из имеющихся вычислительных заготовок, а отсутствующие – приобрести на рынке IP-cores, который бурно развивается в последние годы.

Проблема функциональной верификации. Степень интеграции современных СБИС растет экспоненциально. Сегодня переход к субмикронным технологическим нормам позволяет размещать на кристалле практически любой по сложности проект. Тем не менее, имеется сдерживающий фактор – проблема функциональной верификации СБИС. При разработке сложнофункциональных устройств, т. е. систем, включающих не только цифровые, но и аналоговые, смешанные, а также процессорные ядра со встроенным ПО, означенная проблема стоит наиболее остро. По последним данным, примерно половина всего инженерного состава, работающего над крупными проектами, занята функциональной верификацией, а временные затраты на нее в общем цикле проектирования впечатляют еще сильнее – более 50%. Анализ проблемы показал, что технологии верификации проекта на нынешний день заметно отстают от технологий и вычислительных возможностей систем проектирования. Еще более серьезным видится отставание возможностей верификации от технологических возможностей производства СБИС.

В то же время затраты на создание комплекта фотошаблонов для субмикронных СБИС настолько высоки, что их повторное изготовление из-за обнаруженных ошибок зачастую недопустимо как с точки зрения задержки выхода изделия на рынок, так и в связи с ростом конечной стоимости. Поэтому необходим поиск принципиально новых решений, разработку передовых технологий верификации и подход к новым методам анализа проекта.

Основным недостатком традиционного подхода к функциональной верификации является общепринятый алгоритм проектирования. Дело в том, что здесь система сначала разрабатывается и лишь затем – тестируется. При 60-70 % загрузки по времени на верификацию такая последовательность действий недопустима. При увеличении числа транзисторов на кристалле на первое место выходит факт резкого увеличения необходимого числа и длины тестовых векторов как следствие, размера и сложности программно-аппаратных средств тестирования. Параллельно усложняется процесс поиска неисправности, обнаруженной тестом.

Современные средства проектирования должны обеспечивать сквозную верификацию на всех уровнях по преимущественным направлениям. Здесь необходимо использовать технологии имитационного моделирования, аппаратной эмуляции, интегрированной программно-аппаратной верификации и обязательно аналого-цифрового смешанного моделирования. Программные системы создания и верификации проекта должны поддерживать все стандартные языки проектирования, в том числе весь ряд HDL (VHDL, Verilog, VHDL_AMS, Verilog_A). Также должны поддерживаться системные языки (Spice, C, C++, SystemC, SystemVerilog, MATLAB, PSL assertions и др.).

На стадии проектирования, особенно в САПР, предпочтение отдается цифровым моделям, исходя из общности технических средств проектирования и моделирования, информации

онной базы данных на всех этапах проектирования, большого быстродействия и высокой скорости расчетов современных ЭВМ, широких возможностей наглядного, графического представления моделируемых процессов.

Основные методы исследования математических моделей:

- аналитическое исследование;
- имитационное моделирование.

В зависимости от принадлежности к тому или иному иерархическому уровню выделяются модели системного, функционально-логического, макроуровня (сосредоточенного) и микроуровня (распределенного). Кроме того, используются понятия полных моделей и макромоделей, моделей статических и динамических, детерминированных и стохастических, аналоговых и дискретных. Особое место занимают геометрические модели, используемые в методах конструирования.

5.4. Инструментальные средства проектирования

Универсальные программные средства проектирования. В настоящее время практически все современные САПР имеют встроенные функции символьных вычислений. Однако наиболее известными и приспособленными для математических символьных вычислений считаются Maple, MathCad, Mathematica и MatLab. Отметим, что спектр задач, решаемых подобными системами, очень широк:

- проведение математических исследований, требующих вычислений и аналитических выкладок;
- разработка и анализ алгоритмов;
- математическое моделирование и компьютерный эксперимент;
- анализ и обработка данных;
- визуализация, научная и инженерная графика;
- разработка графических и расчетных приложений.

Ниже приведен краткий обзор возможностей систем MathCad и MatLab, рекомендуемые при обучении.

Пакет MathLab. Система MatLab относится к среднему уровню продуктов, предназначенных для символьной математики, но рассчитана на широкое применение в сфере инженерного проектирования. MatLab построена на расширенном представлении и применении матричных операций. Это нашло отражение и в самом названии системы – MATrix LABoratory, т. е. матричная лаборатория.

Несмотря на то, что изначально MatLab предназначалась исключительно для вычислений, в процессе эволюции к ней была подключена библиотека Simulink, позволяющая построить логическую схему сложной системы управления из одних только стандартных блоков, не написав при этом ни строчки кода. После конструирования такой схемы можно детально проанализировать ее работу.

Благодаря тесной интеграции с MATLAB, Simulink имеет непосредственный доступ к широкому диапазону средств проектирования и анализа. Традиционный подход к проектированию систем обычно заключается в создании прототипа, за которым следует всестороннее тестирование и внесение соответствующих изменений. Этот подход требует больших временных и финансовых затрат. Эффективной и общепринятой альтернативой является имитационное моделирование. Simulink – мощный инструмент для моделирования, обеспечивающий быстрое построение и тестирование виртуальных прототипов, и дающий доступ к любому уровню детализации проекта с минимальными усилиями. Используя Simulink для итеративного исправления проекта до построения

В системе MatLab также существуют широкие возможности для программирования. Ее библиотека C Math (компилятор MatLab) является объектной и содержит свыше 300 процедур обработки данных на языке C. Внутри пакета можно использовать как процедуры самой

MatLab, так и стандартные процедуры языка C, что делает этот инструмент мощнейшим инструментом при разработке приложений (используя компилятор C Math, можно встраивать в свои процедуры MatLab в готовые приложения).

C системой MATLAB могут интегрироваться другие популярные математические системы, такие как Mathcad, Maple V и Mathematica. Есть тенденция и к объединению математических систем с современными текстовыми процессорами. Так, новое средство последних версий MATLAB — Notebook — позволяет готовить документы в текстовом процессоре Word 95/97/2000 со вставками в виде документов MATLAB и результатов вычислений, представленных в численном, табличном или графическом виде. Таким образом, становится возможной подготовка «живых» электронных книг, в которых демонстрируемые примеры могут быть оперативно изменены.

В MATLAB задачи расширения системы решаются с помощью специализированных пакетов расширения — наборов инструментов (Toolbox). Многие из них содержат специальные средства для интеграции с другими программами, поддержки объектов ориентированного и визуального программирования, для генерации различных приложений.

Пакет MATHCAD. MATHCAD — универсальный математический пакет, предназначенный для выполнения инженерных и научных расчетов. Математическое обеспечение пакета позволяет решать многие вычислительные задачи с различными с произвольной точностью типами данных (комплексные, векторы, матрицы).

Основное преимущество пакета перед типичными языками программирования — естественный математический язык, на котором формулируется решаемая задача.

Пакет объединяет в себе: редактор математических формул, интерпретатор для вычислений, библиотеку математических функций, процессор символьных преобразований, текстовый редактор, графические средства представления результатов. Пакет MATHCAD отличается к интегрированным пакетам, т. е. позволяет не только произвести вычисления, но и улучшить документ — итоговый отчет с комментариями, формулами, таблицами и графиками. В отличие от издательских систем формулы в MATHCAD работают!

К положительным качествам MATHCAD следует отнести открытость — все приведенное в документе может быть воспроизведено, а интеграция в одном документе исходных данных, метода решения и результатов позволяет сохранить настройки для решения подобных задач.

Аппаратно-ориентированные системы проектирования. Программируемые логические интегральные схемы (ПЛИС) появились в 90-е годы XX века. Основные их производители: Advanced Micro Devices (AMD), Altera, Xilinx, Atmel, Intel, Texas Instruments и др. Главным отличительным свойством ПЛИС является возможность их настройки на выполнение заданных функций самим пользователем. Современные ПЛИС характеризуются низкой стоимостью, высоким быстродействием, значительными функциональными возможностями (одна ПЛИС может заменить несколько сот корпусов традиционной «жесткой логики»), многократностью перепрограммирования, низкой потребляемой мощностью, гибкостью архитектуры и др.

Ниже приведен краткий обзор возможностей пакетов Quartus II и ModelSim, рекомендуемых при обучении САПР ПЛИС.

Пакет Quartus II. Пакет Quartus II представляет собой автоматизированную систему проектирования цифровых устройств на кристаллах ПЛИС фирмы Altera. Он предоставляет пользователю широкие возможности по вводу описаний проекта, логическому синтезу, компиляции проекта, программированию ПЛИС, функциональному и временному моделированию, временному анализу и анализу потребляемой мощности проекта, реализации внутренней системной отладки.

Пакет Quartus II интегрирует в себе большое количество программных модулей, предназначенных для выполнения различных этапов проектирования. Редакторы исходных файлов проекта (графический, текстовый, редактор символов, содержимого модулей памяти, временных диаграмм, конечных автоматов) используют одинаковые подходы и приемы.

похожие оконные формы, применяемые при создании и редактировании исходных с описанием модулей проектируемого устройства.

В одном иерархическом проекте можно сочетать использование различных типов исходных файлов модулей проекта, подбирая наиболее подходящий тип, для каждого. Quartus II допускает использование в проектах отдельных компонентов, созданных в автоматизации проектирования сторонних производителей.

В состав стандартной библиотеки Quartus II входит большое количество базовых элементов, включая мегафункции и макрофункции. Составной частью мегафункций являются ионные устройства, созданные по стандарту библиотеки параметризуемых модулей (M - library of parameterized modules).

Пакет Quartus II допускает использование средств синтеза сторонних производителей ИС. В этом случае, созданные проектировщиком файлы описания проекта на языке VHDL, Verilog, должны быть преобразованы средствами синтеза в файл соединений в формате G или файл Verilog Quartus Mapping File, которые затем могут быть использованы в пакете Quartus II.

Процедура проектирования устройств на ПЛИС включает в себя следующие этапы.

1. Ввод проекта. Проект или его части могут быть описаны в графическом редакторе или дополнительным способом в виде схемы, содержащей отдельные элементы, соединенные между собой цепями связи. Текстовый редактор используется для ввода описаний на языках HDL, Verilog, AHDL (Altera HDL), System Verilog. Соответствующие редакторы есть для создания модулей памяти, временных диаграмм, конечных автоматов.

2. Компиляция проекта. Компиляция представляет собой процесс преобразования описания проекта в его структурную реализацию на выбранном кристалле ПЛИС. При компиляции соответствующий модуль компилятора Quartus II строит базу данных проекта, которая объединяет все файлы описания проекта в единое целое с учетом иерархического представления проекта. После завершения полной компиляции - полностью оптимизированный, смонтированный проект, который используется для создания файлов, применяемых для временного моделирования, временного анализа, анализа потребляемой мощности и программирования кристалла. Модуль анализа и синтеза Quartus II выявляет синтаксические ошибки в проекте и проверяет возможность реализации проекта на выбранном кристалле ПЛИС. Он также преобразует конструкции используемого языка HDL в их аппаратную реализацию на кристалле.

3. Функциональное моделирование проекта (задержки распространения сигналов не читаются). Реализованы два подхода к заданию входных сигналов: формировании воздействий путем задания последовательности входных сигналов в редакторе временных диаграмм; написание специальной тестирующей программы с помощью одного из языков HDL.

4. Временное моделирование на вентиляльном уровне. Осуществляется проверка правильности функционирования проекта после выполнения этапов синтеза, размещения и трассировки.

5. Программирование ПЛИС. Выполняется загрузка конфигурационной информации в программируемый кристалл, посредством чего реализуется пользовательский проект.

6. Тестирование и отладка проекта в составе системы. Проводятся комплексные испытания реализованного проекта, в результате чего принимается решение о соответствии проекта техническому заданию и завершении проектирования. В противном случае принимается решение о доработке проекта.

Пакет ModelSim. Пакет программных средств ModelSim™ корпорации Model Technology (одного из подразделений компании Mentor Graphics) в настоящее время является наиболее распространенной системой HDL-моделирования. В пакете реализована полная поддержка всех основных стандартов языков VHDL и Verilog и их расширений: IEEE VHDL Std 1076-1987, IEEE VHDL Std 1076-1993, IEEE Verilog Std 1364-1995 и др. В пакете обеспечивается поддержка библиотек всех ведущих фирм-изготовителей как программируемых логических интегральных схем (ПЛИС) семейств FPGA (Field Programmable Gate Array) и CPLD

(Complex Programmable Logic Device), так и ASIC (Application-Specific Integrated Circuit) представляющая разработчику широкие возможности сравнения различных платформ выбора оптимальной для реализации проектируемой системы. Открытая архитектура программных средств ModelSim, что как и в Quartus II, позволяет тесную интеграцию с пакетами САПР «третьих» фирм.

Пакет характеризует высокая скорость компиляции и моделирования (полнофункциональных версий). Одним из главных факторов, повышающих производительность, является использование принципа оптимизированной прямой компиляции. В соответствии с этим принципом исходные VHDL- или Verilog-описания компилируются в машинно-независимый объектный код, исполняемый на любой поддерживаемой платформе.

Расширенные отладочные возможности пакета дают возможность полной отладки «смешанных» проектов, которые одновременно содержат модули, написанные на VHDL и Verilog. Они позволяют не только быстро отыскать и идентифицировать ошибки, но и сразу же устранить причины их возникновения. После обнаружения ошибки достаточно перейти из режима отладки в режим редактирования исходного кода, внести соответствующие изменения в текст описания и после сохранения файла выполнить повторную компиляцию данного модуля. Все перечисленные операции производятся в процессе текущего сеанса работы системы моделирования и требуют минимальных временных затрат. Динамическое обновление окон системы обеспечивает возможность быстрого и легкого перемещения по базе данных проекта.

Общая технология моделирования в ModelSim состоит из следующих этапов:

1. Создание и подключение библиотек (рабочей – для хранения результатов компиляции и ресурсов – для хранения редко изменяемых компонент, мегофункций и т. п.).
2. Компиляция исходных текстов проекта и текстов моделирующих программ.
3. Загрузка проекта.
4. Добавление входных воздействий.
5. Наблюдение результатов моделирования.
6. Перезапуск моделирования.

Литература

[2, 3, 8, 11, 12, 14, 15, 16, 25, 40, 43-46]

Контрольные вопросы

1. Укажите специфические свойства языков логического управления.
2. Какие главные отличительные свойства языка ПРАЛУ?
3. В чем заключается преимущества графовых моделей электронных изделий?
4. Почему язык VHDL получил широкое распространение в проектировании вычислительных систем?
5. Охарактеризуйте, в чем выражается полезность использования систем MATLAB и MATCAD.
6. Как проверяется правильность проекта в Quartus II?
7. Какие главные отличительные особенности пакета ModelSim?

Глава 6. РЕАЛИЗАЦИЯ СЕТЕВЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

6.1. Основные понятия об интеллектуальных информационных технологиях

Понятие технологии имеет своими истоками материальное производство. Поэтому термином до появления ЭВМ обычно определяли как науку о способах воздействия на сырье, материалы и полуфабрикаты соответствующими орудиями производства. Основным носителем технологии выступал человек. Он знал, каким способом, средствами и инструментами вызвать процесс преобразования исходных ресурсов или полуфабрикатов. Появление ЭВМ в качестве своеобразного инструмента, способного выступить в качестве носителя технологии, привело к выработке понятия информационной технологии, а в последующем и интеллектуальной информационной технологии. Новое качество в производстве появилось благодаря введению в реализацию технологий комплекса средств для сбора, обработки, хранения, передачи и отображения информации без участия человека или с его незначительным участием. Технологии, реализуемые с применением таких средств, стали называть информационными. Главным элементом в них была дистанционная передача информации и ее обработка, иногда с выработкой простейших управленческих решений на основе типовых стандартных схем.

Развитие самой вычислительной техники привело в последние годы к существенному прогрессу в создании систем с элементами «искусственного» интеллекта, позволяющих решать задачи, считающиеся интеллектуальными (творческими) для человека. В первую очередь это были системы, обладающие способностями к анализу конкретных ситуаций и самостоятельному поиску рациональных решений на основе баз знаний, содержащих не только данные 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. Этот вывод напрашивается в связи с построением информационного общества во всех развитых 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 многих 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 ВС для игры в шахматы демонстрируют результаты на уровне чемпиона мира. Сочетание ученых созданием шахматных программ базируется на уверенности, что полученные здесь результаты приведут к прорыву и в других областях деятельности человека, 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 окончательна 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 быстрого действия 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.

Остановимся более подробно на дистанционных информационных технологиях управления государством, коммерческими операциями и их роли в создании рабочих мест.

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 особую часть (одну или несколько цифровых электронных подписей).

Названные законы открывают путь к использованию «бесбумажных» 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, разработать 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 не реализована 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 выполняется наложение ЭЦП на него. Подписанное зафиксированное сообщение отсылается клиенту, с 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 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 несколько миллионов 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 эти элементы повышают защищенность пользователя от кражи или потери карты, а также от использования ее другими лицами в магазинах и т. п.

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 коммуникации всегда или время от времени вне пределов предприятия. Ведущим фактором здесь является наличие связи работника, его рабочего места с предприятием через электронные средства связи.

Дистанционная работа имеет много разновидностей в зависимости от местонахождения 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 (дистанционный информационный центр, биржа и т. п.);
- рабочее место в определенные дни и часы находится на предприятии, а в остальное 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, сохранением конфиденциальной информации, оплатой труда, возмещением расходов работника по организации рабочего места и платежам за средства коммуникации, оплатой помещения и временного использования компьютера работника, особенностями налогообложения и др.

Кроме того, возникает и ряд правовых проблем: следует регулировать отношения между работником и работодателем, вопросы по налогам за двойное 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. Общение с работодателем идет путем передачи электронных документов, а иногда и по телефону. 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 свободный график работы в течение дня, недели и т. п., она доступна для инвалидов.

Иногда встречается и промежуточный вариант телеработы, когда рабочее место может находиться в квартире работника или на предприятии с регламентацией времени использования того или 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 разрезным характером работы или сервисным обслуживанием по вызову.

Если исходить из материалов исследований Евросоюза, проведенных Фраунгоферским 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, 1000 – к чисто дистанционной работе и 3 000 – к прочим видам работ.

Широко распространилась дистанционная работа в США с охватом около 20 миллионов человек.

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 трудности, 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 элементов и загрузки коммуникаций и оборудования стоимость одного дистанционного рабочего места можно примерно оценить в пределах 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 затрат работника на жилье 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 содержат 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 различных «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 патентованных средствах, позволяющих эффективно решать поисковые задачи в различных сетях и Интернете. В частности, фирма Amazon.com предложила однощелчковый поиск электронной почты для выполнения закупочных операций.

Рекламные заказы в Интернете становятся сложными и дорогостоящими из-за трудностей вывода потребителя на потенциального изготовителя товара или предоставляющую услугу 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 уроков, 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; общие аспекты создания и функционирования дистанционных рабочих мест (экономические, правовые и информационные); использование автоматизированных средств идентификации личности с учетом уникальных физических особенностей человека; создание средств для современной рекламы в сетях с использованием медиасредств 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.

6.2. Нейронные системы и нейрокомпьютеры в сетях

Причины развития работ по введению интеллектуальных компонентов в высокопроизводительные и обычные вычислительные сети – это в первую очередь обеспечение гибкости сетей 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: воз-

возможность адаптации к решению нужного класса задач методами искусственного интеллекта без больших усилий человека и возможность без разработки алгоритма и программы по настройке системы получить алгоритм и программу в скрытой форме в виде соответствующей нейросети для реализации на нейрокомпьютере.

Нейрокомпьютером (НК) называют ЭВМ (аналоговую или цифровую), основной функциональный блок (центральный процессор), который построен на основе нейронной сети реализует нейросетевые алгоритмы. НК 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 НК выделяют следующие четыре:

1. Решение традиционных задач искусственного интеллекта: распознавание образов, 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 алгоритма на основе НС).

3. Использование НК 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 высокой скорости 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.).

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

Опишем общую структурную схему абстрактного НК (рис. 6.2.1).

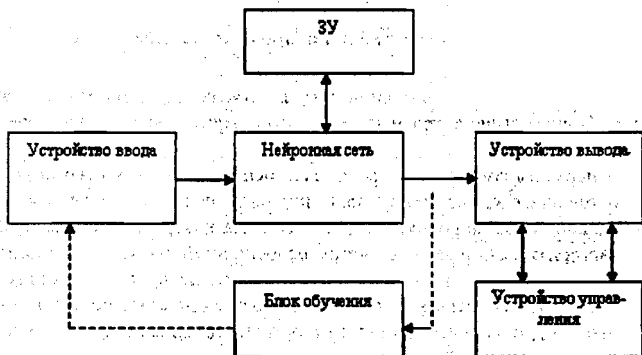


Рисунок 6.2.1 — Структурная схема абстрактного НК

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 ОУ 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.

Системы управления мобильными роботами. 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. 6.2.2.

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

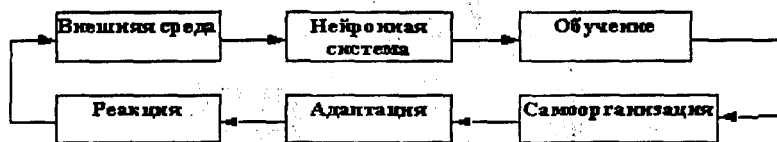


Рисунок 6.2.2 – Взаимодействие нейронной системы с внешней средой

Достаточно задать системе управления координаты конечной точки движения – и автомобиль, к примеру, без участия человека будет двигаться к цели. Такая система разработана в университете Карнеги-Меллона в рамках ALVINN проекта (Autonomous Land Vehicle In a Neural Networks). В ней предполагается, что автомобиль оборудован видеокамерой, которая отображает дорогу с разметкой. Центральным элементом такой системы является трехслойная НС с прямыми связями (рис. 6.2.3).

Входной слой содержит 30×32 нейронных элемента, на которые подается преобразованное от видеокамеры изображение пути. Скрытый слой состоит из пяти, а выходной слой из 3 нейронных элементов. В качестве функции активации используется сигмоидная функция. Активность выходных нейронов характеризует поворот руля в ту или иную сторону. Так, если максимальной активностью обладает центральный нейрон, то это означает движение прямо. Когда наибольшую активность имеет крайний левый нейрон, то это соответствует повороту налево определенное число градусов. Нейронная сеть обучается при управлении автомобилем оператором. При этом оператор управлял автомобилем при движении со скоростью 9,5 км/ч, моделируя различные ситуации. Изображение от видеокамеры использовалось как входные данные, текущее направление руля – как желаемый выход. С целью упрощения получения обучающей выборки используется программное вращение изображения от видеокамеры (рис. 6.2) и соответствующим образом меняется реакция нейронной системы.

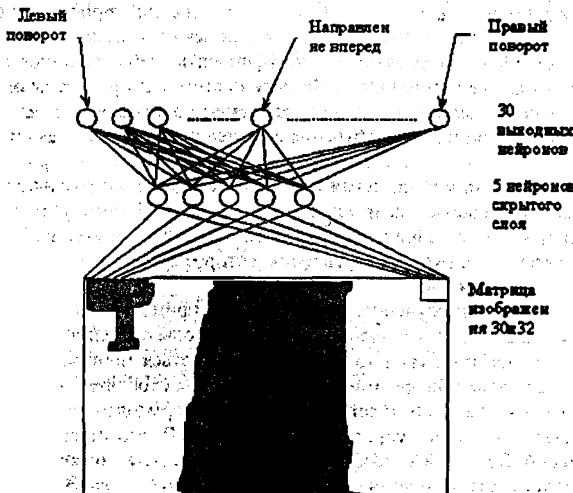


Рисунок 6.2.3 – Отображение изображения дороги на нейронную сеть

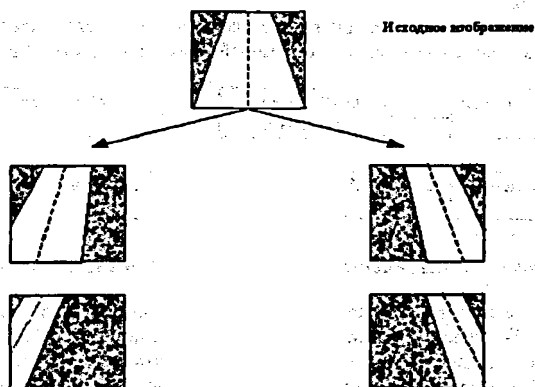


Рисунок 6.2.4 – Вращение исходного изображения от видеокамеры

В результате была создана обучающая выборка, объем которой составил 1200 тренировочных наборов. Обучение нейронной сети проводилось с использованием трех станций «М-4». Время обучения методом обратного распространения ошибки составило 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.

Нейросетевая идентификация. *Нейросетевой подход к идентификации объектов* предполагает разработку НС, алгоритма ее обучения и технологии ее применения. Поиск объектов на основе НС состоит в сканировании изображения окном, размер которого равен размеру обучающего изображения, и определения по отклику НС, является ли изображение внутри окна элементом. Выбор архитектуры НС определяется типом объекта и формой его представления на разных этапах обработки и осуществляется, как правило, путем экспериментальных просчетов и оценок их качества. Учитывая многообразие объектов, используются разнообразные НС. Рассмотрим решение задачи идентификации примере топологических задач, предлагается для решения задачи идентификации совокупность НС, включающая многослойный перцептрон и неокогнитрон, которые апробированы и рекомендуются для использования.

Многослойный перцептрон (рис. 6.2.5) состоит из множества слоев нейронных элементов, причем известно, что достаточно трех слоев для создания сколь угодно сложной решающей граничной поверхности в пространстве обучающих образов. При ее построении количество n нейронов первого слоя определяется в соответствии с количеством информативных признаков (яркостные значения (r, g, b) , семантические дескрипторы и др.). Количество нейронов промежуточных слоев определяется эмпирически. Выходной слой имеет p нейронов, активность которых определяет принадлежность изображения окна к элементу в виде некоторой функции, значение которой сравнивается с некоторым заданным порогом. Каноническим методом обучения такой сети считается алгоритм обратного распространения ошибки. Следует отметить, что данная НС эффективно используется для решения задач прогнозирования.

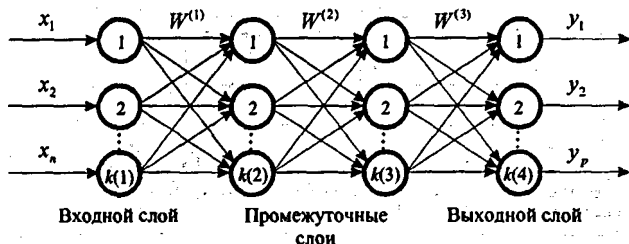


Рисунок 6.2.5 — Архитектура многослойного перцептрона

Неокогнитрон (рис. 6.2.6) позволяет выполнить при поиске элементов сравнение изображения в окне сканирования с эталонным образцом с учетом иерархии признаков, формируемой автоматически в процессе обучения. Слой R является входным. S_1 -слой предназначен для выделения общих признаков, таких, как линейные границы перепадов яркостей различной ориентации. Все подслои этого слоя состоят из нейронов с одинаковыми размерами подгрупп решепторного поля 4×4 нейрона. Обучающие образцы для этих подслоев изображены на рис. 6.2.7.

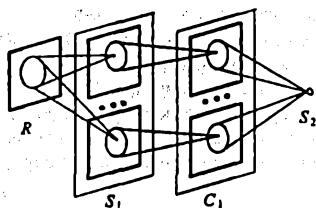


Рисунок 6.2.6 – Используемая архитектура сети



Рисунок 6.2.7 – Обучающие образы S_1 -слоя

S_1 -слой предназначен для обобщения признаков, выделяемых в S_1 -слое, а рецепторные подгруппы его нейронов организованы таким образом, чтобы объединять такие парные признаки, как перепады яркости с темного на светлый или наоборот (такие признаки объединены на рис. 6.2.7 горизонтальной прямоугольной скобкой). В результате получаем четыре подслоя в C_1 -слое. Размер подгрупп рецепторного поля выбран такой, чтобы активность нейронов из этого слоя была инвариантна по отношению к малым сдвигам признаков, выделяемых в предыдущем слое, и составляет поле 2×2 . S_2 -слой предназначен для выделения совокупности признаков, присущих одному конкретному элементу. Он формирует выходное значение сети и состоит из одного нейрона, а размеры подгрупп рецепторного поля этого нейрона совпадают с размерами подслоев C_1 -слоя. Высокая точность идентификации достигается вследствие того, что НС является инвариантной к искажениям формы и яркости изображения распознаваемого объекта. При этом для достижения необходимой точности идентификации требуется меньшее количество обучающих образцов. Типовой алгоритм идентификации (поиска) объектов на изображениях топологических слоев СБИС (структурная схема представлена на рис. 6.2.8).

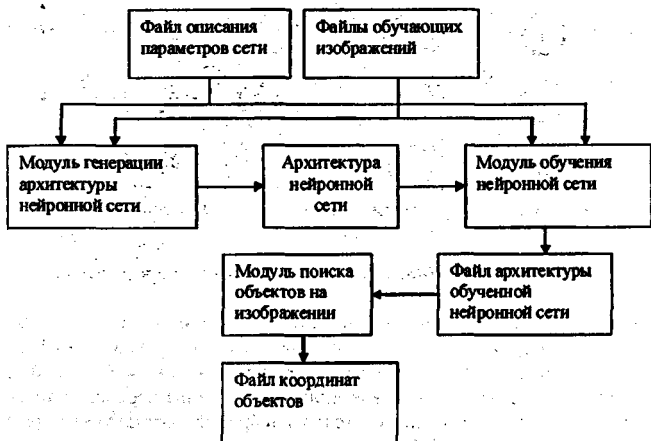


Рисунок 6.2.8 – Схема выделения топологических объектов

Обобщенная структура неоконигитрона для распознавания. Рассмотрим две матрицы

$$W = \begin{pmatrix} w_{1,1} & \dots & w_{1,r} \\ \vdots & \ddots & \vdots \\ w_{x,1} & & w_{x,r} \end{pmatrix} \text{ и } A = \begin{pmatrix} a_{1,1} & \dots & a_{1,r_1} \\ \vdots & \ddots & \vdots \\ a_{x,1} & & a_{x,r_1} \end{pmatrix}, \quad (6.2.1)$$

W – матрица пикселей эталонного изображения;
образ матрицы W , который содержит искажения типа смещения.

Поскольку в матрице A предполагается смещение (изменение) как значений соответствующих пикселей, так и его геометрических координат, то матрица A имеет больший размер, определяемый максимальными величинами смещения по горизонтали и вертикали.

Для определения функции $fd(A, W)$ нечеткого различия двух матриц A и W введем два параметра B и R , которые задают диапазоны нечеткости различия. Параметр B – это диаметр жестких искажений изображения, определяющий максимально допустимое расстояние между двумя пикселями изображения, которые считаются эквивалентными. Параметр R – адгус геометрических искажений, определяющий максимально допустимое смещение пикселей эталона на изображении (образе).

Отметим, что для любых двух соседних пикселей a_{x_1, y_1} , a_{x_2, y_2} и их смещений \bar{v}_{x_1, y_1} , \bar{v}_{x_2, y_2} справедливо $|\bar{v}_{x_1, y_1} - \bar{v}_{x_2, y_2}| \leq \theta$, где $\theta \leq \max(|\bar{v}_{x_1, y_1}|, |\bar{v}_{x_2, y_2}|)$. Данное свойство указывает на однородность геометрических искажений, т. е. на почти одинаковое смещение пикселей.

Построим матрицу

$$D = \begin{pmatrix} d_{1,1} & \dots & d_{1,r} \\ \vdots & \ddots & \vdots \\ d_{x,1} & & d_{x,r} \end{pmatrix},$$

$$d_{x,y} = \begin{cases} 0, & \text{если } \min_{i,j=0,\dots,2R} (a_{x+i,y+j}) \leq w_{x,y} \leq \max_{i,j=0,\dots,2R} (a_{x+i,y+j}); \\ 0, & \text{если } |w_{x,y} - a_{x+R,y+R}| < B; \\ 1 & \text{в остальных случаях.} \end{cases} \quad (6.2.2)$$

Теперь функция нечеткого различия матриц A и W определяется как

$$fd(A, W) = \frac{100}{X \cdot Y} \cdot \sum_{x,y} d_{x,y} \quad (6.2.3)$$

Структура нейрона, реализующего описанное выше правило активации, приведена на рис. 6.2.9, где значения $[\min_k, \max_k]$ определяются так же, как в формуле (6.2.2): $\min_k = \min(a_{k+i})$, $i = 0, \dots, 2R$; $\max_k = \max(a_{k+i})$, $i = 0, \dots, 2R$, $k = 1, \dots, N$, где N – число входов нейрона.

Допустим, что на каждой итерации обучения t поступает обучающий образ A^t , тогда в результате обучения получим последовательность модификаций матрицы пикселей эталонного изображения $\{W^t | t = 0, \dots, T\}$, где T – общее число итераций.

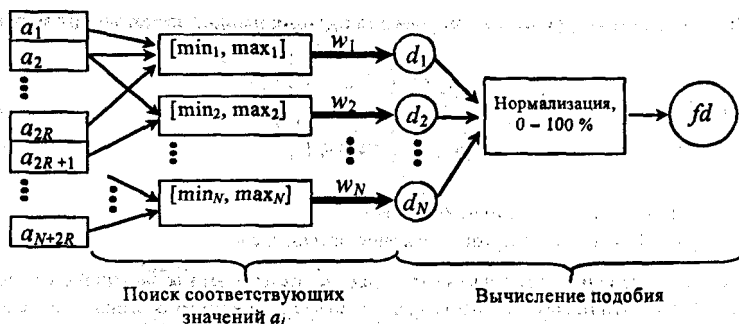


Рисунок 6.2.9 – Общая структура нейрона

Определим функцию нечеткого динамического среднего:

$$w_{x,y}(t+1) = \begin{cases} w_{x,y}(t), & \text{если } \min_{i,j=0..2R} (a_{x+i,y+j}(t)) \leq w_{x,y}(t) \leq \max_{i,j=0..2R} (a_{x+i,y+j}(t)), \\ w_{x,y}(t), & \text{если } |w_{x,y}(t) - a_{x+y,R}(t)| < B, \\ \frac{w_{x,y}(t) \cdot t + a'_{x,y}}{t+1}, & \text{в остальных случаях} \end{cases} \quad (6.2.4)$$

где t – порядковый номер итерации усреднения;

$a'_{x,y} \in \{ a_{x+i,y+j}(t) \mid i, j = 0, \dots, 2R \}$, такое, что $|w_{x,y}(t) - a'_{x,y}(t)|$ принимает минимальное значение.

Общая структура связей предлагаемой НС близка по своей сути и функциям структуре этой неоконитрона с тем лишь отличием, что из-за упрощения структуры нейрона в ней отсутствуют обобщающие S -слои (рис. 6.2.10).

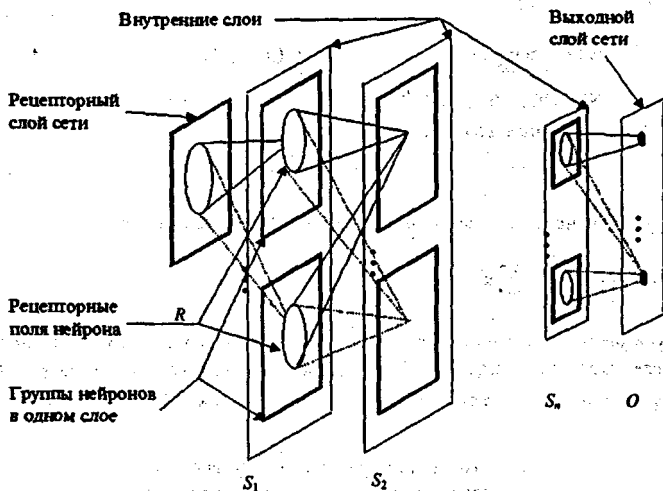


Рисунок 6.2.10 – Структура НС

Рецепторный слой R является входным слоем сети, на который подается обрабатываемый образ. Он преобразует яркости пикселей изображения в значения активностей нейронов. Внутренние слои S_1, \dots, S_{n-1} производят последовательную обработку поступающей информации, а выходной слой сети O генерирует ответ, к какому классу относится входное изображение в виде максимальной активности одного из своих нейронов.

Нейроны, выделяющие одинаковый признак на изображении, объединяются в одну группу. Исходными данными для активации каждого нейрона из отдельной группы является локальная часть данных из предыдущего слоя с соответствующим этому нейрону смещением. На рис. 6.2.11 приведен пример структуры связей нейронов из одной группы в случае одномерных входных данных. Нейроны являются классификаторами признаков определенного уровня иерархии, входные данные для которых поступают с предыдущего слоя и объединяют совокупность признаков предыдущего уровня иерархии. Активность нейрона определяет степень принадлежности рассматриваемой совокупности признаков следующему уровню иерархии.

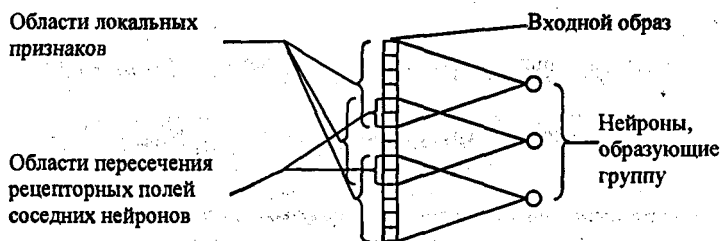


Рисунок 6.2.11 – Схема объединения нейронов в группу

Функционирование нейрона определяется соотношениями (6.2.2)–(6.2.4), в которых переменные R и V являются параметрами. Значение V задано соотношением

$$V = \frac{1}{Y \cdot X} \cdot \sum_{x,y} \left[\frac{1}{L} \cdot \sum_I^L a_{x,y} - \frac{1}{L} \cdot \sum_I^L a_{x,y} \right] \quad (6.2.5)$$

где L – число обучающих изображений размера $X \times Y$. Обозначим через F среднюю частоту смены уровня яркости изображения, для которой соседние точки изображения считаются принадлежащими одному уровню при условии, что абсолютная разница их яркости меньше V . Теперь R зададим как

$$R = [0,3 \times 1 / F], \quad (6.2.6)$$

где $[\times]$ – операция взятия ближайшего целого.

Результаты экспериментальных исследований предложенной НС показали, что функции активации нейрона сильно зависят от выбора параметра V , а анализ результатов обучения – их зависимость от информативности обучающей выборки. Для устранения этого недостатка предлагается следующая функция активации нейрона, которая соответствует модифицированной функции корреляции

$$c = 1 + \frac{n \sum_{x,y} (w_{x,y} a'_{x,y}) - \sum_{x,y} w_{x,y} \sum_{x,y} a'_{x,y}}{\sqrt{\left(n \sum_{x,y} (w_{x,y} a'_{x,y}) - \left(\sum_{x,y} w_{x,y} \right) \left(\sum_{x,y} a'_{x,y} \right) \right)^2}} \quad (6.2.7)$$

где n – количество точек в эталонной матрице; $w_{x,y}$ – значения точек в эталонной матрице; $a'_{x,y}$ – значения соответствующих точек на анализируемом изображении.

$$a'_{x,y} = \begin{cases} 0, & \left[\min_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (a_{x+\pi, y+\gamma}); \max_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (a_{x+\pi, y+\gamma}) \right] \cap \\ & \left[\min_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (w_{x+\pi, y+\gamma}); \max_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (w_{x+\pi, y+\gamma}) \right] \neq \emptyset; \\ a_{x,y}, & \left[\min_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (a_{x+\pi, y+\gamma}); \max_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (a_{x+\pi, y+\gamma}) \right] \cap \\ & \left[\min_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (w_{x+\pi, y+\gamma}); \max_{\substack{\pi x = -R, \dots, R; \\ \gamma y = -R, \dots, R}} (w_{x+\pi, y+\gamma}) \right] = \emptyset. \end{cases}$$

В результате применения функции (6.2.7) получаем значения в интервале $c \in [0; 2]$ где значения $c \in [0; 1)$ говорят об обратной корреляции, значения $c = 1$ – об отсутствии корреляции, а значения $c \in (1; 2]$ – о прямой корреляции. Таким образом, данный подход позволяет отказаться от подбора диаметра яркостных искажений.

Определим значение радиуса геометрических искажений через значение частоты переломов, вычисляемое по формуле

$$F = \frac{F_h + F_v}{2}, \quad (6.2.8)$$

$$\text{где } F_h = \frac{1}{X} \left(1 + \frac{1}{Y} \sum_{x=1, y=1}^{X-1, Y} \delta(g_{x,y}^h, g_{x+1,y}^h) \right), \quad F_v = \frac{1}{Y} \left(1 + \frac{1}{X} \sum_{x=1, y=1}^{X, Y-1} \delta(g_{x,y}^v, g_{x,y+1}^v) \right),$$

$g_{x,y}^h, g_{x,y}^v$ – значения вертикального и горизонтального градиентов в точке (x, y)

$$\delta(a, b) = \begin{cases} 1, & \text{sign}(a) \neq \text{sign}(b); \\ 0 & \text{в остальных случаях.} \end{cases}$$

Значение R зададим, как и ранее, формулой (6.2.6).

Метод генерации обучающего множества. В рассматриваемой постановке задачи распознавания предполагается, что существует набор изображений объектов нескольких типов, причем объекты одного типа имеют разновидности, отличающиеся друг от друга как по яркостным, так и геометрическим характеристикам. Имея такой набор образцов, необходимо разработать метод автоматической генерации обучающих множеств для НС. Для каждого слоя сети можно определить размеры проекции рецепторного поля нейрона на входной слой НС (см. рис. 6.2.12 для случая одномерных входных данных).

Таким образом, для одного обучающего изображения получим K обучающих изображений для некоторого i -го слоя сети: $K = N_{in} - N_i + 1$, где N_{in} – размер входных данных; N_i – размер проекции рецепторного поля i -го слоя сети. В обучающее множество включаем изображения повышенной информативности: данные для i -го слоя сети являются информативными, если выполняется условие: $F_{in} / F_k \leq 0,8$ или $F_k / F_{in} \leq 1,25$.

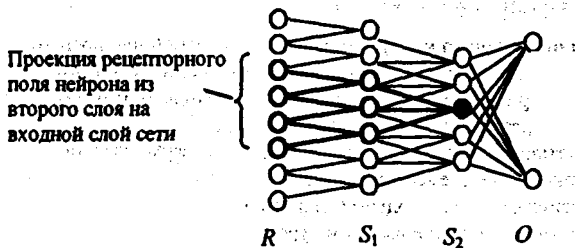


Рисунок 6.2.12 – Метод построения проекции рецепторного поля

Получив обучающее множество для каждого слоя, легко заметить, что задача обучения слоя сводится к задаче кластеризации входных данных слоя, получаемых из обучающего множества. В связи с тем, что входящие в одну группу нейроны имеют одинаковые матрицы весовых коэффициентов, всю группу можно представить в виде одного нейрона, а слой – в виде однослойной НС, в которой каждый нейрон соответствует одной из групп. Для обучения такой сети будем использовать алгоритм кластеризации (k -средних, пикового группирования или типа FOREL в зависимости от свойств обучающего множества), где каждый кластер будет соответствовать одному нейрону. После обучения будем использовать такую НС как прототип для построения искомого слоя в многослойной сети, где количество нейронов в прототипе будет соответствовать количеству групп, а весовые коэффициенты нейронов из прототипа – весовым коэффициентам у нейронов из соответствующих групп.

Генерация обучающего множества для первого слоя НС на примере одного обучающего изображения показана на рис. 6.2.13.

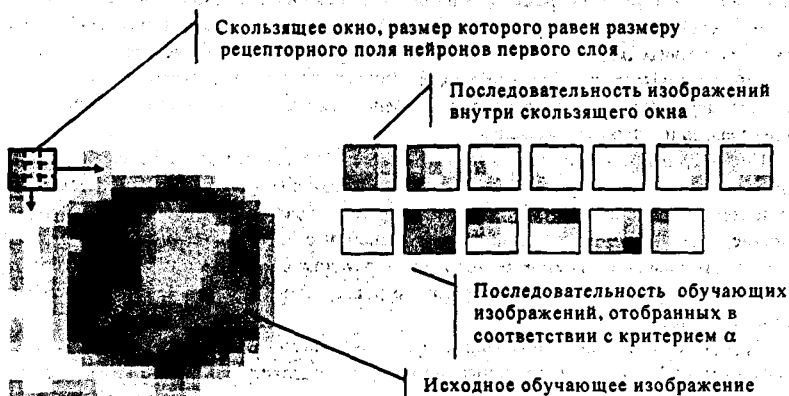


Рисунок 6.2.13 – Генерация обучающего множества слоя

Эксперименты показали, что использование функции (6.2.8) в качестве функции активации нейрона позволяет пропустить этап повышения информативности без ущерба качества обучения. Такой подход сокращает время обучения на время, затрачиваемое для вычисления F_{in} и F_k .

6.3. Системы технического зрения

Методы оптического контроля. Важной проблемой при производстве СБИС является контроль качества на различных технологических этапах. Для этого могут применяться как контактные (электрические), так и бесконтактные методы (оптические, рентгеновские, томографические, ультразвуковые и инфракрасные).

Задача оптического контроля заключается в проверке качества металлизации соединений и контактных окон, элементов топологии поликремниевых и диффузионных областей в ходе изготовления СБИС или подготовки данных и определении дефектов.

Один из шагов на пути к высокому уровню качества – контроль дефектов и обнаружение их источников с применением автоматического оптического контроля. Он позволяет обнаружить до 90 % дефектов, что делает стратегию управления качеством производственного процесса эффективной. По сравнению с визуальным контролем, использование такого контроля имеет следующие преимущества:

- более низкую стоимость обнаружения дефекта, обусловленную ранним детектированием, что дает возможность избегать дефектов на поздних шагах, когда их стоимость выше;
- возможность непрерывного контроля процесса производства;
- сокращение времени контроля.

Оптический контроль базируется на методах морфологических описаний, сравнения с эталоном и контроля проектных норм (конструкторско-технологических ограничений).

Метод морфологических описаний в комбинации с некоторыми базовыми критериями правил проектирования позволяет определять ошибки проектирования. Суть метода заключается в том, что изображение топологии анализируется и представляется как список признаков или образов. Каждый образ классифицируется, и определяется его позиция. Из эталонного изображения топологии СБИС, полученного САПР, формируется список эталонных признаков. Далее список признаков анализируемого образца сравнивается с эталонным списком, и любые различия фиксируются как ошибки. Метод ориентирован на специфику типа топологии и ее изображения, для каждого из которых разрабатываются модели и алгоритмы. Он позволяет также определить дефект, который проявляется в виде некоторой неоднородности на изображении топологического слоя, например, при попадании на пластину чужеродных частиц. Для этого применяют различные методы анализа текстуры изображения.

Метод сравнения с эталоном основан на использовании моделей с заранее определенными информативными признаками в качестве эталона (изображений, получаемых из идеальной топологии или из САПР), с которыми непосредственно сравнивается тестовое изображение, и фиксируются различия. При сравнении с эталоном возможно как непосредственное пиксельное сравнение тестового изображения с изображением эталонного образца (вычитание изображений), так и выделение и последующее сравнение информативных признаков элементов.

Суть *метода контроля проектных норм* заключается в проверке элементов топологии на соответствие КТО, детальное описание которых содержится в руководствах и стандартах SEMI, IPC и др. При этом часто используют операторы математической морфологии и алгоритмы, основанные на анализе границы элементов. Для поиска дефектов может применяться кодирование длин краев элементов.

Каждый из названных методов имеет как свои достоинства, так и недостатки. Техника зрения проста и эффективна с точки зрения вычислений, ограничена необходимостью иметь резец, а также требует точного совмещения изображений. Морфологический подход работает большим числом форм и требуется постоянная доработка алгоритма контроля. Применение процесса верификации правил проектирования непосредственно к изображению образов является затратным по времени и требует больших вычислительных мощностей. Кроме того, проверка правил проектирования приводит к большому числу ложных дефектов. Поэтому важно комбинирование различных методов контроля для достижения лучшего результата (рис. 6.3.1).

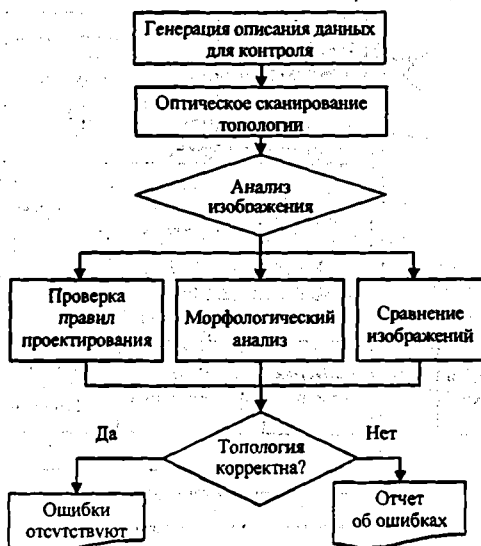


Рисунок 6.3.1 – Схема комбинирования методов оптического контроля

Однако ни один из методов не даст ответа о наличии дефекта, когда на одном изображении одни контактные площадки соединены с другими, а другие являются изолированными. В этом случае оптический контроль дополняется электрическим тестированием. Такой вид контроля получил название автоматического оптического тестирования, при котором анализируется связность объектов топологии.

Общая функциональная схема системы оптического контроля. В настоящее время успехи в машинном зрении позволяют обеспечить автоматизированный процесс производства как электронных компонент, так и СБИС в целом. Типовая система технического зрения (СТЗ) контроля выполняет следующие три основные функции (рис. 6.3.2):

- сбор изображений (оптическое сканирование);
- анализ изображений (детектирование дефектов);
- локализация дефектов (генерация отчета об ошибках).

На основе полученной информации контроллер связи выбирает управляющие сигналы, которые приводят в действие исполнительные механизмы, осуществляющие целенаправленное воздействие на объект. При необходимости обработанная информация об объекте высвечивается на устройстве визуального контроля, записывается на носители информации и выводится на печатающее устройство или иные устройства вывода информации.

В функции блока управления входит управление параметрами обработки, а также синхронизация процессов, выполняющихся в системе.

Следует отметить, что функции отдельных блоков системы могут существенно варьироваться. Это зависит от характера изображений, априорной информации, удачности выбора первичных параметров и класса решающих правил. Кроме того, возможно выполнение одними и теми же блоками нескольких функций.

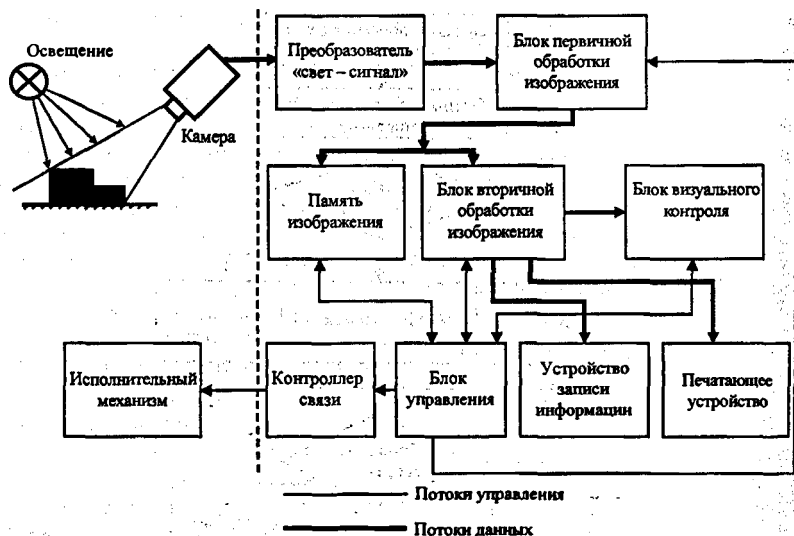


Рисунок 6.3.2 – Функциональная схема СТЗ контроля

Для увеличения скорости обработки используется специальное аппаратное обеспечение. Для эффективной реализации цифровой обработки сигналов хорошо подходят систолические структуры благодаря параллелизму и регулярности потока данных. Для них характерно наличие однотипных модульных локально-связанных процессорных элементов и совмещение в своей архитектуре двух принципов параллельной обработки: конвейерного и матричного. Среди матричных структур, реализующих логические преобразователи информации, широкое применение находят программируемые логические матрицы, программируемые логические интегральные схемы и т. п.

С аппаратным синтезом таких вычислительных систем связан ряд проблем, в частности проблемы межпроцессорных соединений, синхронизации и их модульной организации. Критериями синтеза выступают минимизация потребляемой мощности, задержек и габаритов. В этой связи является актуальным разработка методов и алгоритмы декомпозиции систем булевых функций и покрытия логических схем библиотечными элементами полупроводниковых матричных СБИС, к которым сводится задача оптимизации синтеза по критериям, указанным выше. Совершенствование методов декомпозиции в целом заметно увеличивает возможности автоматизированных систем логического проектирования цифровых устройств.

Надежность системы определяется математическим обеспечением блоков обработки и классификации, а также качеством полученных изображений. Система должна быть тактичной, т. е. адаптировать себя автоматически и достигать постоянно высокой производительности, несмотря на нерегулярность в освещении и условиях фона, приспосабливаться к определенностям в углах, позициях и т. п. Поэтому важна оптимизация параметров освещения, чтобы максимизировать контраст между качественной и дефектной областями.

Таким образом, условия для качественного контроля включают:

- получение качественного изображения, так как плохое изображение дает плохой базис для анализа;
- анализ изображения для получения полезного точного отчета о дефектах, так как избыточная аналитическая логика не идентифицирует все ошибки или будет сверхчувствительной и выдаст слишком много ошибочных дефектов;
- возможность идентифицировать и дать развитое решение о найденных ошибках, потому как плохая локализация дефектов приведет к пропущенной верификации и ошибкам в отчете.

Объектно-ориентированный подход к реализации СТЗ. Функциональная структура процессов обработки информации представляется ориентированным графом, в вершинах которого располагаются процессоры обработки данных, а дуги обеспечивают транспорт данных между ними. Архитектура ПО СТЗ представлена на рис. 6.3.3 с использованием системы обозначений, предложенной Г. Бучем (она соответствует схеме, приведенной на рис. 6.3.2). Здесь показаны два типа связей между объектами (экземплярами классов системы). Классы по своему назначению соответствуют блокам функциональной схемы (для построения диаграмм было использовано средство Rational Rose / C++ Demo Version 4.0.3 фирмы Rational Software Corp.)



Рисунок 6.3.3 – Архитектура ПО СТЗ

Пример клиент-серверных отношений в модели: объект класса «управление отображением» (ответственный за отображение информации в процессе обработки) использует данные, предоставляемые объектами классов «предобработка» (соответствующего блоку предварительной обработки), «память изображения» и «анализ изображения».

Базовые методы обработки изображений представлены на рис. 6.3.4. Конкретные схемы зависят от технологий, реализованных в СТЗ.

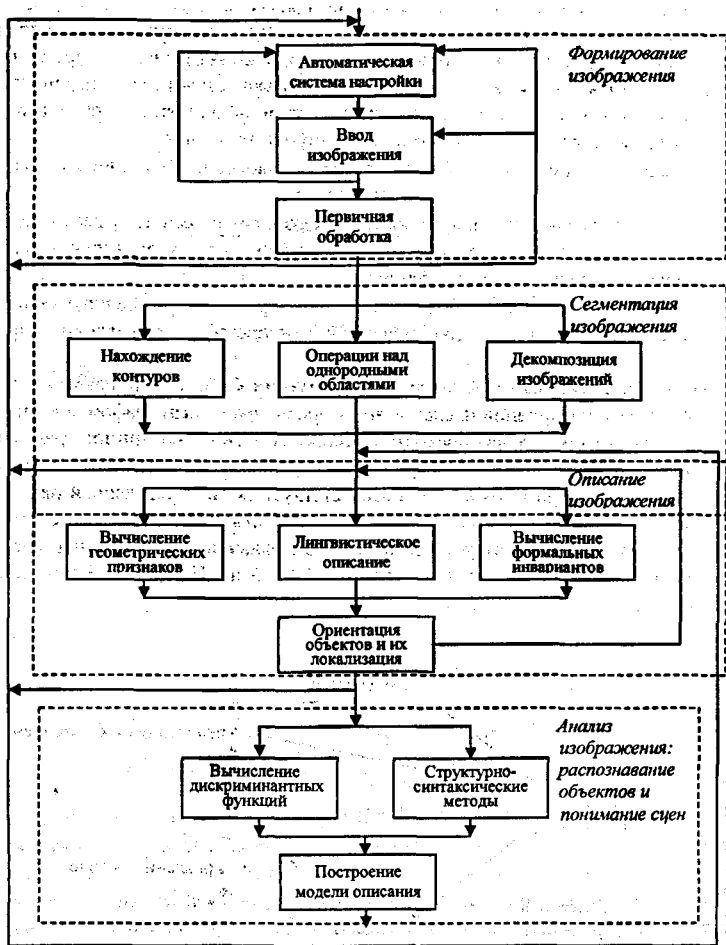


Рисунок 6.3.4 – Структурная схема обработки изображений

Принимая во внимание сложность СТЗ обработки изображений топологии, целесообразно использовать комплексный подход для построения ПО СТЗ: приняв за основу объектно-ориентированное проектирование, использовать метод потоков данных для организации параллельной обработки в рамках многоагентного подхода.

6.4. Мультиагентные системы

На заре развития теории искусственного интеллекта решение задачи сводилось к созданию одной системы, способной её решить. В дальнейшем по мере возрастания потребности из различных областей (наука, техника, экономика, управление государством и крупными

фирмами в мировом масштабе) выявился специфический класс процессов управления, когда элементы системы, владея частичной информацией о глобальной проблеме должны эффективно принимать свои решения с позиций интересов функционирования общего большого объекта. Выяснилась необходимость организации работы многих агентов (программы, людей), обладающих правом на самостоятельные действия при сохранении ответственности за них. Решение такого рода задач вызвало необходимость создания теории мультиагентных систем (раздела теории распределённого искусственного интеллекта, учитывающего возможности самообучения отдельных агентов при принятии решений в рамках своей компетенции). Сложность этих задач возросла и из-за необходимости использования знаний из теорий параллельных вычислений, сетевых компьютерных технологий, защиты информации, мобильной связи и ряда других.

Это привело к развитию новых подходов в создании интеллектуальных систем, в частности, мультиагентных систем прикладного назначения (поддержка принятия решений в условиях недостатка информации из-за её недоступности и/или непредсказуемости, особенности принятия коллективных и личных решений).

Под агентом часто понимается программа (робот или человек), способная действовать в интересах достижения целей, поставленных пользователем.

Индустрия многоагентных систем пока находится в стадии становления. Она постепенно начинает проникновение в следующие области:

- подготовка вариантов информации для отдельных ЛПР или коллективов, принимающих решения;
- поддержка конкурентоспособности производств и завоевание рынков сбыта;
- поиск сервисов и дефицитных продуктов для сложных производств;
- управление роботами и людьми в различных сложных технических объектах.

При полной автоматизации управления сложными объектами интеллектуальными агентами обычно являются программы и роботы. Для мультиагентных систем сформировались некоторые общие требования к агентам:

- автономность (функционирование без прямого вмешательства других агентов);
- взаимодействие с другими агентами на общем коммуникационном языке;
- восприятие внешней среды через различные типы интерфейсов;
- реакция на происходящие изменения в среде;
- проявление инициативы;
- наличие собственной картины окружающего мира;
- способность к интеллектуальному поведению на основе обучения и логических выводов;
- каждый агент способен обмениваться своими знаниями с другими агентами.

Кроме того, при отказе одного или даже нескольких агентов система должна продолжить функционирование, независимые задачи могут выполняться различными агентами. Цели выдвигаются пользователем системы. Замечено, что при требованиях согласования решений между несколькими агентами временные затраты на функционирование системы растут примерно по экспоненциальному закону.

Проиллюстрируем схему организации вычислений на базе многоагентной архитектуры.

Общая архитектура вычислительной платформы показана на рис. 6.4.1. Интерфейс пользователя (блок А) представляет собой средство визуального проектирования и анализа параллельной программы обработки потока данных. Это специализированный графический редактор, использующий графовую модель представления программы. Редактор позволяет создать логический граф программы, определить соответствие вершин графа операциям из библиотек вычислительных гранул (блок В), задать параметры вызова гранул для различных типов обрабатываемых данных.

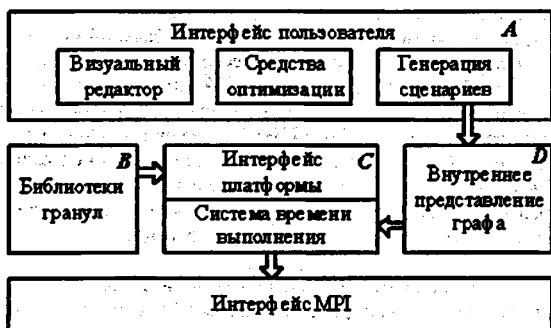


Рисунок 6.4.1 – Архитектура платформы организации параллельных вычислений

Средства оптимизации состоят из алгоритма оптимизации расписания на основе виртуальных ассоциативных сетей и имитационной модели. Имитационная модель используется для оценки вариантов расписания в процессе оптимизации, а также для визуализации расписания. Расписание представляется в виде диаграмм Ганта. Алгоритм оптимизации реализует технологию эволюционной оптимизации и относится к гибридным генетическим алгоритмам, которые имеют преимущества перед эвристическими алгоритмами назначения задач на вычислительные узлы и алгоритмами статической оптимизации вычислений, принимая во внимание специфику обрабатываемого потока (неоднородность данных и задержек при передаче, включая нулевые, изменение состава кластера в процессе обработки и др.).

Созданная в визуальном редакторе программа преобразуется в текстовое представление на языке XML. Это представление содержит информацию для системы времени выполнения, а также отображение графа на топологию ВС. Файл XML при выполнении программы транслируется во внутреннее представление (блок D), позволяющее каждому элементу системы времени выполнения (блок C) получать необходимую информацию во время работы.

Система времени выполнения является многоагентной системой, использующей средства MPI для осуществления координации взаимодействия элементов. Она содержит два типа программных агентов: координатор и исполнитель. Количество агентов-исполнителей при выполнении параллельного приложения равно количеству физических процессоров ВС плюс один агент-координатор. Взаимодействие между агентами осуществляется с помощью передачи сообщений, реализующих простой протокол взаимодействия (рис. 6.4.2).

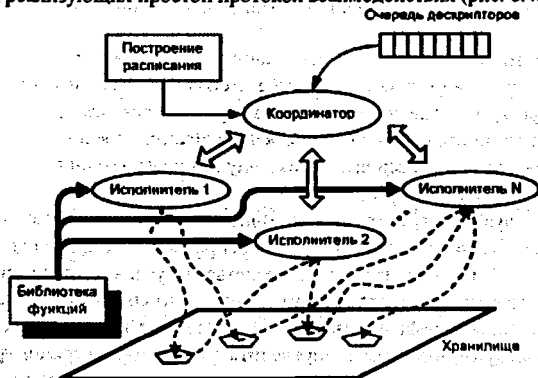


Рисунок 6.4.2 – Архитектура системы времени выполнения

Координатор является главным процессом и управляет выполнением алгоритма, используя очередь дескрипторов обрабатываемых объектов, готовых к выполнению. Основная задача координатора состоит в передаче дескрипторов операций свободным исполнителям в соответствии с отображением операций графа на топологию системы. Кроме того, координатор следит за завершением операций и помещает в очередь новые дескрипторы в соответствии с топологией графа приложения.

Агенты-исполнители являются абстракцией физических процессоров системы. Их основная задача – выполнение вычислительных гранул, которые передаются координатором. Исполнители взаимодействуют с библиотекой гранул, загружая требуемые гранулы и выполняя их на подчиненном процессоре. Взаимодействие между агентами происходит через механизмы разделяемой памяти. После завершения операции исполнитель посылает сообщение координатору, который отмечает момент готовности результатов операции для дальнейшего использования. Синхронизация через координатор гарантирует детерминированное выполнение алгоритма.

Архитектура систем параллельной обработки. Вычислительный кластер может содержать ЭВМ, работающие в различных ОС. Состав вычислительных средств определяет администратор посредством специальных настроек. В состав архитектуры (рис. 6.4.3) входят клиент (на основе платформы Windows) и параллельный сервер (на основе платформы Linux), взаимодействие между которыми осуществляется через файловую систему. Клиент выполняет задачи, обеспечивающие интерфейс системы с пользователем. Он содержит хранилище данных, ПО обработки изображений слоев СБИС и средства удаленного вызова процедур параллельного сервера.

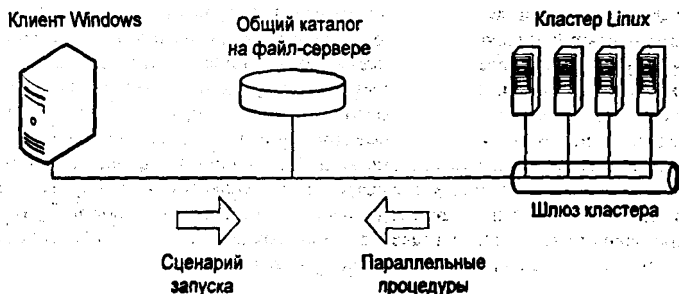


Рисунок 6.4.3 – Архитектура вычислительного комплекса

Клиент имеет доступ к общему каталогу на сервере с помощью средств сервера Samba. Сервер настраивает определенный каталог для общего доступа, и клиент может монтировать его при загрузке Windows. В данный каталог помещаются файлы для обработки (например, кадры), а также сценарии обработки, представленные в виде текстового файла в синтаксисе Linux shell и содержащие команды запуска процедур параллельного сервера с требуемыми параметрами командной строки, указанными в описании процедур обработки. Выполнение сценария со стороны клиента осуществляется посредством протокола SSH (Secure Shell). Реализация этого протокола существует как для платформ Linux, так и для Windows. Со стороны клиента Windows используется программа-клиент Putty, позволяющая настроить доступ к удаленной машине и выполнять на ней команды и сценарии.

Сервер представляет собой параллельный вычислительный кластер на базе ОС Linux, который использует библиотеку поддержки параллельных вычислений MPICH и настраивается на использование SSH без пароля в качестве команды удаленного запуска.

6.5. Транзакции в сетях

Информационные компьютерные технологии в сетях существенно отличаются масштабами, формой организации и средствами управления БД. Наряду с традиционными БД, не используемыми в оперативном режиме для решения задачи, создаются дополнительные хранилища и витрины данных. Второй особенностью является использование в локальных сетях информационных технологий и сервиса Интернет. Появляется возможность реализации дополнительных функций типа электронной коммерции, совместной обработки документов, электронного безбумажного документооборота, телеконференций и т. д.

В третьих появляются специфические алгоритмы поиска и обработки данных, например, обращение к платным хранилищам и услугам.

Специфичен транзакционный подход: на период решения задачи запрещается применение некоторых функций для других проблем и в случае неудачной попытки решить задачу отката системы в исходное состояние.

Основу любых дистанционных операций составляет транзакция. Этот термин используют как в деловом мире электронной торговли, банковских платежах, обращениях с запросом к банкам данных. Однако пользователи термина могут вкладывать в него несколько отличный смысл, исходя из характера выполняемой операции. Например, транзакция это сделка, единица коммуникативного процесса; единичное действие, имеющее два состояния: выполнено (1) и не выполнено (0); минимальная логически осмысленная операция; группа взаимосвязанных операций, которые обладают свойствами атомарности, непротиворечивости, изолированности.

Нами предлагается использовать в дистанционной работе и дистанционных коммерческих операциях термин дистанционная транзакция, включающий обеспечение не только свершение операций (сделки), но и соблюдение ряда требований к ее наиболее ответственным процедурам, касающимся обеспечения конфиденциальности сделки, обеспечения юридической ответственности за передаваемую информацию, невозможность до завершения процедуры всей транзакции начать другую транзакцию, а при срыве выполнения хотя бы одного этапа всю систему затронутых элементов вернуть в исходное состояние. Похожая схема сейчас реализуется при операциях с базами данных в информационных системах. Примером такой же схемы рекомендуется для сделок между хозяйствующими субъектами с требованиями к неделимости, согласованности, надежности и изолированности. Примеры использования транзакций рассмотрим в главе 7 в различных коммерческих и банковских операциях, а также при некоторых видах дистанционной работы.

Литература

[3, 4, 5, 7, 8, 9, 11, 12, 13, 14, 15, 17, 24, 25, 27, 28, 29, 31, 33, 34, 37, 41, 42]

Контрольные вопросы

1. Какое принципиальное отличие имеет интеллектуальная информационная технология от традиционно используемой в промышленности?
2. За счет чего достигается эффективность работы современных шахматных программ?
3. Что является основой для создания дистанционных рабочих мест и средств обучения?
4. В чем состоит отличие документа с электронной подписью от обычного?
5. Какова роль транзакции в дистанционных операциях?
6. Роль отката системы в транзакциях.
7. Причины, побудившие развитие теории мультиагентных систем.
8. Назовите требования к агентам мультиагентных систем.

Глава 7. ПЕРСПЕКТИВЫ РАЗВИТИЯ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

7.1. Влияние нанотехнологий и мобильной связи на развитие информационных технологий

Нанотехнология – область науки и техники, занимающаяся разработкой теоретических и практических методов создания и применения продуктов с заданной атомной структурой путём контролируемого манипулирования отдельными атомами и молекулами. Размеры объектов в нанотехнологиях измеряются в нанометрах (1 нм – миллиардная часть метра) и их диапазон колеблется от 1 до 100 нанометров. Принципиальная особенность нанотехнологической революции – переход от движения в сторону миниатюризации в обратном направлении: складывание больших объектов из молекул и атомов. Развитие нанотехнологий идёт на стыке ряда научных дисциплин и носит надотраслевой характер. Ожидается одно из ярких проявлений в робот- и компьютерной технике благодаря дальнейшей миниатюризации этих устройств и созданию нанотранзисторов.

Нанокomпьютер – вычислительное устройство на основе электронных (механических, биохимических, биологических, квантовых и др.) нанотехнологий с размерами логических элементов в несколько нанометров (1-100). Их основой являются нанотранзисторы с реализацией логических переключаемых функций на базовых элементах разной природы (биологических, электронных и других).

Большинство из них из-за нанозффектов обнаруживают ряд нетрадиционных свойств: энергонезависимое поддержание своего состояния, возможность для одного и того же материала быть сверхпроводником, полупроводником и изолятором и т. д. Биотранзистор настолько мал, что его введение в живую клетку не нарушает режима её функционирования. Это открывает путь к сенсорным измерениям в живых клетках организма и использованию нанороботов в лечебных целях. Биокomпьютер (ДНК-компьютер) функционирует как живой организм и может непосредственно вступать во взаимодействие с внешней живой средой. В области вычислительной техники и нанороботов удалось получить опытные образцы процессоров размером 22 нм. Э.Дрекслер (США) из лабораторий искусственного интеллекта Массачусетского технологического института выдвинул концепцию универсальных молекулярных роботов, работающих по заданной программе по сборке объектов из молекул. Ожидается, что нанороботы будут эффективны при лечении раковых заболеваний и создании новых моделей роботов.

Наиболее продвинутой прикладной базой для построения нанотранзисторов являются углеродные нанотрубки (УНТ). Они характеризуются проводимостью около 1000 раз выше меди и имеют диаметр от 1,2 до 1,5 нм с размерами в длину до 360 нм. В нанодиапазоне электроны демонстрируют одновременно свойства волны и частиц. Заряд можно измерять только порциями (квантами). Половину электрона прибавить нельзя. При приложении небольшого электрического поля вдоль оси нанотрубки с её концов происходит интенсивная эмиссия электронов (полевая эмиссия). Это позволяет создавать полевые транзисторы в качестве переключаемых элементов. Одновременно появляется возможность на несколько порядков увеличить ёмкость и быстродействие памяти.

Очень большое влияние эти достижения оказали на развитие цифровой мобильной связи. Мобильные средства связи сейчас позволяют обеспечить надёжную беспроводную передачу информации в больших объёмах благодаря встроенных в них миниатюрных модемов и комьютеров с возможностями для стационарных ЭВМ. Цифровая связь одновременно в состоянии обеспечить цифровую подпись информации и её секретность.

К мобильным средствам обычно относят устройства, с помощью которых можно поддерживать связь с возможными абонентами, находясь в движении (с отрывом от рабочего стола), которые также могут перемещаться. К таким устройствам обычно относят различные виды мобильных телефонов, пейджеры, радиостанции в автомобиле и т. п.

Мобильная связь порождает дополнительные возможности в управлении предприятием сферы услуг и при выполнении и учёте заказов.

Ремонтные предприятия бытовой техники, таксопарки, мастера, ведущие несколько строительных объектов или один большой объект, могут управлять своими работниками непосредственно на рабочих местах. Это даёт значительный экономический эффект, так как на перемещение работника с одного места выполнения работы на другое требуется меньше затрат и оказываемая услуга или работа в целом выполняется быстрее. Ярким примером являются таксопарки, службы эвакуации сломавшихся машин, предприятия по ремонту бытовой техники на дому и установки различных измерительных приборов в квартирах и т. п. Наличие мобильной связи с менеджерами различного ранга позволяет оперативно принимать решения диспетчером или другим работником, когда на производстве возникает нештатная ситуация. Хорошо зарекомендовала себя мобильная связь в приёме и обработке различных заказов.

Особые преимущества даёт цифровая мобильная связь, так как в этом случае может обеспечиться передача информации непосредственно в компьютерную сеть, а наиболее ответственная информация может подписываться электронной цифровой подписью, например, распоряжения банку или важные решения, влекущие юридическую ответственность.

Сейчас сети мобильной связи переживают очень бурное развитие благодаря расширению международной торговли и ряду специфических коммерческих услуг по передаче информации. Сетевые услуги мобильной связи постепенно преобразуются в «грубопроводы знаний» благодаря их сопряжению с бытовыми приборами, регистрационными и измерительными устройствами со встроенными микропроцессорами и приборами радиосвязи. Благодаря выходу на спутниковую связь появилась возможность оказывать услуги мультимедийной связи и позиционирования подвижных объектов.

В экономике это уже приносит большие экономические эффекты в управлении грузоперевозками и перемещением движущихся объектов по заданным маршрутам. Решение вообще любых логистических задач стало давать значительную прибыль (отслеживание за движением грузов, предсказание времени пересечения границ, контроль расхода горючего и т. д.).

Абоненты цифровых телефонных сетей в огромных масштабах ощутили возможность выполнения платёжных поручений, связанных с различными типами оплаты бытовых и информационных услуг. Возросла эффективность выполнения сетевых коммерческих операций по приобретению бытовой техники и вещей. Последние годы наметился переход к увеличению услуг, связанных со скоростной передачей данных и выполнением аутсорсинга авторитетными специалистами из других организаций (перекладывание рисков фирмы на более подготовленных специалистов).

Наблюдается постепенное расширение ИНТЕРНЕТ-экономики и увеличение масштабов производства и рынков сбыта. Это объясняется ещё и тем, что стоимость использования традиционных (бумажных) технологий при хранении и применении данных стала дороже по сравнению с компьютерными технологиями.

Идет значительный прогресс и в предоставлении и оплате услуг со стороны мобильных операторов. В частности, предприятие, использующее широко в своей деятельности мобильную связь начали ощущать расширение льгот и услуг от мобильных операторов, связанных с введением корпоративных льготных тарифных планов и учёту расходов фирм на мобильную связь (льготные звонки между сотрудниками фирмы, безлимитные тарифы и др.).

С точки зрения перспектив применения мобильной связи следует обратить внимание на более широкое ее использование в сочетании с сетью ИНТЕРНЕТ и увеличение количества операций, связанных с передачей данных, развитием специфической рекламы и организации производств и продаж.

7.2. Интеллектуализация принятия решений в информационных технологиях

Интеллектуализация современных информационных компьютерных технологий становится главным элементом в повышении творческой роли человека в цепочке принятия решений, чтобы отснить его действия на уровни, которые трудно поддаются автоматизации (неопределённость ситуации, недостаток информации и т. п.). Поэтому принятие решений с повышением уровня их сложности требует участия ЛПР и несущего за них ответственность. Задача о полной интеллектуализации принятия решений на машинном уровне задача не ставится. Речь идёт о создании гибкой системы поддержки принятия решений, которая на формализуемых этапах оставляет их выбор за компьютером, а в сложных ситуациях для человека имеющаяся информация приводится в удобный вид для принятия окончательного решения или выбора пути продолжения процесса. Смысл использования системы поддержки принятия решений заключается в облегчении обработки больших объёмов разнородной информации, включающей и такую, где необходимы советы специалистов различных профессий.

Особенно сложными стали управленческие решения при использовании управляющих технических систем в едином комплексе, содержащих людей и программно-управляемое оборудование (станки с ЧПУ, роботы и т. п.). Сложность повышается ещё и от того, что принятое решение надо преобразовать в набор последовательностей сигналов для оборудования. Если же имеется ещё и ряд критериев, среди которых есть противоречашие друг другу, то проблема дополнительно осложняется в выборе компромиссного варианта, который частично учитывает некоторые критерии, например, выбор расписания преподавателя с минимумом ожидания, когда освободится специализированный кабинет, когда их недостаточно. При желании сохранить высокий уровень автоматизации принятия эффективных решений прибегают к использованию заранее выработанной системы приоритетов в конкретных ситуациях.

В современных технологиях процессов принятия решений считается важным использование сетцентрических (GGG) технологий в системах мониторинга и анализа с целью эффективного сбора нужной информации из разнородных средств (операционных систем, компьютеров, датчиков и т. п.) в реальном масштабе времени. В частности, такие системы оправдали себя при управлении военными операциями.

Близкие к ним задачи решаются при использовании громадных объёмов данных на основе витрин, в которых на мировом уровне содержится краткое описание поисковых объектов, чтобы по адресу выйти на их нужную часть.

Хранилища и витрины данных в системах поддержки принятия решений также важны при необходимости быстрого подбора целевой информации из хранилищ, рассредоточенных по всему миру для конкретного предприятия.

Концепция витрины данных – перечень множества тематических баз данных, содержащих информацию, относящуюся к различным аспектам деятельности организаций.

Такой подход полезен для выбора данных, которые реально нужны за счёт их целенаправленного подбора из очень больших баз данных (по нескольким странам и т. п.).

Процесс принятия решений. Решение – это выбор альтернативы в различных сферах деятельности.

Выделяют особо организационные решения, смысл которых обеспечение движения к цели, т. е. создание условий для выполнения поставленной задачи. Число возможных альтернатив для рассмотрения, исходя из ресурсных возможностей (финансовых, технических, кадровых и т. п.). Обычно ограничено и часто удаётся назвать шаги, которые надо выполнить и их порядок для реализации поставленной задачи. Такого типа решения можно запрограммировать и поручать их выработку компьютеру. Принятию таких решений способствуют часто повторяющиеся ситуации.

Обычно выделяют типичные группы решений по видам, выполняемых функций планирования хода различных процессов, контроль.

Однако есть и ряд таких решений, некоторые этапы которых не поддаются программированию или допускают частичное программирование из-за наличия неизвестных и случайных факторов, которые трудно предвидеть. К таким объектам обычно относятся стратегические решения. Наиболее трудным являются те из них, в которых нужно преодолеть различного рода компромиссы (проблема занятости и переучивания работников из-за необходимости сохранить прибыльность предприятия, достижение оптимального решения за заданное ограниченное время и т. п.).

Полной автоматизации в первую очередь поддаются решения, основанные на суждениях. Они представляют собой знания и/или накопленный опыт, приводивший к успеху в прошлом, что побуждает их считать полезными в настоящем.

Принятие рациональных решений отличается тем, что они должны учитывать предшествующую историю развития технологии и/или предприятия и прогнозы развития событий. Поэтому информация для принятия решений базируется на результатах диагностики, которая позволяет сформулировать поле поиска альтернативных решений. Одновременно вырабатываются ограничения и критерии для принятия решений. К поиску рациональных решений побуждает и ограничение времени на их принятие, когда процесс выбора может прекращаться по критерию удовлетворения ограничений.

Риск в принятии решения должен опираться на оценки принимаемых решений по степени определенности их исхода, степень неопределенности может выражаться через вероятностную оценку. Ситуация значительной неопределенности наступает, когда невозможно оценить вероятность результатов. В частности, при изменении среды функционирования объекта в будущем, большими издержками времени и средств на получение информации, когда выгоды от ее получения сравнимы с затратами на ее добычу и обработку, трудность учета влияния принимаемых решений на технологию и условия труда и кадры.

Современные стандарты в области информационной безопасности, использующие концепцию управления рисками, очень важны в построении прикладных систем поддержки принятия решений.

Общие критерии предусматривают наличие двух типов требований безопасности: функциональных и доверия. Функциональные относятся к сервисам безопасности (идентификация, аутентификация, управление доступом). Требования доверия относятся к технологии разработки, тестированию, сопровождению, эксплуатационной документации и т. д. Работы должны производиться в выделенной вычислительной среде.

Интеллектуальная система – решатель творческих задач, который в своём составе имеет блоки: базу знаний, решатель и интеллектуальный интерфейс.

В технологии принятия решений часто рассматриваются интеллектуальные системы с вариантами принятия решений без участия человека и с участием ЛПР.

7.3. Рост сферы услуг в современном информационном обществе и его влияние на экономические и социальные процессы

Бизнес направляет усилия на борьбу за постоянного потребителя за счёт индивидуального учёта его запросов. Аналогично и государство в целом стало ориентироваться на потребности конкретного человека. Электронное правительство по своему замыслу должно предоставлять набор государственных информационных услуг гражданам, бизнесу, другим ветвям власти на основе готовых процедур их оказания. Тем самым оно становится частью электронной экономики. Например, реализация принципа «одного окна», предоставление общественных услуг гражданам (электронная почта, доступ к учебным и образовательным материалам, информация о продуктах и товарах, спорт и досуг, свободное распространение ПО, информация о банковском и страховом обслуживании и т. п.). Поэтому они в перспективе должны в какой-то мере контролироваться государством, чтобы защитить интересы граждан.

Информационные технологии сегодня рассматриваются как неотъемлемый компонент технологий управленческих, так как координация человеческой деятельности на современных предприятиях строится на основе новейших телекоммуникационных систем и компьютеров.

Традиционно под технологией понимается – формализованное описание деятельности, включающее набор ресурсов, инструментов, приёмов их использования и способов организации производства – необходимое и достаточное для воспроизводства процесса получения определённых продуктов, услуг и т. п. с заранее заданными параметрами. Глобальные перемены в сфере традиционных технологий произошли с появлением новой среды коммуникаций, приведшие к появлению особых форм взаимоотношений, названных сетевыми, которые реализуются на основе информационных технологий.

Информационная технология (ИТ) – совокупность методов, приёмов и средств, реализующих информационный процесс в соответствии с заданными требованиями. Они стали основным инструментом. Среди них выделяют базовые ИТ (реализуемые на уровне взаимодействия вычислительных систем, таких как ОС, ИНТЕРНЕТ, СУБД, криптозащита и т. п.) и прикладные ИТ, реализующие типовые процедуры обработки информации в конкретных предметных областях (автоматизация проектирования, управления).

Выделяют три вида информационного менеджмента: управление предприятием, внутренней документацией и публикациями.

В круг задач менеджмента входят также разработка, внедрение, эксплуатация и развитие автоматизированных информационных систем и сетей, обеспечивающих деятельность предприятия, взаимодействия с внешним информационным миром (сети, БД, издательства и т. п.).

Информационный менеджмент превращается в базовую технологию организации управленческой деятельности во всех сферах информационного общества.

Важнейшим аспектом в информационном обществе является организация труда отдельного работника, которая позволяет благодаря ИТ значительно повысить комфортность и производительность труда, по новому организовать его взаимодействие с фирмой.

К дистанционной работе относят такую, которая выполняется работником предприятия с использованием информационно-вычислительной техники и средств коммуникации всегда или время от времени вне пределов предприятия. Основным фактором является наличие связи работника и его рабочего места с предприятием.

Такие новые формы труда дают значительные экономические выгоды предприятию и повышают комфортность труда сотрудника и гибкость графика его работы, а также по исследованиям западных специалистов повышается производительность труда на 45 % и более.

Однако порождается и ряд новых проблем, связанных с контролем деятельности работника, сохранением конфиденциальности информации, оплатой труда, возмещением расходов работника по организации рабочих мест и за средства коммуникации. Такого рода работа имеет место и в условиях РФ. Например, сервисное обслуживание малыми предприятиями клиентов на дому, выполнение различных заявок, вручение подарков, подготовка информации по заданному кругу вопросов, программирование по заданию.

Интерес к дистанционной работе обычно проявляют высококвалифицированные и инициативные специалисты (программисты, инженеры, экономисты, переводчики, появляются свои особенности и в налогообложении, так как личная собственность может использоваться и для нужд предприятия и др.).

Объём современной мировой электронной торговли (Е-коммерция) оценивается в несколько триллионов долларов. Её основой являются дистанционные сделки, осуществляемые, как правило, через Интернет на основе транзакций. Под транзакцией понимается любая сделка между хозяйствующими субъектами по циклу «запрос – выполнение задания – ответ», обладающая свойствами неделимости, согласованности, надёжности, изолированности. Неделимость означает, что операция должна выполняться полностью, либо немедленно при наличии каких-то причин и обстоятельств отменяться (принцип «всё или ничего»). Свойство согласованности подразумевает взаимную целостность данных в базах обоих партнёров, свойство надёжности означает, что при успешном выполнении транзакции ни

при каких обстоятельствах данные не теряются. Изолированность означает, что доступ других транзакций до завершения выполняемой к используемым ею данным запрещён.

На основе понятия «транзакция» реализуется современная сетевая экономика, использующая бизнес-процессы, основой которых является обязательное взаимодействие субъектов через сети различного ранга (Интернет, Интранет, Экстранет и др.).

Исходной точкой для совершения сделок обычно является сетевая реклама, простейшей формой которой являются различные каталоги и специальные сайты «электронных» магазинов и предприятий-производителей продукции или услуг. Для ряда продаваемых продуктов реклама может носить специфический характер, который невозможно отразить за пределами электронных сетей. Специфика рекламы на базе сетевых операций содержит такой элемент, как использование медиасредств с одновременным воспроизведением объекта в действии, показом его различных возможностей с сопровождением демонстрации звуком, таблицами, текстом и т. д. Ещё более широкие возможности открываются в продаже программных и информационных продуктов, когда пользователь по сети может получить демонстрационную версию продукта и испытать её для своих условий, а также получить консультации и за умеренную оплату более совершенные версии продукта. По такого рода продуктам можно делать и заказы на их доработку под условия заказчика.

Особенность продаж предполагает наличие банковских счетов у партнёров, которыми они могут управлять дистанционно, а также наличие специальных средств фиксации оплаты сделки и необходимых перечислений за электронными подписями партнёров или банков, представляющих их интересы.

Под системой безналичных электронных расчётов будем понимать корпоративные сети для дистанционных коммерческих сделок, касающихся выполнения различных платёжных операций, как на основе пластиковых карт, так и путём управления счетами в банках по каналам связи непосредственно.

Основой таких операций являются счета в банках участников сделки. Они могут быть в разных банках одной страны и даже в зарубежных банках тоже. Корпоративная сеть подразумевает наличие ряда соглашений между её участниками, чтобы придать правовой статус любой сделке и должна обеспечивать конфиденциальность и надёжность сделок, а также проверку наличия денег на счетах и выполнения их перечислений в нужные адреса с выдачей подтверждений участникам сделки.

Отдельно рассмотрим платежи непосредственно с помощью пластиковых карт и аппаратуры и программных средств для управления счетами.

В первую очередь отметим общегосударственные эффекты от безналичных расчётов: уменьшение массы наличных денег и экономия на их печатании; экономия на процессах инкассации; уменьшение количества сделок без уплаты налогов, упрощение сделок, упрощение борьбы с преступностью типа грабежей касс и магазинов; краткосрочное кредитование за счёт хранения средств в банках.

Дистанционное управление процессами постепенно выходит на ведущие позиции. Например, БПС-Сбербанк предложил своим клиентам доступ к их банковскому счёту в течение 24-х часов на основе интернет-банкинга (оплата за квартиру, получение и использование пенсий через карт-счёт, оплату мобильных телефонов и ряд других платежей). В Норвегии к 2016 году планируется полностью отказаться от использования денежных знаков в платежах, за счёт применения систем безналичных расчётов.

Беларусбанк проявил себя как высокотехнологичная финансовая структура, выведя на рынок новый для отечественной банковской системы продукт с возможностью дистанционного управления счетом.

Внедряя новый продукт – «Удаленный депозит», он расширил линейку электронного обслуживания своих клиентов, помогая оперативно размещать в депозит временно свободные средства и дистанционно оперативно ими управлять. Дистанционное управление начали успешно использовать и в других сферах деятельности. Начали создаваться своеобразные виртуальные предприятия, ориентированные на электронную коммерцию.

Такое виртуальное предприятие представляет собой объединение географически раз-
несенных экономических субъектов, взаимодействующих в процессе совместной деятельности
на основе электронных средств коммуникаций для решения общих задач в рамках оговорен-
ных сроков и полномочий. Смысл его создания состоит в повышении эффективности опера-
ций, связанных с электронной коммерцией:

- устранение влияния географического фактора для снижения затрат на связь с контр-
партнерами;
- образование общего информационного пространства предприятия (доступ к ресурсам,
активизация сотрудников в формировании и использовании информационных ресурсов);
- улучшение внутрифирменной координации при принятии решений.

Кроме того, повышается возможность сокращения сроков и улучшения качества вы-
полнения заказа потребителя не за счёт организации нового производства, а поиска соответ-
ствующего нового партнёра. Поэтому на межгосударственном уровне потребуются правовое
обеспечение такой деятельности, например, как при пересечении границ товарами, так и ин-
теллектуальными продуктами.

Литература

[6, 9, 11, 13, 14, 22, 24, 26, 27, 29, 37, 41, 42]

Контрольные вопросы

1. Назовите особенности нанотехнологий, отличающие их от традиционных.
2. Что характерно для нанотранзисторов из различных материалов и биологических
элементов?
3. Охарактеризуйте перспективы использования нанокomпьютеров и нанороботов.
4. Какие новые возможности открываются при применении нанотехнологий в мобиль-
ной связи?
5. В чём заключаются преимущества мобильной связи через спутники?
6. Почему возможны сенсорные измерения на уровне живой клетки наноприборами?
7. Какие новые функции стала обеспечивать мобильная цифровая связь в автоматиза-
ции быта и банковских платежей?
8. Какова цель использования сетцентрических технологий (GGG)?
9. Почему необходимо накапливать типовые решения для интеллектуализации техно-
логий?
10. Назовите причины повышения реализации индивидуальных требований заказчиков.
11. Почему спутниковая мобильная связь стала использоваться в управлении на Земле?
12. Какие преимущества даёт переход к системе дистанционных безналичных платежей?

Глава 8. ПРАКТИЧЕСКИЕ И ЛАБОРАТОРНЫЕ ЗАДАНИЯ

8.1. Общие указания к практическим заданиям

Содержание заданий практических занятий и лабораторных работ ориентировано на усвоение теоретических основ алгоритмизации управления на основе типовых задач дискретной математики и защиты информации в прикладной деятельности при создании важных документов для охраны интеллектуальной собственности в виде проектной документации и ведения систем безбумажного документооборота.

Учебное пособие построено с ориентацией на активную самостоятельную работу студентов за счет индивидуализации заданий. Для их выполнения приводятся необходимые минимальные теоретические сведения и схемы. Творческая часть состоит в создании цепочки конкретных вычислительных моделей на основе средств высокоинтеллектуальной системы Mathcad и формулировании выводов по результатам вычислительного эксперимента. Эта система позволяет быстро и эффективно реализовывать различные модели с участием экспериментатора.

Все задания можно условно разбить на три категории: освоение общего математического аппарата и программных систем, используемых в разработке информационных компьютерных технологий; получение навыков по применению базовых элементов математического и программного аппарата при моделировании простейших объектов; освоение методов защиты информации и их использования в создании документов и баз данных с электронной цифровой подписью. Особо следует выделить операции по обеспечению надежности коммерческих операций в сетях и идентификации (аутентификации) пользователей. Ключевыми элементами являются вопросы описания и реализации процессов. Система Mathcad (и ее аналоги) хорошо подходит для первичного освоения навыков моделирования объектов и процессов, так как способствуют лучшему пониманию организации хода моделирования без его реализации в виде полной программы. Однако наличие развитой системы процедур позволяет реализовывать моделирование в смешанном человеко-машинном варианте, когда исследователь управляет процессом переходов между процедурами, описанными средствами Mathcad. По всем заданиям необходимо сделать краткое описание этого процесса и выводы о наблюдаемых его особенностях (регулирование количества итераций и точности, поведение заданной функции, причины останова процесса или отсутствия решения).

Кратко охарактеризуем особенности ключевых элементов математического аппарата криптографии. В первую очередь – это аппарат, развиваемый в теории чисел, теории вероятностей и математической статистики.

Во многих операциях используется операция сравнения по модулю: $x = a \bmod p$, где a и p – целые положительные числа, p – простое число, x при делении на p дает остаток a . На основе этого элемента легко показать реализацию односторонних функций, когда $y =$ (остаток) легко вычисляется при знании конкретного x , но обратная задача по y (остатку) вычислить x является многозначной, и только знание дополнительной секретной информации позволяет однозначно найти x . В том и другом случае решение прямой и обратной задач требует времени t как функции от некоторого полинома.

Вторым важным механизмом являются процедуры, основанные на применении подстановок и перестановок, чтобы менять исходный текст с использованием случайных чисел так, чтобы создать максимальные трудности криптоаналитику (злоумышленнику) при расшифровке закодированных сообщений.

В создании практических механизмов схем цифровой подписи и оплаты коммерческих сделок, а также банковских операций играют криптографические протоколы. Они применяются при обмене шифрованными сообщениями удаленных абонентов по открытым каналам связи. Часто бывает, что информация не является секретной, а для принимающей стороны (например, банка) важно убедиться, что информация действительно от его клиента, а клиенту важно убедиться, чтобы никто не изменил сумму в платежном поручении или кто-то по-

слал поддельное поручение. Задача конфиденциальности информации решается ее шифрованием, а ее целостность и аутентификация участников при обмене сообщениями обеспечивается протоколами (совокупностью алгоритмов, которые скрупулезно должны выполнять участники обмена информацией).

Особую роль в протоколе для идентификации подписавшего сообщение лица играют хэш-функции для преобразования сообщений произвольной длины в более краткие хэш-значения требуемой длины.

Для создания дополнительных трудностей в расшифровке сообщений и аутентификации их авторов широко используются и генераторы случайных чисел, например, при наложении гаммы (случайных битов).

Варианты всех заданий подбираются в зависимости от индивидуальных характеристик студентов: Ф, И и О – количество букв в фамилии, имени и отчестве соответственно, № – численное значение номера в группе (подгруппе).

8.2. Задания для 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. Оптимизация многомерных функций.

6-8. Построение логистического алгоритма доставки груза на основе решения задачи коммивояжера методом ветвей и границ для трёх модификаций алгоритма с целью получения навыков адаптации теоретической модели к решению прикладной задачи.

Выполнение заданий №№ 1-8 предполагает обязательное описание студентом алгоритма любым известным ему методом (например, из [14]). Теоретическая основа для выполнения заданий №№ 1-5 известна из лекционного курса и достаточно полно описана в [14, 27]. Поэтому для этих заданий подчеркнём лишь ключевые моменты, которые желательно выделить при оформлении результата:

– № 1. Главное отличие численных и логических алгоритмов, обеспечение конечности, однозначности и результативности процесса вычислений;

– № 2. Особенности ЭВМ как вычислительного инструмента (учёт погрешностей из-за ограниченной разрядной сетки и перехода к двоичной форме представления чисел, необходимость экономии памяти при экспоненциальном росте промежуточных результатов, желательность автоматизации ввода больших массивов информации, возможность сбоев в работе, необходимость предусматривать эффективные пути восстановления в случае чрезвычайных ситуаций и т. д.).

– № 3. В этой задаче в отчёте надо отметить сходство в программировании и реализации программ на абстрактной машине Поста и реальной ЭВМ.

– № 4. Отметить особенности использования конечного автомата как универсального устройства для управления заданным классом процессов (движение по выбранному типу контуров) на базе описания переходов состояний автомата в виде своеобразной программы.

– № 5. Необходимо усвоить диалоговые механизмы выработки рациональных решений.

– №№ 6-8. В этом (самом сложном) задании надо описать собственные предложения по построению деталей механизмов вычислений в пределах общей схемы реализации метода ветвей и границ в рамках каждого из этапов (построение и улучшение рекорда, выбор принципов оценки вершин, модификация метода под простейшую прикладную ситуацию).

Формулировки вариантов заданий с пояснениями даны ниже.

№ 1. Описать алгоритм Евклида и проверить его работоспособность на примере для положительных чисел $a = 4Ф$ и $b = 2И$. В соответствии с рис. 1.1.1 найти по алгоритму поиск

ка пути в лабиринте при их существовании между вершинами А и Ф, А и (Ф + О). Для использования номеров А, Ф, (Ф + О) приведём цифровые кодировки вершин графов (лабиринта): А = 1, В = 2, L = 3, Н = 4, К = 5, С = 6, D = 7, N = 8, R = 9, F = 10, E = 11, M = 12 (при Ф + О более 12 номер вершины выбирается 11 при нечётном Ф + О и 12 – в противном случае).

№ 2. Описать алгоритмы вычисления квадратного корня из $x > 0$ итерационным методом и малосвязанный алгоритм получения значения полинома, а также проверить их работоспособность для заданных условий: итерационный метод реализуется на основе формул ($y_0 = x$, $E = 0.1$; $y_i = 0.5(y_{i-1} + x / y_{i-1})$, $|y_i - y_{i-1}| \leq E$; $x(\text{test}) = \Phi + \text{И} + \text{О}$; №4 – шаг для сбоя, чтобы продемонстрировать устойчивость алгоритма к сбоям); для второй задачи использовать полином ($y = a_n x^n + a_{n-1} x^{n-1} + \dots + ax + a$) и известную схему для понижения связности алгоритма: $T_i = T_{i-1}x$, $y_i = y_{i-1} + a_i T_i$, $T_0 = 1$, $y_0 = a_0$ (данные для реализации и тестирования алгоритма: $x = 0$, $n = 4$, $a_0 = \Phi$, $a_1 = \text{И}$, $a_2 = \text{О}$, $a_3 = 2$, $a_4 = 1$).

№ 3. Запрограммировать алгоритмы для методов перекатывания и переноса меток при сложении двух расположенных подряд чисел a и b [27]. Исходные данные для проверки работоспособности программ: головка машины Поста находится над последней меткой первого числа, $a = \Phi$, $b = \text{И}$.

№ 4. Запрограммировать движение робота по контуру со следующими координатами угловых точек: (0,1); (0,И); (1,И); (1,И/2); (2,И/2); (2,1). Точка загрузки детали (0,И) и точка её выгрузки (1,И/2). Исходные данные и пример программы для управления роботом описаны в разделе 1.5.

№ 5. Найти максимальное при Ф-четном (или минимальное при Ф-нечетном) значение $F(x,y,z) = xz^2 - xy + (y - 5)^2$, когда $2 \leq x \leq \Phi$, $3 \leq y \leq \text{И}$, $3 \leq z \leq \text{О}$ методом покоординатного спуска (параграф 4.2).

№№ 6-8. Освоить реализацию метода ветвей и границ с прикладной ориентацией, используя в качестве исходной для выбора своего варианта таблицы, которая приводится далее по тексту.

Метод ветвей и границ в задаче коммивояжера используется во многих модификациях. Его относят к сложным дискретным задачам, которые в худшем случае точно решаются при полном переборе всех вариантов решений. Такого типа задачи называют NP полными. Их пытаются решать различными методами и используют также в экспериментах и для решений нейросетевыми методами. Таковой является и задача коммивояжера. Ее постановка: имеется n городов, между которыми заданы расстояния. Требуется из заданного города объехать все города по кратчайшему замкнутому маршруту, побывав в каждом городе только 1 раз и возвратиться в исходный город.

Суть метода ветвей границ заключается в развитии дерева решений по основному закону, когда следующая вершина дерева всегда строится в перспективном направлении на основании лучшего значения избранной функции оценки F , которая состоит из двух частей:

$$F = F_{\text{факт}} + F_{\text{оптимистич.}}$$

Трудность решения задачи состоит в подборе функции быстровычислимой $F_{\text{оптимистич.}}$ и получении любым авторским методом одного начального конечного решения $F = F_{\text{факт}} + F_{\text{оптимистич.}} (=0)$.

Название метода связано с тем, что из одной избранной вершины строятся все продолжения и получают их оценки (ветвление) и после получения оценок F вычеркиваются те вершины, которые имеют оценку $F \geq F_{TR}$ (текущий рекорд – лучший результат решения задачи на данный момент).

Типовая структура описания графа: в вершине (кружок, прямоугольник) располагается три цифры: вверху номер достигнутой вершины, слева F_{Φ} , а справа F_0 , а на ветви – номер вершины, из которой ветвь идет. Конечная вершина отмечается прямоугольником, на котором записано только F_{Φ} , так как $F_0 = 0$.

Покажем решение задачи этим методом, когда F_{TR} получено по принципу FIFO (первым пришел – первым обсуживается), а F_0 равно сумме всех минимальных путей для конкретной вершины с учётом уже пройденного пути.

Разработка авторских вариантов алгоритмов решения задачи коммивояжера методом ветвей и границ преследует несколько целей:

1. Освоить подходы к выработке собственных принципов разработки построения вспомогательных функций для повышения эффективности вычислений (прекращение решения по времени, сокращение числа вариантов, выявление нерешаемости из-за жестких требований и др.). Смысл выполнения задания заключается в описании элементов алгоритма и проведении практических расчетов на основе авторских оценочных функций и методов расчетов F_{TR} .

2. Предложить модификацию алгоритма, когда появляется некоторое ограничение, например, время прибытия не позже заданного для ограниченного количества точек (до начала обеда в школе, после дойки коров, ремонтные работы на дорогах в заданные интервалы времени и т. п.).

Отчетность по работе:

1. граф-схема с отображением всего процесса вычислений по алгоритму до заключительного шага. В записях отобразить списки вершин на момент перехода с яруса на ярус, а так же значение оценочных функций;

2. записать оптимальный маршрут (порядок прохождения точек);

3. описать свои модификации алгоритма и привести результаты вычислений, критически оценить эффективность алгоритма.

Таблица № 8.2.1 – Данные для решения задачи

$i \backslash j$	1	2	3	4	5	Min
1	///	1	2	6	И	1
2	1	///	3	5	3	1
3	3	4	///	Ф	4	3
4	4	4	5	///	4	4
5	3	4	5	3	///	3

Примечание: Ф – количество букв в фамилии

И – количество букв в имени

8.3. Задания для лабораторных работ в системе Mathcad

Этот раздел включает задания для 11-ти лабораторных работ, которые необходимы для усвоения типовых элементов теории моделирования на ЭВМ базовых механизмов при реализации интеллектуальных компьютерных технологий. Основным средством выполнения работ является система Mathcad, так как она содержит ряд полезных процедур для построения и адаптации моделей: символьное и численное решение большого класса задач прикладной математики, аппарат для реализации случайных процессов, механизмы округления чисел до целых с избытком и недостатком, возможность гибкого построения графиков в нужном масштабе и с одновременным построением графиков нескольких функций на одном чертеже. Наличие этих возможностей полезно и для построения криптографических процедур защиты информации, аутентификации пользователей, генерирования электронной цифровой подписи и т. д., и для понимания механизмов защиты передаваемой информации в сетях ЭВМ.

Лабораторная работа № 1. Построение графиков полиномов $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ на отрезке $[p, r]$ в системе Mathcad.

Выполнение задания опирается на типовую структуру системы Mathcad для построения графиков в декартовой системе координат, которая, будучи однажды задана, позволяет управлять процессом вычислений для получения графика заданного полинома в измененном масштабе для любого подотрезка, вызывающего интерес исследователя.

Исходя из этой возможности, следует построить необходимые графики для ответов на следующие вопросы:

1. Найти графическим методом отрезки убывания и возрастания полинома.
2. Найти приближённо, с точностью до $\epsilon = 0.1$, корни полинома.
3. Описать своими словами схему решения задачи и сделать выводы об особенностях исследований в вашем варианте функции.
4. Вычислить полином в корневых точках.

Одна из возможных схем построения декартова графика полинома приведена ниже.

Задаем количество точек на данном отрезке.

$n := 50$ (при необходимости его можно в дальнейшем изменять).

$i := 0..50$ (две точки вводятся нажатием клавиши «ок» (в MATCAD), запись обеспечивает цикл для вычислений 50 точек графика).

Далее задаём свой вариант

$a :=$ $b :=$ $c :=$ $d :=$

и координаты начала и конца отрезка, на котором исследуется функция:

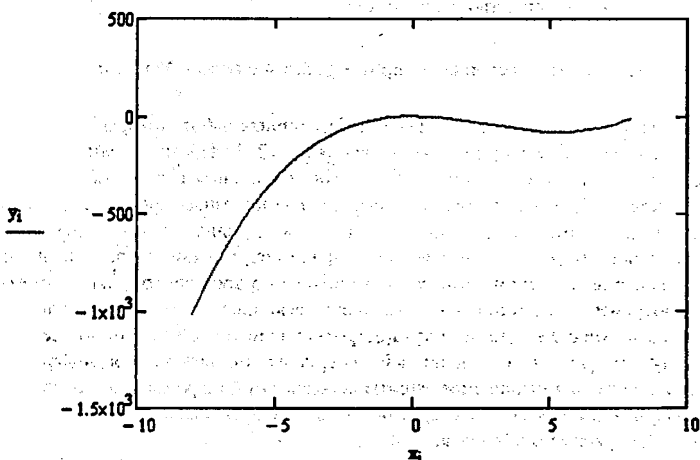
$p := -8, r := 8.$

Определяем текущие значения x и y в рассматриваемой точке:

$$x_i := p + \frac{r-p}{n} \cdot i$$

$$y_i := a \cdot x_i^3 + b \cdot x_i^2 + c \cdot x_i + d.$$

После этого строим график. Для этого на панели инструментов выбираем «График» и в появившемся окне выбираем «График X-Y». После появления его шаблона на чёрных прямоугольниках заполняем x и y , и щёлкаем левой клавишей мыши за пределами графика, что даёт возможность получить первый график функции с целью определения последующих отрезков для более тщательного её исследования.



Новый исследуемый отрезок задается изменением p и r , и результат отображения функции получается после щелчка рядом с графиком.

Варианты для выполнения лабораторной работы даны в таблице 8.3.1.

Таблица 8.3.1 – Выбор варианта

№ варианта	a	b	c	d
1	1	-8	-1	1
2	1	-8	1	1
3	2	-16	1	1
4	1	-15	1	1
5	1	-9	1	15
6	2	-20	1	15
7	2	15	1	1
8	1	5	1	1
9	1	4	1	1
10	1	2	-1	1
11	1	-3	-4	-4
12	1	-2	0	-4
13	1	-2	1	-5

Оформление и предъявление отчета возможно в двух вариантах: в печатном, что лучше, или на компьютере на сайте студента, где будут храниться все лабораторные работы, которые дублируются на флешке студента с целью обеспечения их сохранности.

Лабораторная работа № 2. Решение уравнений n -й степени от одной переменной, содержащих простейшие трансцендентные функции (тригонометрические, логарифмические, показательные) на заданном отрезке $[p, r]$ типа $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 - k \sin(tx); \ln tx; k^x$.

Требуется решить задачу для своего варианта тремя путями:

1. Графическим методом с точностью до 0,1 с одновременным построением графиков на заданном отрезке для двух функций (полинома и трансцендентной) найти координаты их точек пересечения. В качестве корней записываются все точки пересечения двух графиков, построенные по общей процедуре в Mathcad:

$$n := 50$$

$$i := 0..50$$

Далее для своего варианта указываются конкретные значения коэффициентов и координаты начала и конца отрезка, на котором исследуется функция:

$$a := \quad b := \quad c := \quad d := \quad k := \quad t :=$$

$$p := -8$$

$$r := 8$$

Определяется текущее значение x_i , общее для двух графиков, и значения функций y_i и z_i :

$$x_i := p + \frac{r-p}{n} \cdot i,$$

$$y_i := a \cdot x_i^3 + b \cdot x_i^2 + c \cdot x_i + d,$$

$$z_i := k \sin(tx_i).$$

После этого строим декартов график. Для этого на панели инструментов выбираем «График» и в появившемся окне выбираем «График X-Y». После появления его шаблона на черных прямоугольниках внизу, заполняем x_i , а слева посередине записываем y_i , ставим запятую и далее z_i . Щёлкаем мышью рядом с графиком, что дает возможность получить первый график для определения отрезков для более тщательного исследования расположения корней уравнения. При необходимости путем изменения длины исследуемого отрезка получают более точные значения корней и записывают их с нужной точностью.

2. Каждый корень находится вновь с помощью процедуры root для отыскания одного корня. В качестве его начальной величины подставляются в порядке следования корни, найденные графическим путем и соответствующие отрезки для них. Последовательно фиксируются результаты, полученные с помощью процедуры root. Типовая структура для решения задачи (исходные данные используются те же: коэффициенты и длины отрезков), так как они

фиксировались ранее и их можно использовать снова. Перед процедурой `root` задается соответствующее значение x , равное значению корня, и отрезок $[p, r]$, на котором этот корень был найден, т. е.:

$$x :=$$

$$f(x) := a \cdot x^3 + b \cdot x^2 + c \cdot x + d - k \sin(tx)$$

`root (f(x), x, p, r)`, где p и r – границы найденного корня, и щелчком получается результат.

3. Регулирование точности вычислений при применении процедуры `root` и любых других по количеству знаков делается так: открывается меню «формат», затем подменю результат и указывается требуемое число десятичных знаков в открывшемся окне (5 пять! 5 набирается с клавиатуры). После чего пересчитываются все корни с применением процедуры `root`.

4. Вычисляются все значения функции для всех найденных корней. Делаются выводы о степени приближения результатов к нулю в зависимости от точности. Для отчета по работе описывается ход эксперимента, и делаются выводы.

Варианты для выполнения лабораторной работы даны в таблице 8.3.2.

Таблица 8.3.2 – Выбор варианта

№ варианта	a	b	c	d	k	t
1	1	-8	-1	1	5	2
2	1	-8	1	1	6	2
3	2	-16	1	1	7	2
4	1	-15	1	1	5	3
5	1	-9	1	15	6	3
6	2	-20	1	15	7	3
7	2	15	1	1	5	4
8	1	5	1	1	6	4
9	1	4	1	1	7	4
10	1	2	-1	1	5	5
11	1	-3	-4	-4	5	6
12	1	-2	0	-4	5	7
13	1	-3	1	1	5	8

Лабораторная работа № 3. Символьные и численные решения уравнений и неравенств, содержащих функции алгебраического типа.

Требуется освоить применение символьных вычислений с последующим получением конкретных результатов численными методами. Задача решается для функций в виде полиномов $P(x)$ и алгебраических дробей $P(x) / Q(x)$ в соответствии с заданными вариантами

Mathcad является полезной системой для изучения в рамках специальности «Искусственный интеллект», так как содержит процедуры символьных вычислений, когда результаты отображаются в символьной форме: вычисления производных, интегралов, уравнений, неравенств и др. Это позволяет решать ряд задач в общей алгебраической форме, а потом подбирать значения нужных параметров и вычислять конкретные значения для найденных решений в символьной форме. Для этой цели используются палитры символы, вставить функцию, матрица, математика и другие.

Отбор основных функций для изучения диктуется тем, что решение неравенства часто связано с необходимостью решить уравнение, упростить функцию в ее символьной записи или сделать преобразование для толкования результатов.

К таким процедурам для более глубокого изучения отнесем: `simplify` (упростить), `solve` (решить уравнение или неравенство), `factor` (представление выражений в виде сомножителей), `expand` (разложение рациональных дробей $P(x) / Q(x)$) и некоторые другие.

Отбор и освоение этих инструментов диктуется необходимостью освоения полезной информации и навыков в решении неравенств и уравнений, содержащих в своем составе элементарные алгебраические функции. Главным объектом исследования являются полиномы, так как для них наиболее развит аппарат символьных вычислений. В частности, получение решения уравнений в символьной форме и различные преобразования алгебраических выражений. Поэтому основными исходными объектами для исследования выбираются алгебраические уравнения и неравенства типа:

$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ ($= 0$, или > 0 , или < 0), а так же алгебраические дроби типа $P(x) / Q(x)$, содержащие полиномы в числителе и знаменателе.

Конкретный вид объекта определяется вариантом лабораторной работы, который предусматривает получение символьного решения, а затем по полученному объекту с подстановкой данных варианта конкретного решения. Желательно проанализировать возможные границы их колебаний в рамках существования полученного решения и сделать выводы.

Задания для изучения имеют следующую структуру:

1. $P(x) = ax^2 + bx^2 + cx + d$ ($= 0$, или > 0 , или < 0).

2. $P(x) / Q(x) = (ax^3 + bx + c) / (ax^2 + bx + c) > 0$.

По первому заданию получить символьное и численное решение, воспользовавшись процедурой solve с представлением d в буквенной форме при получении символьного решения. Далее против строки слева с символьным решением присваивается конкретное значение d , вычисляются корни, и после этого записывается решение неравенства $P(x) > 0$ по известным из средней школы методам. Далее при конкретных a, b, c, d для полинома $P(x)$ применяется процедура solve и получается решение неравенства в символьной форме. Требуется сравнить оба результата.

По третьему заданию вычисляется предел дроби $P(x) / Q(x)$ с помощью оператора lim, когда x стремится к n , а затем вычисляется лимит при $n = d$.

Описать ход работы и сделать выводы.

Таблица № 8.3.3 – Выбор варианта задания:

№ варианта	a	b	c	d
1	1	2	-1	-2
2	1	0	-3	-2
3	1	-3	-1	3
4	1	-4	-1	4
5	1	-1	-4	4
6	1	-3	-4	12
7	1	-4	-4	16
8	1	-6	3	10
9	1	-6	11	-6
10	1	-7	14	-8
11	1	-8	17	-10
12	1	-9	20	-12
13	1	-8	19	-12

Лабораторная работа № 4. Символьное дифференцирование функций в системе Mathcad и его применение (частные производные в градиентных методах оптимизации).

Требуется вычислить определитель 3-го порядка в символьной форме, содержащей цифровые и буквенные элементы (x, y, z) . От полученной функции $F(x, y, z)$ найти частные производные F'_x, F'_y, F'_z как будущие компоненты градиента (вектора, указывающего своим направлением наискорейшее возрастание функции: $F'_x \cdot \vec{i} + F'_y \cdot \vec{j} + F'_z \cdot \vec{k}$). Далее вычислим их

нужное количество раз в требуемых точках. Найти локальные минимум и максимум $F(x, y, z)$, когда $a - \varepsilon \leq x \leq a + \varepsilon$, $b - \varepsilon \leq y \leq b + \varepsilon$, $c - \varepsilon \leq z \leq c + \varepsilon$. Начальная точка для вычисления частных производных равна количеству букв в Ф, И, О, т. е. $x = \Phi = a$, $y = \text{И} = b$, $z = \text{О} = c$, $\varepsilon = 0,1$. Первая строка определителя – (1, x , 2); вторая – (3, 4, y), третья – (z , 5, 6). Сделать вывод о возможном существовании определителя, равного нулю, так как смысл решения задачи состоит в установлении факта возможности отсутствия решения системы линейных уравнений из-за того, что переменные x, y, z замерялись приборами с ошибкой ε , и в скрытой форме может существовать определитель, равный нулю.

Опишем процедуры отыскания локального минимума (максимума): в полученной текущей точке вычисляются знаки производных по переменным x, y, z для построения следующей точки. Учитывая, что структура исследуемой функции $F(x, y, z)$ такова, что ее максимум (минимум) достигается на концах отрезков области изменения переменных, то допустимо применение метода наискорейшего спуска, когда по знаку частной производной для данной переменной больше нуля при поиске минимума выбирается левая сторона отрезка, а для максимума правая и, наоборот, в противном случае. Процесс поиска обрывается при заикливание, т. е. при получении одной и той же точки (x, y, z) второй раз. Одновременно ведется вычисление значения функции в каждой полученной точке. Заикливание соответствует прекращению нарастания (убывания) функции. Все шаги протоколируются по итоговым результатам. Процедура вычисления определителя представляет собой символьные действия над векторами и матрицами (вычисление определителей, транспонирование и др.). Будем рассматривать все операции над матрицами, так как вектор можно считать однострочной или одностолбцовой матрицей. Матрица может задаваться следующим образом: на панели управления набирается «вставка» и в ее меню отмечается матрица и на экране можно задать количество строк и столбцов. Аналогичный результат получим сразу, если щелкнем в меню по изображению незаполненной матрицы. Заполним конкретными данными матрицу $n \times n (3 \times 3)$. Через панель управления (символика) можно вызвать подменю «матрица» и в нем выбрать нужную операцию (транспозиция, инвертирование, определитель).

Обратившись к меню символика, выполним операцию по вычислению определителя и в итоге получим $F(x, y, z) = 54 - 5y - 18x + zxy - 8z$, соответствующее значению $\det A$ в символической форме. После этого вычислим в символической форме общий вид частных производных F'_x, F'_y, F'_z и, подставляя в них конкретные значения текущих точек, приступим к процедуре выполнения оптимизации.

Описать процесс вычислений и сделать выводы. Дополнительно для связи с задачами, решаемыми в курсе высшей математики, вычислить обратную и транспонированную матрицу для вшего задания. Для контроля результата лабораторной работы можно применить процедуру Minimize (Maximize) по следующей схеме:

$F(x, y, z) :=$ значение символического определителя.

$x := 2; y := 5; z := 8$ (любые числа по возможности близкие к min, max)

$a - \varepsilon \leq x \leq a + \varepsilon$

$b - \varepsilon \leq y \leq b + \varepsilon$

$c - \varepsilon \leq z \leq c + \varepsilon$

$P := (\text{Minimize}) (f, x, y, z), P = (\text{обыч. равно}) f(P_0, P_1, P_2) = (\text{обыч. равно} = \text{ставится в момент чтения}).$

Лабораторная работа № 5. Отыскание экстремумов функций многих переменных в системе Mathcad.

Требуется вычислить \max и $\min F(x, y, z) = x^3y^2z - 2x^2yz + 3xy + c$. От полученной функции $F(x, y, z)$ найти частные производные как будущие компоненты градиента (вектора, указывающего своим направлением наискорейшее возрастание функции: $F'_i \cdot \vec{i} + F'_j \cdot \vec{j} + F'_k \cdot \vec{k}$).

Далее вычислим их нужное количество раз в требуемых точках. Найти локальные минимум и максимум $F(x, y, z)$, когда $\Phi - 2 \leq x \leq \Phi + 2$, $I - 2 \leq y \leq I + 2$, $O - 3 \leq z \leq O + 3$. Начальная точка для вычисления частных производных равна $(b-a) \gamma$ (вычисляется как $x(y, z) = a + rnd(b-a)$), где $(b-a)$ – величина отрезка для соответствующей переменной; Φ, I, O – количество букв в фамилии, имени, отчестве.

Опишем процедуры отыскания локального минимума (максимума): в полученной текущей точке вычисляются знаки производных по переменным x, y, z для построения следующей точки. Учитывая, что структура экстремумов исследуемой функции $F(x, y, z)$ неизвестна, требуется испытать разный шаг h по каждой переменной, по знаку частной производной для данной переменной больше нуля при поиске минимума выбирается движение к левой стороне $(x-h)$ отрезка, а для максимума – к правой $(x+h)$ и, наоборот, в противном случае. Процесс поиска обрывается при заиклипании, т. е. при получении одной и той же точки (x, y, z) второй раз (при выходе за пределы отрезка в качестве значения переменной выбирается пересекаемая граница). Одновременно ведется вычисление значения функции в каждой полученной точке. Заикливание соответствует прекращению нарастания (убывания) функции. Все шаги протоколируются по итоговым результатам. Процедура вычисления F служит дополнительным критерием движения к цели.

Вычислим в символьной форме общий вид частных производных F'_x, F'_y, F'_z и, подставляя в них конкретные значения текущих точек, приступим к процедуре выполнения оптимизации. Рекомендуем начать с шага $h=1$, понижая его в процессе после заиклипания постепенно до уровня 0.1.

Описать процесс вычислений и сделать выводы. Для контроля результата лабораторной работы можно применить процедуру Minimize (Maximize) по следующей схеме:

$$f(x, y, z) := x^3 y^2 z - 2x^2 yz + 3xy + c$$

$x :=$; $y :=$; $z :=$ (любые числа по возможности близкие к точкам min, max)

$$\Phi - 2 \leq x \leq \Phi + 2,$$

$$I - 2 \leq y \leq I + 2,$$

$$O - 3 \leq z \leq O + 3$$

$P :=$ (Minimize) (f, x, y, z) $P =$ (обычное равно) $f(P_0, P_1, P_2)$ (обыч. равно = ставится в момент чтения результата).

Вторая попытка делается из новой случайной точки $(a + rnd(b-a))$ при повторном решении первым методом.

Лабораторная работа № 6. Моделирование информационных процессов со случайными компонентами.

Освоить механизм моделирования независимого от исследователя выбора хода процесса методом статистических испытаний, сделать выводы об особенностях таких вычислений с учётом излагаемой далее теории. Модель – представляемая мысленно или материально реализованная система, которая в процессе познания, анализа замещает реальный объект, сохраняя важные для исследования его черты. По сути дела модель является упрощенным образом реального объекта для более глубокого изучения действительности. Метод исследований, базирующийся на разработке и использовании моделей, называется моделированием. Модель всегда предполагает участие в ее создании (конструировании) исследователя, так как на нем лежит ответственность за адекватность модели объекту с позиции изучаемых его свойств.

Смысл использования моделей состоит в упрощении и удешевлении процессов изучения, а иногда их использование практически представляет почти единственный путь к получению необходимых результатов (угроза взрыва, дороговизна натурального эксперимента и т. п.). Обычно различают методы материального (существует связь с материальными объектами модели) и идеального моделирования (рассматриваются мысленные связи между объектом и моделью).

Среди формализованных моделей выделяют знаковые модели, в которых используются системы знаков с правилами их преобразования и интерпретации. Его частным случаем является математическое моделирование, которое характерно тем, что разные процессы (объекты) могут описываться одинаковыми формулами и уравнениями. Например, законы о течении жидкостей и газов, расширения твердых тел и т. п.

Во всех случаях и видах моделирования соблюдается главный принцип: модель должна соответствовать оригиналу в главном только в том, что интересует исследователя, а несущественное можно отбросить.

Математическое моделирование наиболее важно при использовании вычислительной техники. Это связано с тем, что с его помощью можно получить доказательство существования решения задачи и оценить время получения результата, диапазон изменения переменных, возможную точность результата и т. д.

При математическом моделировании на ЭВМ возможно формировать базы данных для других задач, из типовых элементов – строить прочие модели. Корректное построение всех видов моделей невозможно без участия специалистов: выбор переменных, формирование ограничений.

Среди многочисленных видов моделей важнейшая роль отводится моделям принятия решений и их оптимизации. Наибольшую трудность обычно вызывают задачи многокритериальной оптимизации, которые поддаются получению приемлемых решений с применением ЭВМ, а иногда и с участием человека, который формирует путь поиска решений.

Особая роль отводится вероятностным моделям и реализации на их основе вероятностных процессов. Эти модели используются в имитационном моделировании, когда требуется изучить характеристики процесса, на который влияют случайные факторы, путем его многократного воспроизведения по элементарным шагам и фиксирования различных средних характеристик в целом путем подсчета количества воспроизводимых явлений в изучаемом процессе (например, отыскание емкости резервного бункера заготовок для обработки на автоматической линии по причине поломок или затупления инструмента, поиска показателей реальных случайных процессов для статистических оценок разброса измерений и т. п.).

Вероятностное моделирование на ЭВМ обычно опирается на использование генератора случайных чисел равномерно распределенных в диапазоне (0, 1) или (0, α), который содержится в большинстве вычислительных систем в виде стандартной процедуры. На его основе обычно строятся случайные процессы с использованием стандартных процедур для воспроизведения элементов моделей в виде конкретных случайных законов (экспоненциальный, нормальный и др.).

Второй путь использования генератора случайных чисел состоит в выборе случайных условий или путей продолжения процессов, когда требуется исключить влияние исследователя на выбор текущего продолжения процесса, например, стартовых точек для различных итерационных процессов вычислений для многомерных функций и т. п.

Информационные модели – знаковые модели, описывающие процессы получения, передачи, преобразования и использования информации. Их основой во многих случаях являются вычислительные модели, на базе которых разрабатываются алгоритмы и программы. При машинном моделировании всегда реализуется знаковые модели. Иногда говорят, что компьютерное моделирование – это моделирование теории изучаемого объекта. В этом смысле нейронная сеть является теорией для грубого представления функционирования мозга. Имитационные модели воспроизводят (имитируют) процессы функционирования объекта. Программа для ЭВМ – формализованная информационная модель деятельности. Фотография – статическая информационная модель.

В данной работе предлагается повторить лабораторную работу № 5 многократно при случайном выборе координат 5-ти начальных точек (в вашем варианте функция $F(x, y, z) =$

$= x^2y^2 + z + 3x^2y - yz$ с областью определения $\Phi - 2 \leq x \leq \Phi$, $\text{И} - 3 \leq y \leq \text{И}$, $\text{О} - 1 \leq z \leq \text{О}$, градиентным методом найти минимум и максимум функции (лучшие результаты из 5 попыток, их же вычислить с помощью процедуры Maximize (Minimize). Шаг изменения каждой координаты может быть разным в диапазоне от 1 до 0.1. Протокол одной попытки поиска результата можно представлять в укрупнённой форме в виде логической цепочки векторов, каждый из которых должен содержать значения компонент: координаты точки, значение функции и частных производных, рекомендуемый следующий шаг по каждой координате (0 или $-h$ или $+h$). При обобщении шагов координаты одного знака суммируются. Для эффективной работы полезно построить такую структуру: первая строка – переменные с присвоением текущих значений координат, вторая строка – функция в алгебраической форме и в 3-5 строках аналогично записываются частные производные. Такая структура позволяет сразу вычислить весь вектор за исключением шагов, подбираемых исследователем. Оформить выводы, обобщив результаты всех попыток и проверок.

Лабораторная работа № 7. Моделирование решения задач методом Монте-Карло.

Освоить метод вычисления площадей и сложных функций на основе моделирования случайных процессов, осветить проблемы достижения точности результатов и провести их сравнительный анализ с обычными численными методами для различных типов функций.

Метод Монте-Карло (статистических испытаний) – численный метод решения математических задач при помощи моделирования случайных процессов и событий. Его название происходит от города Монте-Карло, известного своими игорными домами.

Для реализации случайной величины используют датчики, генерирующие случайную последовательность чисел, равномерно распределенных на интервале (0, 1). Различают чаще всего два типа генераторов случайных чисел: физические датчики (источник – реальные физические процессы типа шумов электронных приборов, радиоактивные излучения и т. п.); программные датчики псевдослучайных чисел (эти числа по своим свойствам практически неотличимы от полученных физическими датчиками).

Чтобы обеспечить получение цепочки случайных событий, обычно их воспроизводят, используя набор стандартных программ для моделирования известных случайных событий с фиксированными параметрами (математическое ожидание, дисперсия и т. п.).

Это порождает проблему доказательности доверия к результату, полученному методом Монте-Карло. Поэтому в серьезных исследованиях проверяются источники моделирования на соответствие теоретическим параметрам при выполнении заданного количества вычислительных опытов (обычно от 100 до 10 000). Это связано с тем, что ошибка вычислений обычно пропорциональна величине $1/\sqrt{N}$, где N – количество опытов, например, при 400 опытах можно ожидать ошибку до 0,05 от конечного результата.

В данном случае ставится задача проверить на пригодность модели для получения чисел, распределенных по равномерному закону, который моделируется для базовых параметров в интервале (0, 1) и математического ожидания 0,5 и среднего квадратичного отклонения 0,083.

Практическое применение в различных математических задачах метод нашел при вычислении площадей, ограниченных сложными контурами, когда обычным интегрированием решить задачу очень сложно, а иногда отсутствуют и готовые методы ее решения. Идея предлагаемого метода заключается в следующем: сложную фигуру неизвестной площади помещают в квадрат (круг, прямоугольник), площадь которого известна. После этого моделируют вбрасывание N случайных точек в квадрат, и фиксируют количество точек n , попавших в искомую фигуру. При большом числе точек справедливо равенство: $S_{\text{фигуры}}/S_{\text{квадрат}} \approx n/N$. Этот способ можно использовать для проверки качества генератора случайных чисел из интервала (0, 1).

Для этих целей возьмем квадрат со стороной 1 и впишем в него четверть круга единичного радиуса. Тогда $S_{\pi} = 1$; $\frac{1}{4} S_{\pi} = \frac{1}{4} \pi R^2 = \frac{1}{4} \pi$. Тогда при получении n/N приблизительно равно 0,78 ($\pi/4$) можно сделать вывод о работоспособности генератора. С целью выполнения такой проверки вбрасывают N случайных точек. Если хотят обеспечить точность 0,01, то для доверия к результату необходимо вбросить около 10 000 точек.

Проверить на практике, при скольких точках может быть достигнута заданная точность E и построить график зависимости точности от N ($N = 100, 200, \dots, 1000$). Один из возможных вариантов реализации этой проверки можно осуществить по следующей схеме, позволяющей получить для любого N декартов график и n/N :

$$N := T_0 = 0$$

$$i := 1..N$$

$$t_i := (\text{rnd}(1))^2 + (\text{rnd}(1))^2$$

$$T_i := T_{i-1} + (2 - \text{ceil}(t_i))$$

$$K_i := i$$

$$R_i := T_i / K_i$$

Замечания: N – задаваемое количество точек, t_i – сумма квадратов координат текущей точки, T_i – попавшие в четверть круга точки, R_i – результат, на графике по оси x записывается K_i , а по оси y – R_i . Для вычисления интегралов материал подбирается самостоятельно из курса высшей математики. Для $f(x) > 0$ на отрезке (a, b) можно при $a > 0$, $\max f(x) = c$ воспользоваться формулой $S = c(b - a)n/N$. В этом случае x вбрасывается координата $(a + \text{rnd}(b - a))$, а в качестве y – координата $\text{rnd}(c)$. Считается, что испытание прошло успешно (случайная точка попала под кривую или на неё: $f(a + \text{rnd}(b - a)) - \text{rnd}(c) \geq 0$), а в противном случае – она выпала над кривой в площади окаймляющего прямоугольника.

Лабораторная работа № 8. Одноключевая система шифрования Диффи и Хеллмана.

Построить систему шифрования Диффи и Хеллмана для a = (количество согласных букв в фамилии студента), $p \geq$ (количество всех букв в фамилии). Подобрать a и p самостоятельно методом проб и ошибок, выбрать два секретных числа X_i и X_j для связи пользователей сети i и j вычислить числа Z_i и Z_j .

В методе Диффи и Хеллман реализована идея использования функций с лазейкой для построения криптосистемы в сети с открытым распределением ключей. Для решения этой задачи они предложили использовать функцию $F(x) = a^x \text{ mod } p$, где p – большое простое число, x – произвольное натуральное число из множества $\{1, 2, \dots, (p-1)\}$, a – целое число из множества $\{2, 3, \dots, p\}$, для которого выполняется требование, чтобы все степени a от 1 до $(p-1)$ в произвольном порядке по модулю p дали все числа из множества $\{1, 2, \dots, (p-1)\}$.

Например, при модуле $p = 7$ можно выбрать $a = 3$:

$$f(1) = 3^1 \text{ (mod } 7) = 3, \quad f(2) = 3^2 \text{ mod } 7 = 2, \quad f(3) = 3^3 \text{ (mod } 7) = 6, \quad f(4) = 3^4 \text{ (mod } 7) = 4,$$

$$f(5) = 3^5 \text{ (mod } 7) = 5, \quad f(6) = 3^6 \text{ (mod } 7) = 1.$$

Предполагается, что всем пользователям сети известны a и p . Пользователь i случайным образом выбирает число x_i (свою лазейку), т. е. секретное число известное только ему из множества $\{1, 2, \dots, (p-1)\}$. Далее он вычисляет $y_i = a^{x_i} \text{ (mod } p)$ и помещает его в открытый для доступа всех пользователей сети справочник. При желании установить секретную связь с пользователем j он берет из справочника его число y_j и с помощью своего секрета x_i для обмена сообщениями с j вычисляет ключ $Z_{ij} = (y_j)^{x_i} \text{ (mod } p)$. После установления

контакта аналогичную работу проделывает пользователь j , который с помощью своего секретного числа x_j вычисляет $Z_{ji} = (y_i)^{x_j} \bmod p$. Ограничения, наложенные на выбор a , обеспечивают получение равенства $Z_{ij} = Z_{ji}$, т. е. одинаковых ключей для обмена сообщениями.

В самом деле, $Z_{ij} = y_j^{x_i} \bmod p = (a^{x_j})^{x_i} \bmod p = a^{x_i x_j} \bmod p$ и $Z_{ji} = a^{x_j x_i} \pmod{p}$.

Пример: ($p=7, a=3, x=\{1, 2, 3, 4, 5, 6\}$).

$x_i = 3$ (секрет i), $y_i = 3^3 \bmod 7 = 6$.

$x_j = 4$ (секрет j), $y_j = 3^4 \bmod 7 = 4$.

$Z_{ij} = 4^3 \bmod 7 = 1$.

$Z_{ji} = 6^4 \bmod 7 = 1296 \bmod 7 = 1$.

Цифра 1 может означать некоторую функцию, которая используется при кодировании; страничку в заранее разосланных пользователям материалах и т. д.

Недостаток описанной криптосистемы с открытым распространением ключей состоит в том, что она требует абсолютного доверия партнеров по связи друг к другу, так как в этой одноключевой системе они могут изменять переданный текст. Поэтому она непригодна, например, для не доверяющих друг другу удаленных абонентов. Вычисление остатков x при делении целых чисел на модуль y можно выполнять с помощью функции $\text{mod}(x, y)$.

Лабораторная работа № 9. Двухключевая система RSA.

Построить двухключевую систему с использованием алгоритма RSA и выполнить в ней операцию шифрования и дешифрования трех первых букв фамилии студента (при количестве букв меньше 3, недостающие буквы берутся из имени). Пара простых чисел P и Q выбирается из диапазона ближайших к количеству букв в фамилии и имени студента.

Например, Петров Владимир, P (5 или 7), Q (7 или 11). Методом испытаний подбирается также ближайшая пара чисел E и D .

В нашем случае это могут быть $P = 5, Q = 7$.

В случае неудачных сочетаний из названного диапазона берутся рядом другие ближайшие простые числа, например, $P = 5, Q = 13$.

В системе RSA каждый пользователь имеет свой ключ шифрования. Ключи дешифрования известны всем, а шифрующий ключ держится в секрете. Криптографические системы типа RSA подходят для реализации цифровой подписи, применяемой в системах электронных платежей и при передаче сообщений с помощью устройств телесвязи.

К недостаткам системы RSA и аналогичных ей относят ее существенно более низкое быстродействие и потребность в более длинных ключах. Наиболее эффективные реализации RSA характеризуются скоростью шифрования порядка нескольких тысяч бит в секунду. Тогда как аналогичные реализации более простых систем шифруют несколько миллионов бит в секунду. В связи с этим наиболее целесообразным применением RSA считается организация обмена секретными ключами, необходимыми для обеспечения безопасности в сетях связи.

Основная проблема для системы RSA – генерация соответствующей пары ключей. Для генерации используется следующая процедура:

1. Выбрать 2 простых числа P и Q ;
2. Найти произведение $N = PQ$ и число $L = (P - 1)(Q - 1)$;
3. Выбрать случайное число D такое, что оно должно быть взаимно простым с числом L (числа называются взаимно простыми, если они не имеют общего делителя);
4. Определяют другое число E такое, что $(ED) \bmod L = 1$;
5. Как только все числа найдены, мы имеем: секретный ключ – E ; открытый ключ – пара чисел D и N ;

Тогда при шифровании сообщения его разбивают на блоки M . В результате шифрования для каждого блока M получим число

$$C = (M^E) \bmod N.$$

При дешифрации получаем:

$$M^* = (C^D) \bmod N.$$

Рассмотрим это на примере алфавита из букв $\{A, O, Я\} = \{1, 2, 3\}$ для передачи текста «ОЛЯ» (или 2, 1, 3). Цифровые обозначения букв или блоков обязательны, так как метод основывается на обработке натуральных чисел.

1. Выберем $P = 3$ и $Q = 11$;
2. Найдем $N = PQ, N = 33; L = (P - 1)(Q - 1); L = 20$;
3. Выберем D , взаимно простое с $L: D = 3$;
4. Выберем E такое, что $(ED) \bmod L = 1: E = 7$, действительно, $(7 \cdot 3) \bmod 20 = 1$;

5. Тогда открытый ключ $\left. \begin{array}{l} D = 3 \\ N = 33 \end{array} \right\}$ секретный $E = 7$.

Производим шифрацию своим закрытым ключом 7:

$$C1 = (2^7) \bmod 33 = 29,$$

$$C1 = (M1^E) \bmod N: C2 = (1^7) \bmod 33 = 1,$$

$$C3 = (3^7) \bmod 33 = 9.$$

Зашифрованный текст получается (29,1,9).

Расшифровка текста открытым ключом 3 и 33:

$$M1^* = 29^3 \bmod 33 = 2,$$

$$M1^* = (C1^D) \bmod N: M2^* = 1^3 \bmod 33 = 1,$$

$$M3^* = 9^3 \bmod 33 = 3.$$

В результате мы получили исходный текст.

Остается только добавить, что для получения достаточно стойкой шифровки необходимо брать очень большие простые числа.

Выполнение соотношения $(ED) \bmod L = 1$ позволяет использовать этот факт для проверки подлинности подписи без знания секретного ключа E с помощью аппарата ХЭШ-функций.

В практической работе необходимо идентифицировать автора электронного документа и предприятие не по особенностям подписи и печати (например, по образцам подписей и печатей в банковской карточке клиента), а по наличию у него электронного ключа для подписывания документов. В этом случае конкретное число-подпись под данным документом в фиксированное время, может сделать только законный обладатель ключа (E).

Процедура электронной подписи включает в себя два этапа: первый – подписывание (вычисление параметров подписи, зависящих от текста конкретного документа, один из которых (E) хранится в секрете); второй – проверка получателем с помощью несекретных параметров (D, N) подлинности сообщения (подписи)

Сообщение шифруется по алгоритму RSA, где E подбирается и известно только отправителю, а D, N знает и получатель. Получатель должен иметь возможность с помощью открытого ключа проверить подлинность сообщения. Для этой цели в сообщение добавляется еще одно число, которое является результатом вычисления хэш-функции $h(T)$, зависящей от текста T .

Лабораторная работа № 10. Обеспечение подлинности сообщений.

Подобрать хэш-функцию $h(T)$ или применить из теоретического материала и используя секретный ключ E из предыдущего задания и зашифрованное сообщение (три буквы) вычислить $m = h(T)$ и $S = (m^E) \bmod N$. Далее, пользуясь открытым ключом D вычислить m из соотношения $(S^D) = m \bmod N$ и убедиться в его совпадении с m владельца секретного ключа. В конечном виде передаваемое $m < N$.

К хэш-функции предъявляется ряд требований:

- невозможность (или за очень длительное время) найти по значению $h(T)$ само T (т. е. требуется построить практически необратимую функцию);

- для заданного T нельзя найти такое T' , чтобы $h(T) = h(T')$;

- вообще нельзя найти пару различных слов T и T' такую, что $h(T) = h(T')$;

- сообщение T (например, текст договора, платежного поручения и т. п.) по заданной функции сжимается в целое число $m = h(T)$, причем $1 < h(T) < N$.

Число m позволяет с помощью открытого ключа констатировать подлинность документа.

С этой целью автор документа с помощью своего секретного ключа E получает второй параметр подписи $S = (m^E) \bmod N$. Параметры m и S вставляются в текст сообщения на место подписи и печати. Все сообщение по телекоммуникационным каналам передается получателю. Он проверяет правильность цифровых параметров m и s , исходя из знания функции $h(T)$, полученного символического объема зашифрованного сообщения ($T1$) и «лазейки» для вычисления $h(T)$, $h(T1)$.

Проверка параметра S производится путем идентификации условия:

$$(S^D) = m \bmod N$$

Математически доказано, что результат проверки m и S будет положительным в том случае, когда в их формировании использовался секретный ключ E , соответствующий открытому ключу D . Вероятность расшифровки секретного ключа E по открытым параметрам S , m , D и N считается ничтожно малой из-за затрат времени на решение задачи взлома системы велико по сравнению со временем полезного действия сообщения.

Покажем в упрощенном варианте проверку подписи сообщения $T = (\text{ОЛЯ})$, с дополнением его параметрами m и S . В качестве функции хеширования $h(T)$ возьмем произведение Π сумм из двух элементов для каждой буквы: ее цифрового кода и простого числа r из ряда $(2, 3, 5, 7, \dots)$ с позиции, соответствующей расположению буквы в тексте. Например, для слова ОЛЯ, коды букв $O = 2$, $L = 1$, $A = 3$, а их позиции в слове 1, 2, 3, тогда $T = \Pi \bmod 33 = (2 + 2)(1 + 3)(3 + 5) \bmod 33 = 29$. Можно убедиться, что эта функция удовлетворяет требованиям к ней по крайней мере для любого сообщения из 3-х разных букв. Это обнаруживается при вычислении всех возможных шести разнобуквенных сообщений T : (ОЛЯ) – 29, (ОЯЛ) – 12, (ЛОЯ) – 21, (ЛЯО) – 27, (ЯОЛ) – 18, (ЯЛО) – 8, т.е. нет равных $h(T)$.

Следует заметить, что с ростом длины сообщения может оказаться, что такая функция не удовлетворяет требованию, когда два сообщения разной длины имеют одинаковое значение, т. е. $h(T) = h(T')$. Чтобы избежать такого явления можно разбивать сообщения на блоки заранее ограниченной длины. Поэтому выбор хорошей функции $h(T)$ является очень трудной задачей и ее решением занимаются специалисты, которые разрабатывают стандарты шифрования. В описанном нами примере ограничимся лишь демонстрацией процедуры признания подписи.

Текст T , который получает партнер позволяет проверить подлинность подписи по параметрам m , S и $D = 3$ (известно как открытая часть ключа), S и m приходят с текстом отправителя ($S = m^E \bmod N = 29^7 \bmod 33 = 17$; $m = 17$, т.е. меньше 33). Проверка подлинности сообщения: $S^D = m \bmod N$; $17^3 \bmod 33 = 29$, т.е. результат проверки положителен и под-

пись подлинна. Кроме того, если получатель тоже знает как вычислить функцию хеширования от текста, то по прочитанному тексту он может ее вычислить.

В настоящее время в Республике Беларусь выпущен предварительный стандарт СТБ П34, 101.25-2008 для электронной подписи, в котором алгоритм RSA один из трех рекомендуемых для применения.

Лабораторная работа № 11. Методы борьбы с контрафактной продукцией на основе БД.

Проверить штрих-код любого предприятия по описанному далее алгоритму и отметить его полезные функции в автоматизированных системах управления, логистике, создании баз данных. Предложить скрытую часть кода (пломбируемую или защищенную краской), например, какую-то функцию от серийного номера изделия, гаммированный передаваемый в двоичной форме серийный номер изделия. Гаммирование можно выполнить с помощью части таблицы случайных двоичных чисел: 0101100010111010101110001010110. В этом случае передающая и принимающая стороны знают начальную строку и позицию цифры в ней, с которой начинать отсчет. Пусть требуется гаммировать блок 10101111010011100110 с помощью таблицы с пятой цифры в ней. Тогда, например; запись 7624 переводится в двоично-десятичный код каждой из цифр: 0111011000100100 и получим гаммированный блок: 1000001011000010.

Особенности использования штрих-кодов в борьбе с контрафактной продукцией (подделки от имени предприятия – производителя) можно использовать также особого рода протоколы.

В основе борьбы от подделок лежат скрытые и открытые маркировки производителя. Обычно такие маркировки состоят из двух и более частей. Открытая часть может содержать штрих-код (например, EAN-13), из этого используемого в Европейской практике международного классификатора видна страна происхождения товара по первым трём цифрам, по следующему 4-м код продукта и ещё по следующему 4-м код предприятия и последняя цифра является контрольной). Всей этой информации достаточно, чтобы обратиться к производителю. Кодирование скрытой части индивидуально для каждого изделия и представляет собой набор случайных генерируемых программой цифр и машинно-читаемых знаков. Скрытую часть невозможно прочитать без нарушения целостности изделия или упаковки или без удаления стирающегося слоя краски (подобно защите лотерейных билетов спортлото).

Производитель создаёт уникальную электронную базу данных с неповторяющимися идентификаторами тождественными скрытой маркировке на изделии. При первом запросе покупателя на предприятие для подтверждения подлинности кода изделия, этот код удаляется в другую базу данных, т. е. вторая и последующие авторизации невозможны, а при несоответствии первого предъявления уникального кода или его повторном предъявлении, можно приступить к борьбе с авторами подделок.

Эта же система исключает и продажу неучтённой продукции, выпущенной на самом предприятии. Кроме того, исключаются и грубые подделки штрих-кода. Существует простой алгоритм проверки кода EAN-13.

Алгоритм проверки кода:

1. Сложить цифры, стоящие на четных местах S_r .
2. Получить $S_1 = S_r \times 3$.
3. Сложить цифры на нечетных местах S_n (без контрольной).
4. Получить $S = S_1 + S_n$.
5. Оставить от S только число в младшем разряде (l).
6. Найти разность $P = 10 - l$.

При правильном коде P должно совпасть с контрольной цифрой.

Пример. 8590721001209.

1. $S_r = 5 + 0 + 2 + 0 + 1 + 0 = 8$.

2. $S_l = 8 \times 3 = 24$.

3. $S_n = 8 + 9 + 7 + 1 + 0 + 2 = 27$.

4. $S = 24 + 27 = 51$.

5. $t = 1$.

6. $P = 10 - 1 = 9$ (совпадает с контрольной цифрой)

Изменим любую цифру кода, например 7 на 5.

Тогда

3. $S_n = 8 + 9 + 5 + 1 + 0 + 2 = 25$.

4. $S = 24 + 25 = 49$.

5. $t = 9$.

6. $P = 10 - 9 = 1$, т. е. контрольная цифра говорит о чувствительности кода (она не совпала).

Важность этого типа кода определяется еще и тем, что ряд банков Республики Беларусь используют для счетов предприятий 13-разрядные коды типа EAN-13. Ошибка персонала предприятия при заполнении документов, тогда легко вскрывается по контрольному разряду и банк не выполняет перевод денег.

Идея скрыть некоторые элементы передаваемой информации или сам факт ее передачи и хранения оказалась плодотворной в создании сложных многоступенчатых систем защиты информации.

Многоступенчатость защиты информации выражается как в смысле средств защиты информации по блокам, использовании разных ключей, поэтапном сочетании методов стеганографии и криптографии.

Компьютерная стеганография – это сокрытие сообщения или файла в другом сообщении или файле. Информация может быть в виде текста, изображения, звука или их сочетания. Для сокрытия хранимой или передаваемой информации используется контейнер – специально подобранная другая информация. Защищаемая информация встраивается в контейнер по заданным правилам так, чтобы на фоне контейнера она ничем не выделялась. Для сокрытия зашифрованной защищаемой информации применяется секретный стегоключ.

Часто в шифровании используется и операция гаммирования, когда по определенному закону перед шифрованием на открытые данные (обычно в двоичном виде) делается наложение гаммы – псевдослучайной последовательности (D_r^i). Процесс шифрования тогда содержит процедуру генерации гаммы шифра и ее наложения на исходный текст обратимым образом. Обычно гаммирование исходного двоичного текста выполняется путем его сложения с гаммой по модулю 2 (\oplus). Процедура гаммирования характерна для блочного шифрования, когда открытые данные разбиваются на блоки D_o^i одинаковой длины (чаще всего 64 бита). Каждый открытый блок D_o^i путем сложения с гаммой преобразуется в гаммированный блок D_m^i аналогичной длины готовый для шифрования. Уравнение гаммирования для каждого блока i из набора k блоков, тогда записывается так $D_m^i = D_o^i \oplus D_r^i$. Обратная процедура на приемном конце сводится к повторной генерации гаммы по известному для принимающего закону. Тогда расшифрованный текст легко получается по формуле: $D_o^i = D_m^i \oplus D_r^i$. Такой метод позволяет изменять гамму для каждого шифруемого блока случайным образом за счет генерации псевдослучайных чисел.

Для компьютерных продуктов характерна защита интеллектуальной собственности в форме ноу-хау, так как эти объекты пока не патентуются.

Частью системы правовой охраны промышленной собственности в Республике Беларусь является защита от недобросовестной конкуренции.

«Недобросовестная конкуренция – любые направленные на приобретение преимуществ в предпринимательской деятельности действия хозяйствующих субъектов, которые противоречат требованиям добросовестности и разумности и могут причинить или причинили убытки другим хозяйствующим субъектам-конкурентам либо нанести ущерб их деловой репутации».

Любые действия, направленные на ограничение или устранение конкуренции путем нарушения прав других хозяйствующих субъектов на свободную конкуренцию, а также нарушающие права и законные интересы потребителей, не допускаются (незаконное использование фирменного наименования, товарного знака, копирование внешнего вида товара другого хозяйствующего субъекта, введение в гражданский оборот товаров другого хозяйствующего субъекта с использованием собственных средств индивидуализации товара, неправомерные утверждения при осуществлении предпринимательской деятельности, способные дискредитировать хозяйствующий субъект, товары или предпринимательскую деятельность конкурента).

Объекты промышленной собственности охватывают результаты интеллектуальной деятельности, имеющие производственную направленность. К объектам промышленной собственности относят изобретения, полезные модели, промышленные образцы, селекционные достижения, топологии интегральных микросхем, нераскрытую информацию.

Заключительные общие рекомендации. При необходимости использования результатов в конкретных целях (в выполнении курсовой, дипломной, магистерской или другой работы, носящей исследовательский характер) полезно прибегнуть к дополнительному изучению литературы из списка источников [4, 13, 36, 37]. Смысл их использования состоит в необходимости системной увязки вопросов защиты информации и интеллектуальной собственности с экономических и юридических позиций. Это необходимо при подготовке объектов для выхода на международные рынки, так как здесь решающим фактором становятся экономические и правовые вопросы: оцениваются затраты на рыночное продвижение продукта и его защищенность (патентование объектов промышленной собственности, регистрация товарного знака, патентная чистота объекта и т. д.). Вторым важным объектом являются различные дистанционные операции в сетях ЭВМ (электронная коммерция и платежи между субъектами хозяйствования, выполнение дистанционных совместных работ и заказов).

8.4. Задания для лабораторных работ по дисциплине «Автоматизация проектирования вычислительных машин и систем»

Лабораторная работа № 1. Освоение навыков работы в САПР Quartus II.

Создайте файл блок-диаграмм с описанием разрабатываемого модуля `resclk` (рис. 8.4.1), выполните компиляцию и моделирование (рис. 8.4.2).

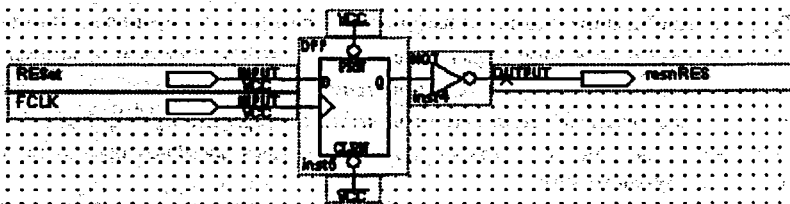


Рисунок 8.4.1 – Схема

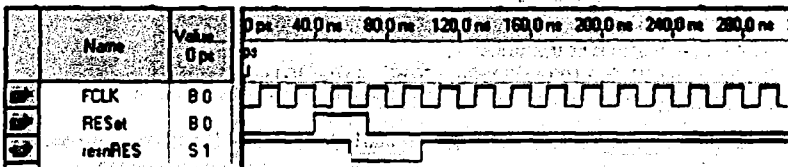


Рисунок 8.4.2 – Результат моделирования

Ответьте на вопросы.

- 1) В каком редакторе создан проект? Назовите тип созданного файла и его расширение.
- 2) Какие другие редакторы есть в Quartus II? Назовите типы создаваемых с их помощью файлов и их расширений.
- 3) Как проверяется правильность проекта?
- 4) Какие в Quartus II 7.1 есть средства анализа и синтеза? Посмотрите установки в созданном проекте. Как их изменить?
- 5) Что такое Pin Planner? Прокомментируйте его работу.
- 6) Что такое Chip Planner? Прокомментируйте его работу.
- 7) Как запрограммировать ПЛИС?

Выполните индивидуальное задание – проектирование мультиплексора, демультимплексора, шифратора, дешифратора, сумматора, вычитателя, счетчика, компаратора, регистра, схемы сдвига.

Лабораторная работа № 2. Маршрут моделирования в системе ModelSim.

Создайте VHDL-модель полусумматора и тестирующую программу.

VHDL-модель полусумматора представляет собой текстовый файл add1.vhd.

```
entity add1 is
port (b1,b2 : in BIT; -- входные сигналы
cl,s1 : out BIT); -- выходные сигналы: s1 – сумма, cl – перенос в следующий разряд
end add1;
architecture struct_1 of add1 is
begin
s1<= ((b1 and (not b2)) or ((not b1) and b2)); cl<= b1 and b2;
end struct_1;
```

Тестирующая программа для проверки правильности VHDL-модели полусумматора представляет собой текстовый файл test_add1.vhd.

```
entity Test_add1 is
end Test_add1;
architecture Behavior of test_add1 is
component add1
port (b1,b2 : in BIT; -- в VHDL не различаются строчные
cl,s1 : out BIT); -- и прописные буквы
end component;
signal b1, b2 : BIT;
signal cl, s1 : BIT;
begin
pl : add1 port map (b1 => b1, b2 => b2, cl => cl, s1 => s1);
```

```

b1 <= '1', -- входные воздействия
'0' after 50 ns, '1' after 100 ns, '0' after 150 ns, '1' after 200 ns;
b2 <= '1', -- входные воздействия
'1' after 50 ns, '1' after 100 ns, '0' after 150 ns, '0' after 200 ns, '1' after 250 ns;
end behavior;

```

Создайте новый проект, добавьте файлы add1.vhd и test_add1.vhd, задайте очередность компиляции (рекомендуется выбрать Auto Generate) и выполните компиляцию.

Затем в окне «Simulate-Start Simulation» раскройте рабочую библиотеку, выберите имя головной программы (test_add1), остальное – по умолчанию (в ModelSim 6.5 и более поздних версиях в окне Start Simulation флаг Enable optimization не должен быть установлен, иначе не будут видны сигналы в окнах Objects при моделировании). Откройте окно Objects (в главном меню «View - Debug Windows – Objects» установить флаг Objects). Из этого окна мышью можно будет перетащить отдельные требуемые для наблюдения сигналы в окно Wave (это окно может быть открыто командой «View - Debug Windows -Wave»).

Выполните моделирование (команда Run All в окне Wave). Результат моделирования показан на рис. 8.4.3.

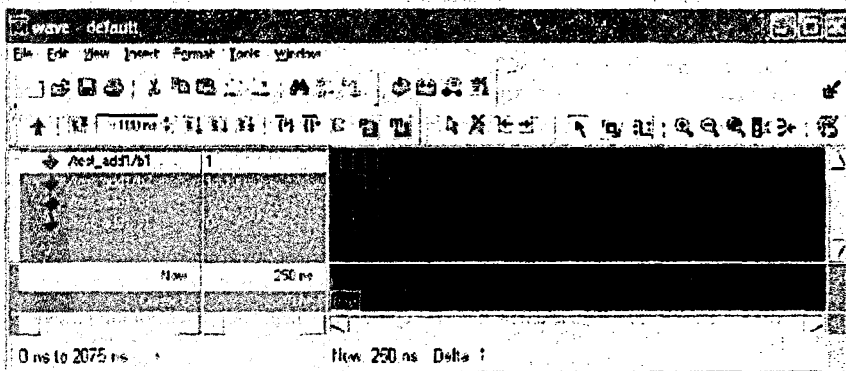


Рисунок 8.4.3 – Результат моделирования

Лабораторная работа № 3. Описание и моделирование нерегулярных логических схем.

1. Для заданной нерегулярной логической схемы составьте ее описание в графическом редакторе Quartus II, подготовьте файл временных диаграмм и выполняйте моделирование на всех наборах значений входных переменных.

2. Для той же схемы подготовьте структурное VHDL-описание, компоненты для нее составьте с использованием их таблиц истинности или логических выражений.

3. Составьте тестирующую программу для всех наборов значений входных переменных и выполните моделирование в ModelSim.

4. Сравните результаты моделирования.

Варианты заданий даны на портале VHDL [45, работа № 3.1, работа № 3.2].

Лабораторная работа № 4. Проектирование функциональных устройств последовательного типа на языке VHDL.

1. Для заданного в виде таблицы автомата Мура (Мура) подготовьте VHDL-описание.

2. Составьте тестирующую программу для всех наборов значений входных переменных и выполните моделирование в ModelSim.

Варианты заданий даны на портале VHDL [45, работа № 4 (часть 2)].

8.5. Пример заданий для письменной работы

Учреждение образования
«Брестский государственный технический университет»

Письменная работа студента (Ф.И.О.) _____

группы ИИ-11 факультета ЭИС по дисциплине:

«Традиционные и интеллектуальные информационные технологии».

Задание № 1. Описать схему итерационного алгоритма вычисления $y = \sqrt{x}$ на базе метода, использующего формулы $y_i = 0.5(y_{i-1} + x/y_{i-1})$ с критерием по точности вычислений $|y_i - y_{i-1}| \leq \varepsilon = 0,1$, где $x = И \times \text{rnd}(И) + \Phi \text{rnd}(\Phi)$ – данное для проверки алгоритма, для доказательства его устойчивости к сбоям сделать ошибку на шаге 3 и сравнить результаты, полученной без сбоя с результатом со сбоем, при вычислениях сбой сделать в целой части y на 1. (И, Φ – количество букв в имени и фамилии студента).

Задание № 2. Методом ветвей и границ решить задачу коммивояжера для 5 городов, расстояния a_{ij} между которыми заданы матрицей $[a_{ij}]$, где $a_{ii} = 0$, $a_{ij} = 1 + \text{ceil}(\text{rnd}(\Phi))$.

В ответе указать найденный маршрут в виде вектора $(x_1, x_2, x_3, x_4, x_5, x_1)$ и длину кольцевого маршрута L , где x_i – номер города

Задание № 3. Решить трансцендентное уравнение $a_3 x^3 + a_2 x^2 + a_1 x + a_0 - k \sin(tx) = 0$ на заданном отрезке $[p, r]$ графическим методом в системе Mathcad с точностью 0,1.

Перечислить все действительные корни и выполнить проверку полученных результатов с помощью процедуры root. Исходные данные: $p = И \text{rnd}(И)$; $r = p + \Phi \text{rnd}(\Phi)$; $a_3 = 1$; $a_2 = 1 + \text{rnd}(2)$; $a_1 = 2 \text{rnd}(3)$; $a_0 = -И$, где И, Φ – количество букв в имени и фамилии.

Сделать вывод о разнице результатов, полученных по двум методам.

Задание № 4. Построить систему шифрования Диффи и Хелмана для $a = 1 + \text{ceil}(\text{rnd}(И / 2))$; p – ближайшее простое число к сумме $(a + \Phi)$, удовлетворяющее правилу построения системы. Выбрать два секретных числа x_i и x_j и для связи пользователя i и j вычислить числа z_{ij} и z_{ji} .

Задание № 5. Методом Монте-Карло вычислить интеграл

$$\int_0^{\Phi} \sqrt{(x^3 - x^2 + u)/(x^2 + \Phi)} dx$$
 путем вбрасывания $N_1 = 1000 + И$ и $N_2 = 5000 + \Phi$ точек. Этот

же интеграл найти численным методом, используя систему Mathcad и сравнить оба результата. Оценить величину ошибки моделирования в % для этих двух вычислений и сделать выводы (Φ – количество букв в фамилии, И – количество букв в имени). Записать в системе Mathcad ход вычисления $n = T_i$ (количество удачных попыток в испытаниях для N_1 и N_2).

Задание № 6. Построить двухключевую систему шифрования с использованием алгоритма RSA и выполнить в ней операцию шифрования и дешифрования осмысленного слова из четырех букв в алфавите {А, Б, В, Л, О, Я}: Р и Q выбрать из пар (3, 11), (5, 7), (5, 11), (5, 13), (5, 19) – пара определяется по номеру, равному величине $\text{ceil}(\text{rnd}(4))$.

Заведующий кафедрой _____ В.А.Головко Преподаватель _____ Л.П.Матюшков

Дата утверждения: . 2015, протокол № заседания кафедры.

ЗАКЛЮЧЕНИЕ

Перспективы внедрения в повседневную жизнь человека различных систем «искусственного» интеллекта грандиозны:

- стремительное нарастание количества «умной» 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 совместного взаимодействия с мобильными средствами цифровой связи;

- развитие систем «электронных» правительств, обеспечивающих контакты избирателей по получению информации о постановлениях и готовящихся проектах;

- управление из космоса движением на Земле;

- 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.

- создание виртуальных предприятий.

Можно специально выделить процессы обучения, так как их совершенствование коренным образом должно повлиять на индивидуальную подготовку отдельного специалиста:

- внедрение дистанционных систем обучения с индивидуальной 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).

В последние годы усилились разработки по организации прямого поиска и передачи информации в системах из разнородных технических и программных компонентов (GGG-системы). Эти исследования открывают пути к глобальному получению информации в реальном масштабе времени работы пользователя из любого уголка Земли.

Нанотехнологии становятся также одним из важных элементов бурного развития компьютерных технологий: совершенствование компьютерной базы (биокомпьютерные, квантовые и т. п.), прямое биоуправление роботами, создание роботов для синтеза себеподобных объектов и т. д.

Полезность предложенного нами подхода в методическом плане состоит в лёгкой адаптируемости использования материалов для ряда специальностей, широко применяющих описанные методы. Построение системы обучения включает такие важные моменты: развитие исследовательских навыков во всех практических и лабораторных работах у студентов младших курсов, выдача индивидуальных заданий для каждого студента, возможность проверки результатов другими методами (косвенное подтверждение правильности выполнения задания для студента и преподавателя), рекомендуемая письменная работа по сути является комплексным закрытым тестом по всему учебному курсу.

ЛИТЕРАТУРА

1. Логика. Автоматы. Алгоритмы / М.А. Айзерман [и др.]. – М. : Физматгиз, 1963. – 556 с.
2. Таненбаум, А. Архитектура компьютера: Пер. с англ. / А. Таненбаум. – СПб. : Питер, 2002. – 704 с.
3. Бибило, П.Н. Основы языка VHDL / П.Н. Бибило. – М. : Солон, 2002. – 224 с.
4. Баричев, С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов // –М. : Горячая линия, Телеком, 2001. – 120 с.
5. Информационные системы и технологии в экономике / Т. П. Барановская [и др.]. – М. : ФиС, 2003. – 416 с.
6. Витязь, П.А. Основы нанотехнологий и наноматериалов: уч.пособие // П.А. Витязь, Н.А. Свидурович. – Минск : Выш.шк., 2010. – 302 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. Бондаенко, В.Ф. MATLAB. Основы программирования, компьютерная математика. Учебный курс / В.Ф. Бондаренко, В.Д. Дубовец. – Минск : Харвест, 2010. – 256 с.
9. Глушков, В.М. Основы безбумажной информатики / В.М. Глушков. – М. : Наука. Ф.М., 1982. – 552 с.
10. Головки, В.А. Нейронные сети: обучение, организация и применение. Кн. 4, Учебное пособие для вузов / В.А. Головки. – М. : ИПРЖР, 2001. – 256 с.
11. Норенков, И.П. Основы автоматизированного проектирования / И.П. Норенков. – Издательство МГТУ имени Н.Э. Баумана, 2000. – 360 с.
12. Татур, М.М. Классификаторы в системах распознавания: прикладные аспекты синтеза и анализа / М.М. Татур, Д.Н. Одинец. – Минск : Бестпринт, 2008. – 165 с.
13. Головки, В.А. Основы защиты информации и управления интеллектуальной собственностью: учебно-методический комплекс / В.А. Головки, Л.П. Матюшков. – Брест : Изд-во УО «БрГТУ», 2011. – 76 с.
14. Головки, В.А. Основы вычислительных систем: методическое пособие // В.А. Головки и [др.]. – Брест : Изд-во УО «БрГТУ», 2013. – 148 с.
15. Дудкин, А.А. Обработка изображений в проектировании и производстве интегральных схем / А.А. Дудкин, Р.Х. Садыхов. – Минск : ОИПИ НАН Беларуси, 2008. – 270 с.
16. Дзо, Н. Комбинаторные алгоритмы / Н. Дзо, Ю. Нивергельт, Э. Рейнгольд // Пер. с англ. – М. : «Мир» 1980. – 476 с.
17. Искусственный интеллект. Кн.1: Системы общения и экспертные системы / Под ред. В. Попова. – М. : Радио и связь, 1990. – 420 с.
18. Карпов, Б. Microsoft Access 2000: справочник / Б. Карпов. – СПб. : Питер, 2001. – 416 с.
19. Кемени, Дж. Кибернетическое моделирование / Дж. Кемени, Дж. Снелл // М. : «Советское радио», 1972. – 192 с.
20. Криницкий, Н.А. Алгоритмы и роботы / Н.А. Криницкий. – М. : Радио и связь, 1983. – 168 с.
21. Программирование и алгоритмические языки / Н.А. Криницкий [и др.]. – М. : Наука, 1979. – 509 с.
22. Головки, В.А. От многослойных персептронов к нейронным сетям глубокого доверия: парадигмы обучения и применения / В.А. Головки // Лекции по Нейроинформатике. – М. : НИЯУ МИФИ, 2015. – С. 47-84.
23. Базы данных. Интеллектуальная обработка информации // В.Г. Корнцев [и др.] – М. : Нолидж 2001. – 496 с.
24. Лисьев, Г.А. Технология поддержки принятия решений // Г.А. Лисьев, И.В. Попова. – М. : изд. Флинта, 2011. – 133 с.

25. Макаров, Е.Г. Инженерные расчеты в Matcad-14 / Е.Г. Макаров. – СПб. : Питер, 2007. – 592 с.
26. Мацукевич, В.В. Основы управления интеллектуальной собственностью. Учебно-методический комплекс: учеб. пособие / В.В. Мацукевич, Л.П. Матюшков. – 2-е изд. – Минск : Выш. шк., 2013. – 224 с.
27. Матюшков, Л.П. Традиционные и интеллектуальные информационные технологии. Методические указания к выполнению практических занятий и лабораторных работ / Л.П. Матюшков, В.А. Головки. – Брест : БрГТУ, 2014. – 30 с.
28. Матюшков, Л.П. Основы машинной математики / Л.П. Матюшков, А.А. Лихтарович. – Минск : Нар. Асвета, 1988. – 240 с.
29. Матюшков, Л.П. Перспективы информационных технологий в развитии дистанционных рабочих мест и коммерческих операций / Л.П. Матюшков // «Вучоныя запіскі БрДУ». – Брест : БрДУ, 2006. – Т.2. – Ч.1. – 107-115 с.
30. Пархоменко, П.П. Основы технической диагностики / П.П. Пархоменко, В.С. Сагомоян. – М. : Энергониздат, 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 / Э.В. Попов. – 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. Положение о коммерческой тайне (утв. СМ РБ 06.11.1992 №-670).
36. Предварительный государственный стандарт Республики Беларусь СТБ П 34.101.25-2008. Информационные технологии. Стандарт электронной цифровой подписи.
37. Об электронном документе и электронной цифровой подписи. Закон Республики Беларусь (28 декабря, 2009 года, №113-3) – 10 с.
38. Тихонов, А.Н. Вводные лекции по прикладной математике / А.Н. Тихонов, Д.П. Костомаров. – М. : Наука, 1984. – 192 с.
39. Успенский, В.А. Машина Поста / В.А. Успенский. – М. : Наука, 1979. – 95 с.
40. Гэри, М. Вычислительные машины и труднорешаемые задачи: Пер. с англ. / М. Гэри, Д. Джонсон. – М. : Мир, 1982. – 416 с.
41. Якимахо, А.П. Управление объектами интеллектуальной собственности в РБ / А.П. Якимахо. – Минск : Амадфея, 2005. – 472 с.
42. Янсен, Ф. Эпоха инноваций / Ф. Янсен. – М. : ИНФРА-М, 2002. – 308 с.
43. Грушвицкий Р.И., Мурсаев А.Х. Угрюмов Е.П. Проектирование систем на микро-схемах с программируемой структурой. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006. – 736с.
44. Quartus II Handbook Version 10.1 [Electronic resource]. – 2008. – Mode of access: http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf. – Date of access: 21.10.2015.]
45. VHDL - обучающий портал [Электронный ресурс]. – 2015. – Режим доступа: www.bsuir.by/vhdl. – Дата доступа: 21.10.2015.
46. ModelSim — система HDL-моделирования цифровых устройств [Электронный ресурс]. – 2015. – Режим доступа: http://www.compitech.ru/html.cgi/archiv/02_06/stat_122.htm. – Дата доступа: 21.10.2015.

ГЛОССАРИЙ

- Аутентификация** – проверка соответствия пользователя по его особым характеристикам (отпечаток ладони, пальца, части сугубо личной информации и т. п.)
- 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.
- 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.
- Идентификация** – опознание пользователя на право доступа к системе (пароль и т. п.)
- 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.
- Ключ** – информация, необходимая для беспрепятственного шифрования и расшифрования текстов.
- 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 заместителя.
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.
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 команд	– совокупность частей, связанных общей функцией. – 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.
Шифрование	– процесс преобразования исходного текста, который носит также название открытого текста, в зашифрованный текст.
Численный метод	– метод приближенного или точного решения математических задач, основанный на построении конечной последовательности действий над конечным множеством чисел.
Электронная цифровая подпись	– присоединяемая к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения.
Язык алгоритмический	– средство точного формулирования вычислительных процессов для их последующей реализации на автоматических вычислительных машинах.

УЧЕБНОЕ ИЗДАНИЕ

*Головки Владимир Адамович
Дудкин Александр Арсентьевич
Матюшков Леонид Петрович*

ОСНОВЫ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

*Рекомендовано УМО по образованию в области информатики
и радиоэлектроники в качестве учебно-методического
пособия по учебным дисциплинам «Основы искусственного
интеллекта», «Традиционные и интеллектуальные
информационные технологии», «Автоматизация
проектирования вычислительных машин и систем»
для специальностей 1-40 03 01 «Искусственный интеллект»,
1-40 02 01 «Вычислительные машины, системы и сети»,
1-53 01 02 «Автоматизированные системы
обработки информации»*

2-е издание, исправленное и дополненное

Текст печатается в авторской редакции

Ответственный за выпуск: Матюшков Л.П.

Редактор: Боровикова Е.А.

Компьютерная вёрстка: Боровикова Е.А.

ISBN 978-985-493-347-4



9 789854 933474

Издательство БрГТУ.

Свидетельство о государственной регистрации
издателя, изготовителя, распространителя печатных
изданий № 1/235 от 24.03.2014 г.

Подписано к печати 17.03.2015 г. Формат 60×84 ¹/₁₆.

Бумага «Снегурочка». Гарнитура «Times New Roman».

Усл. п. л. 10,5. Уч.-изд. л. 11,25. Тираж 70 экз.

Заказ № 1253. Отпечатано на ризографе учреждения
образования «Брестский государственный технический
университет». 224017, Брест, ул. Московская, 267.